Virtual University	CS609-System Programming (Solved Macq's) For Final Term TOPIC 93 to 190 Also 70 to 90	VU LMS HELP vulmshelp.com
Junaidfazal08@gmail.com Bc190202640@vu.edu.pk	FOR MORE VISIT VULMSHELP.COME	JUNAID MALIK 0304-1659294



AL-JUNAID TECH INSTITUTE

1. The GetCommandLine() function returns the command line as a single_____
   - Float variable
   - Integer Array
   - Integer Variable
   - Character string Topic 101
2. Simple job management shell will allow _____ commands to run.
   - Two
   - Five
   - Three Topic 104 (jobbg,jobs,kill)
   - Four
3. In job management shell the shell uses ____ specific file keeping track of process ID and other related information
   - Shell
   - Process
   - System
   - User Topic 104
4. Process identify can be obtained from the _____ structure.
   - CreateProcess()
   - CREATE_PROCESS()
   - PROCESS_INFORMATION Topic 93
   - GetProcessInfo()
5. If the system is _____ time is multiplexed among multiple processes in an interleaved manner.
   - Uniprocessor system Topic 99
   - Linear processor system
   - Multicore processor system
   - Multiprocessor system
6. If a system is _____ then windows scheduler can run process threads on separate processors.
   - Uni processor
   - Linear processor
   - Single core processor
   - Multiprocessor Topic 99
7. FILETIME parameter of GetThreadTime() function is a ____ bit value.
   - 32
   - 8
   - 16
   - 64 Topic100
8. Closing the child process handle _____ the process.

- Does not create
- Create
- Destroy
- Does not destroy Topic 93

9. Closing the child process only closes the access of the _____.
   - All parent process
   - Parent process Topic 93
   - All children processes of parent process
   - Child process
10. In windows, there are _____ ways to get command line parameter for a process.
   - Two Topic 96
   - Four
   - Five
   - Three
11. Ip ApplicationName handle's value _____ be NULL.
   - Should
   - Should not According to me
   - May
   - May not
12. Control of a running fiber can be given to another fiber by using --------- function.
   - GetCurrentFiber()
   - SwitchToFiber() Topic 129
   - CreateFiber()
   - ConvertThreadToFiberEx()
13. A thread can enable fiber operation by calling _____function.
   - GetCurrentFiber()
   - SwitchToFiber()
   - CreateFiber()
   - ConvertThreadToFiberEx() Topic 129
14. Inherited handles are _____ copies that a parent and child might be accessing.
   - Connected
   - Similar
   - Distinct Topic 118
   - Related
15. A fiber can obtain its identity by calling _____function.
   - GetFiberData()
   - GetCurrentFiber() Topic 129
   - GetFiberId()
   - GetFiberIdentify()
16. The _____ function is used to obtain the process handle using the process ID.

- openProcesses()
- GetCurrentProcess()
- OpenProcess() Topic 93
- OpenprocessID()

17. Parent process usually creates a _____ handle if parent and child process require ____ access rights.
    - Duplication, same
    - Duplication, different Topic 94
    - New, same
    - New, different
18. There are ------ APIs to manage fiber.
    - 10
    - 9
    - 7 Topic 129
    - 8
19. One process can finish another process using the function _____
    - FinishProcess()
    - CreateProcess()
    - ExistProcess()
    - TerminateProcess() Topic 95
20. The simplest form of synchronization can be achieved through___ construct.
    - Run
    - Halt
    - Lock
    - Wait Topic 96
21. The single object for which the process waits is activated by _____.
    - lpHandle
    - bWaitAll
    - hHandle Topic 96
    - nCount
22. The array of handles for which the process wait is activated by _____
    - lpHandle Topic 96
    - hHandle
    - nCount
    - bWaitAll
23. _____is the number of objects in an array. Should not exceed MAXIMUM WAIT OBJECTS
    - lpHandle
    - hHandle
    - nCount Topic 96

- bWaitAll

24. _____is the timeout period for wait. 0 for no wait and INFINITE for indefinitewait.
    - IpHandle
    - hHandle
    - nCount
    - dwMilliseconds Topic 96
25. _____handle describes if it's necessary to wait for all the objects to get free
    - IpHandle
    - hHandle
    - nCount
    - bWaitAll Topic 96
26. How many The possible return values are___
    - 2
    - 3
    - 4
    - 5 Topic 95
27. The timeout period for wait() function is measured in _____.
    - Millisecond Topic 96
    - Kilo hertz
    - Mega hertz
    - Microsecond
28. Its not a good idea to use _____ within the program as it will not give it a chance to release resources.
    - createProcess()
    - TerminateProcess()
    - ExitProcess() Topic 95
    - FinishProcess()
29. A new fiber can be created by using_____
    - ConvertThreadToFiberEx()
    - ConvertThreadToFiber()
    - CreateFiber() Topic 129
    - GetFiber()
30. In Environment Block, IpName is a ____ name.
    - Stack
    - Array
    - Process
    - Variable Topic 97
31. In Environment Block, GetEnvironmentVariable() function returns _____ in case of failure.
    - 0 Topic 97

- NULL
- -1
- UNDEFINED

32. In Environment Block, PATH is an example of an environment _____
    - Stack
    - Array
    - Process
    - String Topic 97
33. An Environment Block is associated with _____ process(es) in the system.
    - Some
    - No
    - Each and every Topic 97
    - Exactly one
34. In Environment Block (EB), each string is _____
    - Five character long
    - Null terminated Topic 97
    - Undefined
    - Empty
35. The wait() function is limited to _____ handles.
    - 32
    - 8
    - 64 Topic 98
    - 16
36. A _____ value will cause a thread to move from the running state to the ready state.
    - Negative
    - INFINITE
    - 1
    - 0 Topic 127
37. Fiber which is shared to different threads_____
    - Should not access private data
    - Should not access global data
    - Should access thread specific data
    - Should not access thread specific data Topi 128
38. Fibers are scheduled by the _____
    - Hardware
    - Operating system
    - Application Topic 128
    - BIOS
39. IPC stands for
    - Inter process communication My Point of View

- Inter privacy communication
- Information process communication
- Information and privacy communication

40. The sleep function allows a thread to move the ____ to the ____ state.
    - Running, Suspended
    - Running, terminated
    - Running, ready
    - Running,wait Topi 127

41. The function SwitchToThread() provides a way for a thread to yield its processor to another _____ thread if there is one that is _____ to run.
    - Ready, Running
    - Running, Running
    - Ready, Ready Topic 127
    - Running, Ready

42. The time period in sleep function is specified in _____.
    - Seconds
    - Milliseconds My point of View
    - Nanoseconds
    - Microseconds

43. Default stack size for a thread is _____.
    - 1MB Topic 126
    - 1 Byte
    - 1 Bit
    - 1KB

44. A scheduler will move a _____ threads to the _____ state if the threads time slice expires without the thread waiting.
    - Waiting, Waiting
    - Running, Ready Topic 125
    - Running, Waiting
    - Waiting, Ready

45. When a thread is suspended it goes into _____
    - Terminated state
    - Waiting state
    - Suspended state
    - Ready state Topic 125

46. A thread that is ready but do not have required resource will go into _____
    - Running state
    - Suspended state
    - Waiting state My point of View
    - Execution state

47. The scheduler will run the _____ priority ____ thread when a processor becomes available.
    - Highest, ready Topic 125
    - Lowest, ready
    - Highest, running
    - Lowest, running
48. If there are no THREAD_PRIORTY_TIME_CRITICAL thread, the process will run _____ first.
    - THREAD_PRIORTY_LOWEST
    - THREAD_PRIORTY_NORMAL My point of View
    - THREAD_ORIORITY_BELOW_NORMAL
    - THREAD_PRIORTY_IDLE
49. Which of the following priority is the highest class in the thread?
    - THREAD_PRIORTY_ABOVE HIGHEST
    - THREAD_PRIORTY_TIME_CRITICAL My point of View
    - REALTIME_ORIORITY_CLASS
    - THREAD_PRIORTY_HIGHEST
50. The range of relative thread priorities is between_____.
    - 0 to 1
    - 0 to 4
    - -2 to +2 (+-2) Topic 123
    - -1 to +1
51. Which of the following thread priorities will run first?
    - Thread with THREAD_PRIORTY_ABOVE_NORMA My point of View
    - Thread with THREAD_PRIORTY_BELOW_NORMAL
    - Thread with THREAD_PRIORTY_ _IDEAL
    - Thread with THREAD_PRIORTY_NORMAL
52. In windows, most common processes have _____ priority class
    - HIGH_PRIORTY_CLASS
    - IDLE_PRIORTY_CLASS
    - NORMAL_PRIORTY_CLASS My Point View
    - REALTIME_PRIORTY_CLASS
53. The value returned using TIsSetValue() function is in the form of _____
    - BOOL My Point of View
    - INT
    - LPVOID
    - DWORD
54. In Thread Local Storage (TLS) arrangement, row represents _____ while column represent _____
    - Thread TLS Index Topic 123

- Process Number, TLS Index not sure
- TLS index, Thread
- Thread, process Number

55. Which of the following function frees the specified index numbers?
    - TIsClear()
    - TIsFree() Topic 123
    - TIsReSet()
    - TIsDelete()
56. API is used to allocate the index and it returns the TLS index in theform of the double word
    - TIsClear()
    - TIsAlloc() Topic 123
    - TIsReSet()
    - TIsDelete()
57. _____Provided valid indexes are used, The programmer can access TLS spaceusing these simple GET/SET APIs
    - TlsGetValue ()
    - Both A and C Topic 123
    - TlsSetValue ()
    - TIsDelete()
58. In the form of failure, TIsAlloc() function returns_____.
    - DWORD
    - 0
    - -1 Topic 123
    - BOOL
59. Which of the following is a correct statement?
    - Every worker work with multiple processors on multiple processor
    - Every worker work as a separate thread on a separate processor Topic 122
    - Every worker work as a separate thread on a single processor
    - Every worker work with multiple processors on single processor
60. Which of the following is an optimal situation?
    - After certain limit processor speed can also be enhanced
    - After certain limit processor speed cannot be enhanced Topic 122
    - Multiprocessing is not responsible for multiple flows of execution
    - Output of parallel program should not be same when it is serialized
61. Which of following statement is incorrect?
    - Program performance can be scaled without any certain limit.
    - Program performance can be enhanced by using multithreading.
    - Program performance can be enhanced with Parallelism.
    - Program performance scales automatically, up to some limit.

62. Four threads are created i.e. thread 0, thread 1, thread 2, and thread 3 and are running to sort a large file, select the most appropriate statement?
    - Wait for thread 1 to complete and merge it with thread 2.
    - Wait for thread 0 to complete and merge it with thread 2.
    - Wait for thread 1 to complete and merge it with thread 0. My point of view
    - Wait for thread 2 to complete and merge it with thread 0.
63. Why a file is always mapped before accessing it?
    - To access it just like accessing some data from main memory My point of view
    - To make it secure
    - To reuse it any time
    - To access it in the form of LinkedList
64. Once all the threads are created, then can be run using _____ function.
    - ResumeThread()
    - ReadyThread()
    - RunThread() Topic 121
    - StartThread()
65. The basic difference between boss-worker thread model and client-server model is _____
    - In boss worker all the thread are run at the boss's end, but in client-server model, each client run a different thread
    - In the client-server all the thread are run at the server end, but in the boss worker model, each worker runs different thread Conceptual
66. Which of the following is not a thread model?
    - Peer-to-peer Topic 119
    - Pipeline
    - Boss-worker
    - Client-server
67. In client server model, rather than_____ work is done _____
    - Concurrently, sequentially
    - Problematic, Efficiency
    - Sequentially, Concurrently Topic 119
    - Linear, Straight
68. In which of the following models, work moves from one thread to the next thread?
    - Peer-to-peer Model
    - Pipeline Model Topic 119
    - Boss-worker Model
    - Client-server Model
69. The _____ flag is set to be _____ in the CreateProcess() function, which will determine whether child process will inherit copies of parent open handles.
    - bInheritFlag, FALSE

- bInheritHandle, TRUE My point of View
- bInheritFlag, TRUE
- bInheritHandles, FALSE

70. Inherited handles are _____ copies that a parent and child might be accessing
    - Similar
    - Connected
    - Distinct Wikipedia
    - Related
71. The get.JobNumber() function looks into the file for a vacant place. If no place is available, It appends a new record at the_____
    - Next
    - Start
    - Middle
    - End My point of View
72. In getting a job number, the job Management Shell uses a number of job management functions to manage jobs information within the _____ life.
    - Process
    - User Topic 103
    - Shell
    - System
73. Fiber management occurs at the _____ level.
    - Hardware
    - User space Topic 128
    - Kernel
    - BIOS
74. In Asynchronous input/ Output _____.
    - Input from the keyboard will be taken after the song is ended.
    - Song cannot be played until the other operations are not completed.
    - Song can be played with other operations being executed same time.

      Topic 119 also Conceptual (correct Accordingly)

    - Two operations cannot be run at the same time.
75. The first two parameters in argc will be _____ and ____ respectively.
    - Process name, pattern which is to be searched Topic 118
    - Inputted file names, pattern to be searched
    - Pattern to be searched, process name
    - Inputted file names, process name
76. Total number of files inputted, can be obtained with _____-
    - Argc-1
    - Argc-2 Topic 118 (Confusion kindly Correct Accordingly)

- Argc-3
- Argc

77. C Library Threading functions are _____ than windows library functions but not _____
    - Simpler, Hard
    - Hard, Simpler
    - Diverse, Simpler
    - Simpler, Diverse Topic 118
78. LIBCMT is a _____
    - BIOS Library for thread
    - Java Library for threads
    - C Library for threads Topic 118
    - Windows Library for threads
79. _____function is used to extract ____ from a string.
    - gettok(), Token
    - getTok, String
    - strtok(), String
    - strtok(), Token Topic 118
80. The return type of _beingthreadex() function _____
    - is HANDLE, but we need to type caste it for further processing.
    - Is not HANDLE, but we cannot type caste it for further processing.
    - is not HANDLE, we need to type caste it for further processing.

      Topic 118 Conceptual
    - is HANDLE, but we don't need to type caste for further processing.
81. _____ library function are thread safe than _____ library functions.
    - C, java
    - Java, C
    - C, Windows Topic 118
    - Windows, C
82. Windows treats threads as _____.
    - Objects Topic 116
    - Threads
    - Processes
    - Classes
83. When a thread exits, the thread _____ is deallocated and the handle referring to the thread in invalidated.
    - Stack Topic 113
    - List
    - Queue
    - Array
84. ResumeThread() function will _____.

0304-1659294 www.vulmshelp.com

- Decrease the value of suspend count Topic 112
- Reset the value of suspend count
- Increase the value of suspend count
- Not affect the value of suspend count

85. By default, the value of Suspend count, when creating a thread, is_____.
    - 2
    - 1
    - 0 Topic 112
    - -1
86. Which of the following version of windows was not compatible with GetProcessIdOfThread() function?
    - Windows 2003
    - Windows 7
    - Windows NT
    - Windows XP Topic 115
87. Which of the following function can be used to map a process with a thread?
    - GetProcessIdOfThread() My Point of View
    - DWORD ResumeThread (HANDLE hThread)
    - DWORD SuspendThread (HANDLE hThread)
    - GetThreadOPendingFlag()
88. A thread will execute if and only if its Suspend Count value is _____
    - Non zero
    - Above 1
    - 0 Topic 115
    - 1
89. Threads can also be treated as parent and child although the _____ is unaware of that.
    - Kernel
    - Thread
    - Program
    - Operating system Topic 112
90. Threads share resources within a _____
    - Code
    - Program
    - Thread
    - Process Topic 110
91. Threads uses the space assigned to a _____
    - Thread
    - Process Topic 111
    - Code
    - Program

0304-1659294 www.vulmshelp.com

92. In a multi-threading system, multiple threads may exist within a single _____
    - Thread
    - Process Topic 109
    - Code
    - Program
93. _____ is an independent unit of execution within a process.
    - Thread Topic 109
    - Process
    - Code
    - Program
94. A job object is used to _____ process execution time and obtain user time statistics.
    - Limit Topic 108
    - Write
    - Read
    - Open
95. The getJobNumber() function looks into the file for a vacant place. If no place is available, it appends a new recover of the file.
    - Start
    - Next
    - Middle
    - End My Point of View
96. In job Management Shell, the shell uses _____ specific file keeping track of process ID and other related information.
    - User Topic 104
    - Process
    - Shell
    - System
97. In finding a Process ID, the FindProcessId() obtains the process ID of the given job number. It simply looks on job number and _____ the record at the specific location.
    - Sums
    - Executes
    - Reads Topic 106
    - Writes
98. Simple job management shell will allow _____ commands to run.
    - Two
    - Three Topic 104 (Jobbg, jobs, kill)
    - Four
    - Five
99. In listing background jobs, the job management function uses display jobs () function. This function _____ a file

- Run
- Start
- Process
- Open Topic 105

100. In Job Objects, firstly a job object is created using CreateJobObject(). It _____ a name and security attributes.
- Writes
- Closes
- Uses Topic 107
- Throws

101. Control of a running fiber can be given to another fiber by using _____ function.
- GetCurrentFiber()
- SwitchToFiber() Topic 129
- CreateFiber()
- ConvertThreadToFiberEx()

102. A thread can enable fiber operation by calling_____ function.
- GetCurrentFiber()
- SwitchToFiber()
- CreateFiber()
- ConvertThreadToFiberEx() OR ConvertThreadToFiber() Topic 129

103. Fiber can obtain its identity by calling _____ function.
- GetFiberData()
- GetCurrentFiber() Topic 129
- GetFiberId()
- GetFiberIdentity()

104. There are _____ APIs to manage fibers.
- 10
- 9
- 7 Topic 129
- 8

105. A new fiber can be created by using_____
- ConvertThreadToFiberEx()
- ConvertThreadToFiber()
- CreateFiber() Topic 129
- GetFiber()

106. The wait() function is limited to _____ handles
- 32
- 16
- 8
- 64 Topic 98

0304-1659294 www.vulmshelp.com

107. Windows OS does not have any structure that keeps track record of the _____ processes.
- Child
- Grand-child
- Parent
- Parent-Child My point of View

108. In comparison of DLL, executable library files are linked at _____ time.
- Link
- Run
- Compiler My Point of View and GOOGLE
- Load

109. Interlocked functions are _____to use.
- Simpler, Faster but hard
- Simpler, Faster and easy Topic 135
- Not simpler but faster and easy
- Simpler easy but slow

110. Thread should not change the _____ environment.
- Process Topic 137
- Hard disk
- Integer
- Ram

111. Which of the following function can be used to map the process with thread?
- DWORD Resume Thread (HANDLE hThread)
- DWORD Suspend Thread (HANDLE hThread)
- GetProcessIdOfThread() My Point of View
- GetThreadIOPendingFlag()

112. Mutexes, Semaphores, Events, and Critical Section are the four synchronization objects provided by _____.
- Software
- RAM
- Windows Topic 138
- processor

113. Like every other resource, threads are also treated as_____
- Code
- Object Topic 112
- Program
- Thread

114. The process Execution times uses the API GetCommandLine() to get the command line as a single _____.
- Bool

0304-1659294 www.vulmshelp.com

- Float
- String Topic 102
- None

115. lpApplicationName _____ be NULL.
- May
- Should
- Should not My point of View
- May not

116. Volatile stage is a _____ level facility.
- Compiler or windows Topic 133
- Programmer
- BIOS
- Hardware

117. CRITICAL_SECTION Objects do not have_____.
- Loops
- Handler Topic 139
- Variables
- Process

118. CRITICAL_SECTION Objects do not have___and are not shared among the____.
- Handler
- Both A and C Topic 139
- process
- Process

119. _____ variables should not be accessible globally.
- String
- Locally required Topic 139
- Long
- Integer

120. Thread IDs and handles can be obtained using functions quite similar to the ones used with_____
- Process Topic 114
- Program
- Thread
- Object

121. A thread can be signaled from which of the following?
- ResumeThread()
- CloseThread()
- CreateThread()
- ExitThread() OR TerminateThread() Topic 117

0304-1659294 www.vumshelp.com

122. If there are 129 objects, how many times we will have to call the wait () function?
- 4
- 6
- 3 My Point of View
- 2

123. ResumeThread() function will ___
- Increase
- Decrease Conceptual
- None of given
- Both

124. If you want to create 4 thread i.e, thread 0, thread 1, thread 2, and thread 3, they all must be created at _____ state.
- Suspended Topic 120
- Blocked
- Running
- Ready

125. It must be ensured that _____ thread(s) is/ are not modifying ____ data at same time.
- Many, Many
- 1, Many
- Many,1 Topic 131 Conceptual
- 1,1

126. The result of concurrent processing and normal processing _____
- Yield same output
- Is always different
- Is always same
- May differ in some cases My point of View

127. In memory architecture and barriers, before writing the value s back to memory, the processor usually keeps them in _____
- Hard disk
- Memory card
- Processor
- Cache Topic 134

128. _____ are used to ensure that memory is accessed in the desired order.
- Memory barriers or memory fences Topic 134
- Memory paths or memory houses
- Multi processors
- Memory blocks or memory chains

129. Turning off _____ may adversely affect the performance but sometimes may _____ the program.

- Usability, faster
- Optimization, slow Topic 133
- Performance, slow
- Security, faster

130. Turning off Optimization may adversely affect the performance but sometimes may slow the program. ANSI C provides a qualifier _____ for this purpose.
- Volatile Topic 131
- Non-volatile
- Both A and B
- None of Given

131. The volatile qualifier does not guarantee that the modifications will be visible to _____ in a desired order.
- Processors Topic 134
- User
- Ram
- Hard disk

132. The _____ functions provide memory barriers.
- Interlocked Topic 139
- User define
- Interchange
- Intermediate

133. Interlocked increment () takes a 32-bit signed variable as an arguments, which should be placed in memory at the _____ boundary.
- 4-byte Topic 135
- 8-byte
- 16-byte
- 2-byte

134. If a variable with a volatile scope only needs to be incremented, decremented, or exchanged, interlocked functions are the _____ choice.
- Last
- Best Topic 135
- 2nd last
- Worst

135. _____are most suited if variable with volatile scope only need to be incremented,decremented and exchanges
- Interlocked functions Topic 135
- Local
- Global
- None

136. The most basic of Interlocked Functions

- Interlockedincrement()
- InterlockedDecrement()
- Both A and B Topic 135
- None of Given

137. How many Basic Function of Interlocked()
- 1
- 2 Topic 135
- 3
- 4

138. If any needed variables are not initialized, then create _____ threads until variables are initializes.
- Suspended Topic 137
- Double
- Slow
- Fast

139. In thread safe code, _____ conditions are avoided.
- Race Topic 137
- Loop
- Local
- Global

140. When more than one thread can run the same code without introducing synchronization issues, it is said to be _____
- Thread safe Topic 137
- Local
- Threaded
- Initialized properly

141. InterlockedIncremen64() and interlockedDecrement64() can be utilized on 64-bit systems if the addend is paced at the _____ boundary.
- 8-byte Topic 135
- 4-byte
- 2-byte
- 16-byte

142. _____ have a performance disadvantage since they create a memory barrier.
- Interchanged function
- Interlocked function Topic 135
- Intermediate function
- Interlocked memories

143. If a _____ is used to store thread-specific data, other threads will also use it.
- Small variable
- Large variable

- Global variable topic 136
- Local variable

144. Make a variable_____ if we know it includes thread-specific information.
- Static
- Local Topic 136
- Global
- Random

145. According to the criterion for good thread code, _____ storage should not be used for the purpose of local storage.
- Hard disk
- Global Topic 136
- Cache
- Ram

146. If we know the information in a variable will be used by other threads, we can make it _____.
- Global Topic 136
- Local
- Main
- Auto

147. When using thread synchronization objects, there are always inherent risks associated with them such as _____
- Lower power
- Multi clocks
- Load
- Deadlocks Topic 137

148. A thread waits for other threads to ____ using the waitForString0bject() or waitForMultipleObjects() methods.
- Wait
- Terminate Topic 138
- Start
- Update

149. The part of the code that can only be executed by one thread at a time is referred to as the _____
- Process section
- Loop section
- Top section
- Critical section Topic 139

150. Only _____ can be in the CRITICAL_SECTION variable at a time
- Odd threads
- Many threads

0304-1659294 www.vumshelp.com

- One thread Topic 139
- Even threads

151. Which function is used to initialized the CRITICAL_SECTION.
- InitializeCriticalSection() Topic 139
- DeleteCriticalSection()
- Both A and B
- None of Given

152. Which function is used to Delee the CRITICAL_SECTION.
- InitializeCriticalSection()
- DeleteCriticalSection() Topic 139
- Both A and B
- None of Given

153. Multiple threads may call EnterCriticalSection() but only _____ thread is allowed to enter the critical section while the rest areblocked.
- One Topic 139
- Two
- Three
- Foure

154. A call to LeaveCriticalSection() must match a/ an____
- EnterSection()
- Variables
- EnterCriticalSection() Topic 139
- CriticalSectionPart()

155. CRITICAL_SECTION object do not have _____
- Process
- Variables
- Handles Topic 139
- Loops

156. EnterCriticalSection() can be called by ____ but only one of them is allowed to enter the critical section, while the others are blocked.
- Two threads
- One thread Topic 139
- Half thread
- Many threads

157. Using the critical section construct is ____ and intuitive.
- Easy Topic 140
- Difficult
- Short
- Long

0304-1659294 www.vumshelp.com

158. It would be ____ to use different objects in the same thread or across numerous threads that share the same data.
- Easy
- Incorrect Topic 141
- Correct
- Difficult

159. For mutual exclusion to work, _____ variables must be protected by a single object across all threads.
- Integer
- Important
- Shared Topic 141
- specific

160. Within the critical section, all the variables must be guarded by _____
- Critical object
- All objects
- A single object Topic 141
- Double object

161. What is the limit for object to be waited for waitForMultiple Objects () function in window?
- 63
- 65
- 62
- 64 Topic 141

162. Producer consumer problem is a classical problem in_____
- Critical Section
- Conical collection
- Mutual Exclusion Topic 142
- None of given

163. The producer periodically _____ a message.
- Creates Topic 142
- Delete
- Update
- None

164. The producer also computes a simple_____ of the message
- Checksum Topic 142
- Contents
- Update
- None

165. _____is a short form of Mutual Exclusion
- Mechanism

0304-1659294 www.vulmshelp.com

- Checksum
- Mutex Topic 143
- All of the given

166. Which of the following Windows functions used to manage mutexes.
- CreateMutex()
- ReleaseMutex()
- OpenMutex()
- All of the given Topic 144

167. Home Many windows Function are used to manage mutex.
- 1
- 2
- 3 Topic 144
- 4

168. lpMutexName is the name of the mutex.
- True Topic 144
- False

169. OpenMutex() is used to _____ and _____ named mutex
- Existing,open
- Open,Existing Topic 144
- Open,Delete
- Exsiting,Delete

170. However, abrupt termination of a thread indicates a serious programming flaw. Mutex waits can_____
- Time In
- Time Out Topic 146
- Has
- Does Not

171. If NULL is returned it indicates dose not failure.
- True
- False Topic 144

172. Semaphore maintains a count
- True Topic 147
- False

173. A semaphore is in a signaled state when the count is greater than.
- 1
- -1
- 0 Topic 147
- None of given

174. Semaphore is_____ when the count is 0.
- Signaled

- Un-signaled Topic 147
- Release
- None of the given

175. Which of the following are used in semaphore Count?
- CreateSemaphore()
- CreateSemaphoreEx()
- OpenSemaphore()
- All of the Given Topic 147

176. cReleaseCount gives the count after the release and must be greater than____.
- 1
- -1
- 0 Topic 147
- None of given

177. Events are classified as either
- manual-reset
- auto-reset
- Both A and B Topic 150

178. A _____ event can signal several waiting threads simultaneously and can be reset.
- manual-reset Topic 150
- auto-reset

179. An _____event signals a single waiting thread, and the event is reset automatically.
- manual-reset
- auto-reset Topic 150

180. If bInitialState is TRUE, the event is set to a_____state
- Signaled Topic 150
- Un-Signaled

181. If the event is manual-reset, it remains signaled until a thread explicitly calls ResetEvent().
- True Topic 150
- False

182. How many ways to use Events?
- 1
- 2
- 3
- 4 Topic 151

183. Windows operating system is a multithreaded _____ that provides support for real time applications and multiprocessors.
- Hardware

0304-1659294 www.vulmshelp.com

- User space
- Kernel Topic 153
- BIOS

184. Program design and performance can be simplified and improved by
- Odd threads
- Many threads
- Threads Topic 153
- Even threads

185. _____Synchronization is a way to coordinate processes that use shared data
- Shell
- Process Topic 153
- System
- User

186. Which of the following Synchronization objects.
- Critical Section
- Semaphore
- Mutex
- All of the Given Topic 153

187. The_____functions provide a simple mechanism for synchronizing access to a variable that is shared by multiple threads.
- Interlocked Topic 155
- Local
- Global
- None

188. Memory allocation is then performed using using _____rather than using_____
- HeapAlloc() and HeapFree(),malloc() and free() Topic 56
- malloc() and free() and HeapAlloc() and HeapFree(),

189. Each thread that performs memory management can create a Handle to its own heap using _____
- HeapCreate() Topic 156
- HeapAlloc()
- HeapFree()
- None of the given

190. Threads allocate memory and free memory using _____functions respectively
- malloc () and free () Topic 156
- free () and malloc ()

191. How may aspects compare the performance on the Basis.
- 1
- 2

- 3 Topic 157
- 4

192. Which of the Following aspects that are used to compare the Performance.
- Real-Time
- User-Time
- System-time
- All of the given Topic 157

193. MX(Mutexes) version costs more than _____ times than IN
- 2 times
- 2 to 30 times Topic 158
- 1 to 30 times
- 3 times

194. CS (Critical Section) version costs _____ times more than IN
- 2 to 30 times
- 2 times Topic 158
- 1 to 30 times
- 3 times

195. For older version of windows CS was not scalable.
- True Topic 159
- False

196. Critical Section works in_____
- Hardware
- User space Topic 161
- Kernel
- BIOS

197. How many methods of Lightweight Reader Writer Locks.
- 1
- 2 Topic 163
- 3
- 4

198. Which of the following methods of Lightweight Reader Writer Locks
- Exclusive Mode
- Shared Mode
- Both A and B Topic 163
- None of given

199. How many APIs which are used to access SRWs in Exclusive mode.
- 1
- 2 Topic 164
- 3
- 4

200. When a thread is created almost _____space is reserved for that thread and with increasing threads memory area is piled up.
- 1 MB Topic 166
- 1 KB
- 1 Bit
- 1 Byte

201. If you_____ the threads the you can lose the benefits of parallelism and synchronization
- Minimize Topic 166
- Maximize

202. Which of the following optimization technique?
- Use of semaphore throttles
- Asynchronous I/O
- Using I/O completion ports
- All of the Given Topic 166

203. Every process has a dedicated_____ pool.
- Odd threads
- Many threads
- Threads Topic 172
- Even threads

204. The thread pool is used to _____the number of application threads and provide management of the worker threads
- Increased
- Reduce Topic 172
- Constant
- Equal

205. How many types of call back Function.
- 1
- 2 Topic 173
- 3
- 4

206. How many sorts of parallelism?
- 1
- 2 Topic 177
- 3
- 4

207. Which of the following parallelism define every loop iteration can execute concurrently.
- Loop parallelism Topic 177

0304-1659294 www.vulmshelp.com

- Fork-join parallelism

208. _____the control flow divides (like the shape of the fork) into multiple flows that join later
- Loop parallelism
- Fork-join parallelism Topic 177

209. Each process has its own process affinity mask and a system affinity mask.
- True Topic 180
- False

210. How many Bits vector of MASK?
- 1
- 2
- 3 Topic 180
- 4

211. Which of the follwing Bits vector of MASK?
- System Affinity Mask
- Process Affinity Mask
- Thread Affinity Mask
- All of the Given Topic 180

212. IPC stand for_____
- Internet process control
- Inter-process Communication Topic 180
- Inter-parameter control
- None of the Given

213. How many Types of Pipe?
- 1
- 2 Topic 181 (Anonymous Pipes, Named Pipes)
- 3
- 4

214. File-like object called Pipe can be used for IPC.
- True Topic 181
- False

215. Which of the following are correct statement of Anonymous Pipes?
- Simple anonymous pipes are character based and half duplex. Topic 181
- They allow network wide communication.

216. Which of the following are correct statement of Named Pipes?
- There can be multiple open handles for a pipe
- They allow network wide communication.
- Much more powerful than anonymous pipes
- All of the Given Topic 181

0304-1659294 www.vulmshelp.com

217. Anonymous pipes allow _____(half-duplex) communication
- one-way Topic 182
- two-way

218. The default set of cbPipe is____
- 0 Topic 182
- 1
- 2
- 3

219. Named Pipe is____
- Directional
- Bi-Directional Topic 184
- Tri-Directional
- None of the Given

220. The pipe name would be like this.
- \\.\pipe\[path]pipename Topic 187
- \\servername\pipe\pipename

221. Which of the following connection sequences for the Client.
- The client connects with a server
- Communicates with the server using CreateFile()
- Performs Read and Write operations and ultimately disconnects
- All of the given Topic 190

222. Which of the following connection sequences for the Server.
- Communicates with the client.
- As a result ReadFile() returns FALSE
- The server-side connection is disconnected
- All of the given Topic 190

223. Which of the following Returns information whether the pipe is in blocking or non-blocking mode, message oriented or byte oriented, number of pipe instances, and so on.
- SetNamedPipeHandleState()
- GetNamedPipeHandleState() Topic 188
- GetNamedPipeInfo()
- None of the given

224. Which of the following Allows the program to set the same state attributes. Mode and other values are passed as reference so that NULL can also be passed indicating no change is desired.
- SetNamedPipeHandleState() Topic 188
- GetNamedPipeHandleState()
- GetNamedPipeInfo()
- None of the given

225. Which of the following Determines whether the handle is for client or a server, buffer sizes, and so on.
- SetNamedPipeHandleState()
- GetNamedPipeHandleState()
- GetNamedPipeInfo() Topic 188
- None of the given

226. Call_____to disconnect from the handle
- DisconnectNamedPipe() Topic 189
- WaitNamedPipe()
- ConnectNamedPipe()
- All of the given

227. _____ is used to synchronize connections to the server
- DisconnectNamedPipe()
- WaitNamedPipe() Topic 189
- ConnectNamedPipe()
- All of the given

228. _____are the security attributes as discussed previously
- lpSecurityAttributes Topic 186
- nOutBufferSize
- dwOpenMode
- nMaxInstance

229. _____and _____ give the size in bytes of input and output buffer
- nOutBufferSize and nInBufferSize Topic 186
- nInBufferSize and nOutBufferSize

230. which of the following determines maximum number of pipe instances?
- lpSecurityAttributes
- nOutBufferSize
- dwOpenMode
- nMaxInstance Topic 186

231. which of the following indicates whether writing is message oriented or byte oriented?
- dwPipeMode Topic 186
- nOutBufferSize
- dwOpenMode
- nMaxInstance

232. _____ is the default timeout period.
- nDefaultTimeOut Topic 186
- nOutBufferSize
- dwOpenMode
- nMaxInstance

233. The period (.) stands for local machine.
- True Topic 186
- False

234. In mutex which type of data structure that stores the resource should also be used to store the mutexes because mutexes correspond to the resources.
- Different Topic 154
- Same
- Equal
- None of the Given

235. Only the_____ Scheduler decides which thread has the priority according to its scheduling policy.
- Hardware
- Application
- OS Topic 154
- BIOS

236. The shell uses a …… file keeping track of process ID and other related information?
- **User-specific Topic 104**

237. The array of handles for which the process waits.
- **lpHandle**

238. In Listing Background Jobs the job management function used for the purpose is DisplayJobs(). The function _____ the file.
- Process
- Runs
- Opens Topic 105
- Starts

239. In using Job objects, job objects are used to _____ process execution time and obtain user time statistics.
- Open
- Limit Topic 108
- Read
- White

240. FILETIME is a(an) _____-bit value. GetThreadTime()can be used similarly and required a hread handle.
- 32
- 64 Topic 100
- 16
- 8

241. In Finding a process Id, the FindProcessId() obtains the process Id of given job number. It simply looks up into the File based on the job number and _____ the record at the specific location.

0304-1659294 www.vulmshelp.com

- Executes
- Reads Topic 106
- Writes
- Sums

242. Thread Issues Threads Share Resources Within a_____. One Thread May inadvertently another Threads' Data.

- Process Topic 10
- Threads
- Program
- Code

243. An Environment Block is associated with_____process.

- Some
- One
- Each Topic 97
- No

From 70 to 90 TOPIC

244. When a fixed size data structure is allocated from a single heap, it reduces_____

- Fragmentation
- Errors
- Memory density
- Throughput

245. The heapReAlloc() API has _____ parameter(s).

- 4
- 1
- 3
- 2

246. The heapAlloc() API has _____ parameter(s).

- 3
- 4
- 2
- 1

247. When a heap (logical structure) is created the memory is _____ allocated at the program.

- Partially
- Completely

0304-1659294 www.vulmshelp.com

- Not directly
- Directly

248. _____ are the APIs for heap memory allocation.

- Heapcreate ()and HeapRealloc()
- Allocheap () and HeapRealloc()
- HeapAlloc() and HeapRealloc()
- HeapAlloc() and HeapRealloc()

249. For a non growable heap, the value of dwbytes in heap memory alloction is _____

- 0*7FEE8
- 0*7FDD8
- 0*AAAA8
- 0*7FFF8

250. _____ is the first step to allocate heap in a program.

- HeapDestroy()
- HeapFree()
- Release and handle
- Get heap handle

251. The function heapSize() returns the size of a block, or _____ in case failure.

- NULL
- 1
- -1
- 0

252. _____ is used to deallocate the entire heap.

- HeapDestroy()
- HeapFree()
- HeapTruncate()
- HeapDelete()

253. Sorting is performed in the _____

- RootHeap
- RecHeap
- ProcHeap
- NodeHeap

254. _____ stores the root address.

- RootHeap
- RecHeap
- ProcHeap

0304-1659294
www.vulmshelp.com

- NodHeap

255. The NodeHeap maintains a _____
- Data
- Data structure
- Record
- Root

256. There are _____ parameters taken by the HeapCreate() API.
- 3
- 4
- 2
- 1

257. Which of the following is the correct windows API for accessing heap?
- INT GetProcessHeap(VOID)
- VOID GetProcessHeap(HANDLE)
- HANDLE GetProcessHeap(VOID)
- INT*GetProcessHeap(VOID)

258. When a fixed size data structure is allocated from a single heap, it reduces _____
- Memory density
- Errors
- Throughput
- Fragmentation

259. The parameters “flOption” in the HeapCreate() API is a combination of _____ flafs.
- 1
- 2
- 4
- 3

260. While using CreateFileMapping(), _____ allow the mapping object to be secured.
- INVALID_VALUES
- PSECURITY_ATTRIBUTES
- LPSECURITY_ATTRIBUTES
- INVALID_HANDLE_VALUES

261. While using CreateFileMapping(),setting lpMapName to _____ disables the map sharing.
- -1
- NULL

0304-1659294
www.vulmshelp.com

- 0
- 1

262. _____ is the API for file mapping objects.

- Create_File_Mapping()
- CreateFileMapping()
- FileCreateMapping()
- MakeFileMapping()

263. Which of the following are the number of parameters taken by CreateFileMapping()?

- 7
- 6
- 5
- 4

264. The _____-- flag is set to be_____ in the CreateProcess() function, which will determine whether child process will inherit copies of parent open handles.

- blanheritFlag, TRUE
- blnheritHandles, FALSE
- blanheritFlag,FALSE
- bInheritHandles, TRUE

265. IPC stands for_____.

- Information and privacy communication
- Inter privacy communication
- Information process communication
- Inter Process Communication

266. Inherited handles are _____ copies that a parent and child might be accessing.

- Connected
- Similar
- related
- Distinct

267. Process IDs are always _____.

- Frequent
- Repeated
- Constant
- Unique

268. The process obtains environment and other information from_____ call.

- CreateThread()
- GetEnvironmentinfo()

0304-1659294 www.vulmshelp.com

- ❖ Getinfo()
- ❖ CreateProcess()

269. lpApplicationName handle's value_____ be NULL.

- ❖ May not
- ❖ May
- ❖ should
- ❖ Should not

270. In windows there are _____ ways to get command line parameters for a process.

- ❖ Five
- ❖ Four
- ❖ Two
- ❖ Three

271. Windows OS does not have structure that keeps track record of the _____ processes.

- ❖ Child
- ❖ Grand –child
- ❖ Parent
- ❖ Parent_Child

272. The most fundamental process management function in windows is CreateProcess() that has_____--- parameters.

- ❖ 12
- ❖ 6
- ❖ 4
- ❖ 10

273. The process can share memory and files but the process itself lie an individual _____ memory space .

- ❖ Non_volatile
- ❖ Physical
- ❖ permanent
- ❖ Virtual

274. Thread Local Storage (TLS) is an array of collection of pointers enabling a thread to _____ storage to create its unique data environment.

- ❖ De-allocate
- ❖ Clear
- ❖ Re-allocate
- ❖ Allocate

275. Each thread has its own_____.

AL-JUNAID TECH INSTITUTE
0304-1039294
www.vulmshelp.com

- TLS
- Environment Block
- Stack
- TLS and Stack

276. The process of DLL detachment in explicit linking is invoke by_____ function call.

- Free()
- freeLib()
- Flibra
- FreeLibrary()

277. Information regarding DLLs is placed in the _____data structure.

- dwBuilderNumber
- dwPlatform
- MAJORVERSION
- DLLVERSION

278. LoadLibrary() and LoadLibraryEx() should never be called from_____ as it will create more DLL entry Points.

- ThreadLibrarycalls()
- DllMinFunc()
- DisableThreadLibraryCalls()
- DllMain()

279. LoadLibraryEx() can suppress the execution of entry point, in _____-- linking of DLL.

- Implicit
- Static
- Dynamic
- Explicit

280. "Application that require newer updated functionality may sometime link with older DLL version". This statement refers to _____ of DLL versioning

- Strength
- Advantages
- Caution
- Problem

281. If entry point of DLL is not specified, then it is an example of _____-- linking.\

- Explicit
- Dynamic
- Hard

0304-1659294 www.vulmshelp.com

- Implicit

282. In case of_____ linking the DLL attaches at the time of process start and detaches when process ends

- Explicit
- Dynamic
- Hard
- Implicit

283. Explicit linking requires the program to explicitly specify the DLL to be _____.

- Freed
- Loaded
- Loaded and freed
- Ans: Loaded or freed

284. In a pointer function declaration for DLL explicit linking, HMODULE is NULL in case of_____.

- Execution
- Waiting
- success
- Failure

285. Once the DLL is loaded, the programmer needs to obtain _____ into the DLL for an entry point.

- Dynamic address
- Physical address
- Bus address
- Procedure Address

286. We write and _____ function in DLL and invoke them explicitly

- Compile
- Encrypt
- decrypt
- Encapsulate

287. In DLLs the executable library files are linked at_____ time

- . Ans: Compile

288. Each DLL program will have its own copy of _____ variables.

- Ans: Globle

289. In_____ operating system DLLs are used to invoke all kernel services.

- Ans: Windows

290. Dynamic memory is allocated from the

- Cache

0304-1639294 www.vulmshelp.com

- ❖ Paging file
- ❖ Stack
- ❖ Static memory

291. Which of the following is recommended to use while dealing with memory mapped file to look for EXCEPTION_IN_PAGE_ERROR exception?

- ❖ ESH exception handling
- ❖ SHE exception handling
- ❖ HE exception handling
- ❖ HES exception handling

292. To create a file mapping object, we have to declare _____ maximum parameters>

- ❖ 4
- ❖ 2
- ❖ 6
- ❖ 8

293. It is not possible for a system to map a file greater than _____ Into virtual memory space, while using Win32 OS.

- ❖ 2GB
- ❖ 3MB
- ❖ 3GB
- ❖ 2MB

294. It is much_____- to sort large data available in memory rather than in files.

- ❖ Harder
- ❖ Costly
- ❖ Unyielding
- ❖ Easier

295. qsort() is a _____ function.

- ❖ Standard library
- ❖ EXE
- ❖ Windows DLL
- ❖ User defined

296. When we create a file mapped object for sorting 1000 numbers in a file recorder will be saved in a/an_____-.

- ❖ Heap
- ❖ Stack
- ❖ Queue
- ❖ Array

0304-1659294 www.vulmshelp.com

297. Which of the following are the number of parameters taken by MapViewFile()?

- 2
- 4
- 3
- 5

298. While using MapViewOfFile(), which of the following are the three commonly used flags?

- FILE_WRITE, FILE_READ, AND FILE_ALL_ACCESS
- FILE_MAP_WRITE, FILE_MAP_READ,AND FILE_MAP_ALL_ACCESS
- MAP_WRITE,MAP_READ, AND MAP_AL_ACCESS
- WRITE, READ, AND ALL_ACCESS

299. _____ and _____ specify the starting address of the file from where the mapping starts.

- High, low
- dwFileHigh,dwFileLow
- dwFileOffsetHigh, dwFileOffsetLow
- dbFileOffsetHigh, dbFileOffsetLow

300. unmapViewOfFile() takes _____argument(s)

- 2
- 0
- 4
- 3

301. _____ Is the API for file mapping objects.

- MakeFileMapping()
- CreateFileMapping()
- FilecreateMapping()
- Create_file_Mapping()

302. While using CreateFileMapping(),_____ refers to the paging file.

- LPSECURITY_ATTRIBUTES
- PSECURITY_ATTRIBUTES
- INVALID_HANDLE_VALUES
- INVALID_VALUES

303. While using CreatFileMapping(),_____ allows the mapping object to be secured.

- LPSECURITY_ATTRIBUTES
- PSECURITY_ATTRIBUTES
- INVALID_HANDLE_VALUES

- INVALID_VALUES

304. While using CreateFileMapping(), setting IpMapName to_____ disables the map sharing.

- 0
- 1
- -1
- NULL

305. DLL stand for

- Direct layout library
- Dynamic link library
- Dynamic layout library
- Direct link library

306. The approach to gather all the source code and library functions after encapsulation into a single executable file, is called as _____

- Process linking
- Static linking
- Dynamic linking
- Thread linking

307. Each DLL program will have its own copy of ____ variables.

- Global
- Local
- Dynamic
- Static

308. In _____ operating system DLLs are used to invoke all kernel services.

- Windows
- Unix
- Linux
- Solaris

309. In DDLs the executable library files are linked at ____ time.

- Link
- Run
- Compile
- Load

310. The entry point in DLL defined structure (DWORD) ____ values.

- 8
- 4
- 2

0304-1659294
www.vulmshelp.com

- 16

311. ReadFile() and writeFile() functions perform much _____ than memory mapped file processing

- Slower
- Faster
- Convenient
- Nimble

312. Which of the following controls the paging file?

- The pager
- Direct memory access
- Memory mapped I/o
- Virtual memory management system

313. While using memory mapped I/O there is/are _____ to manage buffers for repetitive operation on the file operations.

- Needed
- Not needed
- Useful
- Mandatory

314. In order to make a program more efficient, _____ heap(s) may be required.

- partial
- only one
- several
- Minimum number of

315. There are _____ parameters taken by the HeapCreate() API.

- 3
- 2
- 1
- 4

316. The parameter "flOptions" in the HeapCreate() API is a combination of _____ flags.

- 2
- 4
- 3
- 1

317. A process can have _____ heap(s).

- Only two
- At the most one

0304-1659294 www.vulmshelp.com

- only one
- Many

318. _____ API is used to create a new heap.

- createHeap()
- HeapCreate()
- BuildHeap()
- NewHeap()

319. If threads have separate memory space, then it will reduce_____

- Memory contention
- Access speed
- Direct memory access
- Memory density

320. _____ is an appropriate API to dispose-off a heap handle.

- shudderHandle()
- DestroyHandle()
- DeleteHeap()
- HeapDestroy()

AL-JUNAID TECH INSTITUTE
0304-1659294
www.vulmshelp.com

بسم الله الرحمن الرحيم

Virtual University	**CS609** **SYSTEM PROGRAMMING** **Update File (Topics)** **Final Term Mcqs (2023)**	**Current Paper 2023**
100% Correct	**Solved by Faisal (J.c FIA)**	**VU**

Question No: 1 (Marks: 01)
If the file is not opened in concurrent mode. then ReadFile() API starts reading from the ___
Answer: Current Position

Question No: 2 (Marks: 01)

The exception handler is actually a code portion associated with ____ block
Answer: **_Except**

Question No: 3 (Marks: 01)

If invalid file handle is passed-as parameter to the CloseFile() APL then it will return ___
Answer: **False value**

Question No: 4 (Marks: 01)
Which function is used to obtain the actual handle of a process?
Answer: OpenProcess()

Question No: 5 (Marks: 01)

It is _____ to refer to a mapping file multiple times while using pointers as data structure. Answer: Useful

Question No: 6 (Marks: 01)

The filter function _____ the type of an exception.
Answer: Identifies

Faisal (Jc FIA)

Question No: 7 (Marks: 01)

Which option is incorrect when the TraverseDiractory0 API is required to be use?
Answer: Recursive and non-recursive traversal option is irrelevant

Question No: 8 (Marks: 01)

The_____ function terminates the process if the program indicates the error is fatal.
Answer: TerminateProcess()

Question No: 9 (Marks: 01)

If binary functions inside DLL are accessed at load time, then called as
Answer: Implicit Linking

Question No: 10 (Marks: 01)

In a GetProcessTime() APL, the elapsed time of a process can be computed by subtracting the
Answer: CreationTime from ExitTime parameter

Question No: 11 (Marks: 01)

Which statement is incorrect when processes are made synchronized"?
Answer: It can decrease the overhead of a system.

Question No: 12 (Marks: 01)

Which parameter is required to be pass to the ExitThread() API?
Answer: Thread ExitCode
Question No: 13 (Marks: 01)

How many number of time's parameters are provided to a GetProcessTime() API to compute the elapsed time of a process?
Answer: 4

Faisal (Jc FIA)

Question No: 14 (Marks: 01) -

_____ API creates thread handle from a thread ID

Answer: OpenThread()

Question No: 15 (Marks: 01) -

When all the threads of a process are made exit or terminate, then the process _____

Answer: **Will Terminate**

Question No: 16 (Marks: 01) -

If you want to create 4 threads, Thread 0. Thread 1, Thread 2 and Thread 3 all must be created at ___ state.

Answer: Ready

Question No: 17 (Marks: 01)

A fiber can obtain it's identify by calling ____ API.

Answer: **GetCurrentFiber()**

Question No: 18 (Marks: 01)

Which of the following is not thread model?

Answer: Peer to Peer

Question No: 19 (Marks: 01)

In Thread Local Storage (TLS) arrangement. row represents _____ while column represent _____.

Answer: Thread Number, TLS Index.

Question No: 20 (Marks: 01)

The use of global variable flag in the solution of critical section problem suffers or fails due to the _____ of threads.

Faisal (Jc FIA)

Answer: Context switch

Question No: 21 (Marks: 01)

A thread can enter _____ time(s) into its critical section at different instances in thread synchronization.

Answer: Several

Question No: 22 (Marks: 01)

The producer also computes a simple _____of the message in the Producer-Consumer problem.

Answer: checksum

Question No: 23 (Marks: 01)

The use of _____ ensures that the main memory is updated and cache of all the processor is coherent.

Answer: Memory barrier

Question No: 24 (Marks: 01)

For the efficient use of critical section and mutexes, the thread priority should be decided by the_____.

Answer: Program

Question No: 25 (Marks: 01)

For mutual exclusion to work_____ variable(s) must be protected by a single synchronization object across all threads.

Answer: Shared

Question No: 26 (Marks: 01)

Producer consumer problem is a classical problem in_____.

Answer: mutual exclusion

Faisal (Jc FIA)

Question No: 27 (Marks: 01)

Windows provides an object called a _____ Using this object we can enforce mutual exclusion.
Answer: mutex

Question No: 28 (Marks: 01)

Mutexes have some advantages beyond CRITICAL SECTION. Mutexes can be named and have_____.
Answer: Handles
Question No: 29 (Marks: 01)

Which of the following synchronization construct is used for semaphore?

Answer: Wait

Question No: 30 (Marks: 01)

To minimize the concurrency related issues _____.
Answer: Size of the critical region should be minimized

Question No: 31 (Marks: 01) -

Which of the following synchronization objects is not a named object?
Answer: Critical Section.

Question No: 32 (Marks: 01)

According to the MSDN the optimal spin count for heap related functions is
Answer: 4000

Question No: 33 (Marks: 01)

Optimal spin-count ensures_____.
Answer: **Less system calls**

Faisal (Jc FIA)

Question No: 34 (Marks: 01)

A(n) _____ is a collection of worker threads that execute asynchronous callbacks on behalf of the application.
Answer: Thread pool.

Question No: 35 (Marks: 01)

When CreateThreadpoolWork() is called then is(are) executed using the process thread pool.
Answer: Callback functions.

Question No: 36 (Marks: 01)

_____ indicates the processors configured on the system.
Answer: Process Affinity Mask

Question No: 37 (Marks: 01) -

By default, the _____ makes scheduling decisions and allocates processors to each thread.
Answer: Kernel

Question No: 38 (Marks: 01) -

_____ can be used to assign a thread to a specific processor.
Answer: Processor Affinity

Question No: 39 (Marks: 01) -

Thread affinity mask is a subset of _____.
Answer: Processor affinity.

Faisal (Jc FIA)

Question No: 40 (Marks: 01)

Faisal (Jc FIA)

inprotected.com

Virtual University	**CS609** **SYSTEM PROGRAMMING** **Update Handouts (Topics)** **Final Term Mcqs (2023)**	**Sample Paper 2022**
100% Correct	**Solved by Faisal (J.c FIA)**	**VU**

Question No: 1 (Marks: 01) - Please choose the correct option

A _____ API is used to create a single threaded child process.

A. CreateFile()
B. CreateProcess()
C. CreateThread()
D. CreateProcessThread()

Question No: 2 (Marks: 01) - Please choose the correct option

Total number of parameters required for **CreateProcess()** API is_____.

A. 9
B. 10
C. 11
D. 12

Question No: 3 (Marks: 01) - Please choose the correct option

From which structure, we obtain the newly created child process's handle and ID?

A. LPSTARTUPINFO structure
B. LPPROCESS_INFORMATION structure
C. lpEnvironment Variable
D. LPSECURITY_ATTRIBUTES

Question No: 4 (Marks: 01) - Please choose the correct option

_____API is used to map a view of a file-mapping object into the

Faisal (Jc FIA)

address space of a calling process.

A. CreateViewOfFile()
B. MapViewOfFile()
C. CreateFileMapping()
D. OpenFileMapping()

Question No: 5 (Marks: 01) - Please choose the correct option

---

Which one is not a disadvantage of the static linking during compilations?

A. A big executable file
B. Takes more space on disk
C. Take less physical memory
D. Consume more bandwidth

Question No: 6 (Marks: 01) - Please choose the correct option

---

In Implicit linking, the library functions are linked to the source code at _____.

A. Written time
B. Compile time
C. Run time
D. Load time

Question No: 7 (Marks: 01) - Please choose the correct option

---

In which type of linking, the DLLs are not used?

A. Static linking
B. Explicit Linking only
C. Implicit linking only
D. Both Implicit and Explicit linking

Question No: 8 (Marks: 01) - Please choose the correct option

---

The **CreateProcess()** API returns _____ handle(s) after successful execution.

A. Zero
B. One
C. Two

Faisal (Jc FIA)

D. Three

Question No: 9 (Marks: 01) - Please choose the correct option

The return type of **ExitProcess()** API is ____.

Answer: does not have a return type

Question No: 10 (Marks: 01) - Please choose the correct option

). The number of parameters required for **TerminateProcess()** API is _____.

A. 1
B. 2
C. 3
D. 4

Question No: 11 (Marks: 01) - Please choose the correct option

. After loading a DLL explicitly, we can obtain the address of any entry point in DLL using the function_____.

A. GetModuleHandle
B. GetProcAddress()
C. LoadLibrary()
D. LoadLibraryEx()

Question No: 12 (Marks: 01) - Please choose the correct option

2. The number of parameters required for **CreateJobObject()** API is _____.

A. 0
B. 1
C. 2
D. 3

Question No: 13 (Marks: 01) - Please choose the correct option

Faisal (Jc FIA)

3. The return type of **CreateThread()** API is_____.

A. DWORD
B. LPDWOR
C. HANDLE
D. LPTSTR

Question No: 14 (Marks: 01) - Please choose the correct option

4. The default stack size for a thread is _____.

A. 1 MB
B. 1 KB
C. 1 Byte
D. 1 bit

Question No: 15 (Marks: 01) - Please choose the correct option

5. The number of parameters required for **GetExitCodeThread()** API is _____.

A. 0
B. 1
C. 2
D. 3

Question No: 16 (Marks: 01) - Please choose the correct option

6. TLS stands for _____.

A. Thread Local Storage
B. Thread Last State
C. Thread Local Size
D. Timestamp Local Storage

Question No: 17 (Marks: 01) - Please choose the correct option

7. A unit of execution that is scheduled by the application rather than by the OS Kernel, is called as _____.

Faisal (Jc FIA)

A. Fiber
B. Thread
C. Process
D. Mutex

Question No: 18 (Marks: 01) - Please choose the correct option

3. A fiber can obtain its starting address by using the API _____.
A. CreateFiber()
B. GetFiberData()
C. GetCurrrentFiber()
D. ConvertThreadToFiber()

Question No: 19 (Marks: 01) - Please choose the correct option

9. The return type of **CreateFiber()** API is _____.
A. VOID
B. LPVOID
C. HANDLE
D. BOOL

Question No: 20 (Marks: 01) - Please choose the correct option

0. Which one of the following is not a Process or thread synchronization object?
A. Fiber
B. Critical Section
C. Mutex
D. Semaphore

Question No: 21 (Marks: 01) - Please choose the correct option

1. Interlocked functions apply to _____integers.
A. 32 bits signed
B. 32 bits unsigned

Faisal (Jc FIA)

C. 64 bits signed
D. 64 bits unsigned

Question No: 22 (Marks: 01) - Please choose the correct option

2. Which one of the following is not a requirement for correct thread code?
A. Global Storage
B. Volatile Storage
C. Memory Barrier
D. Critical Code Region

Question No: 23 (Marks: 01) - Please choose the correct option

3. The return type of **CreateSemaphore()** API is _____.
A. DWORD
B. HANDLE
C. BOOL
D. VOID

Question No: 24 (Marks: 01) - Please choose the correct option

4. The code region where only one thread can execute at a time is known as _____.
A. program
B. critical section
C. non – critical section
D. Exception Handler

Question No: 25 (Marks: 01) - Please choose the correct option

5. Which one of the following APIs block a thread if another thread is in the section?
A. InitializeCriticalSection
B. EnterCriticalSection

Faisal (Jc FIA)

C. DeleteCriticalSection
D. LeaveCriticalSection

Question No: 26 (Marks: 01) - Please choose the correct option

6. Which one of the following numbers is good spin count for heap management according to MSDN?
A. 1000
B. 2000
C. 3000
D. 4000

Question No: 27 (Marks: 01) - Please choose the correct option

7. Which one of the following concepts limit the number of active, contending threads?
A. Mutual Exclusion
B. Thread Contention
C. Semaphore Throttle
D. Parallelism

Question No: 28 (Marks: 01) - Please choose the correct option

8. Which one of the following APIs returns a pointer to a TP_WORK structure?
A. TrySubmitThreadpoolCallback
B. WaitForThreadpoolWorkCallbacks
C. CreateThreadpoolWork
D. SubmitThreadpoolWork

Question No: 29 (Marks: 01) - Please choose the correct option

9. Thread synchronization is a process of handling situation when _____

Faisal (Jc FIA)

or more threads need access to a shared resource.

A. One
B. Two
C. Three
D. Four

Question No: 30 (Marks: 01) - Please choose the correct option

). Which one of the following options best describe the mutex?

A. is a binary mutex
B. does not have handles
C. must be accessed from only one process
D. can be accessed from multiple processes

Question No: 31 (Marks: 01) - Please choose the correct option

1. What will happen if a non-recursive mutex is locked more than once?

A. Starvation
B. Deadlock
C. Critical Section
D. Semaphore

Question No: 32 (Marks: 01) - Please choose the correct option

2. What are the two kinds of semaphores?

A. mutex & counting
B. binary & counting
C. counting & decimal
D. decimal & binary

Question No: 33 (Marks: 01) - Please choose the correct option

A. _____ function is supported only on the InterlockedExchange

Faisal (Jc FIA)

3. processor.

B. InterlockedExchangeAdd

C. InterlockedExchange

D. InterlockedExchangeAcquire

E. lockedCompareExchange

Question No: 34 (Marks: 01) - Please choose the correct option

---

4. Instead of starting a new thread for every task to execute concurrently, the task can be passed to a _____.

A. process

B. thread pool

C. thread queue

D. CPU

Question No: 35 (Marks: 01) - Please choose the correct option

---

5. Each process has its own process affinity mask, which is a _____ vector.

A. bit

B. byte

C. Nibble

D. Word

Question No: 36 (Marks: 01) - Please choose the correct option

---

6. Which one of the followings is not a feature of named pipes?

A. message-oriented

B. unidirectional

C. independent instances of pipes

D. Networked clients can be access pipes by name

Question No: 37 (Marks: 01) - Please choose the correct option

---

7. Which one of the following is not a characteristic of Anonymous pipe?

Faisal (Jc FIA)

A. Character based
B. Half Duplex
C. Full Duplex
D. Communicate with Linux & UNIX both

Question No: 38 (Marks: 01) - Please choose the correct option

8. Which one of the followings has two handles: Read handle and Write handle?
A. Named Pipes
B. Anonymous Pipes
C. Unnamed Pipes
D. Interprocess Pipes

Question No: 39 (Marks: 01) - Please choose the correct option

9. The return type of **CreatePipe()** API is _____.
E. DWORD
F. HANDLE
G. BOOL
H. VOID

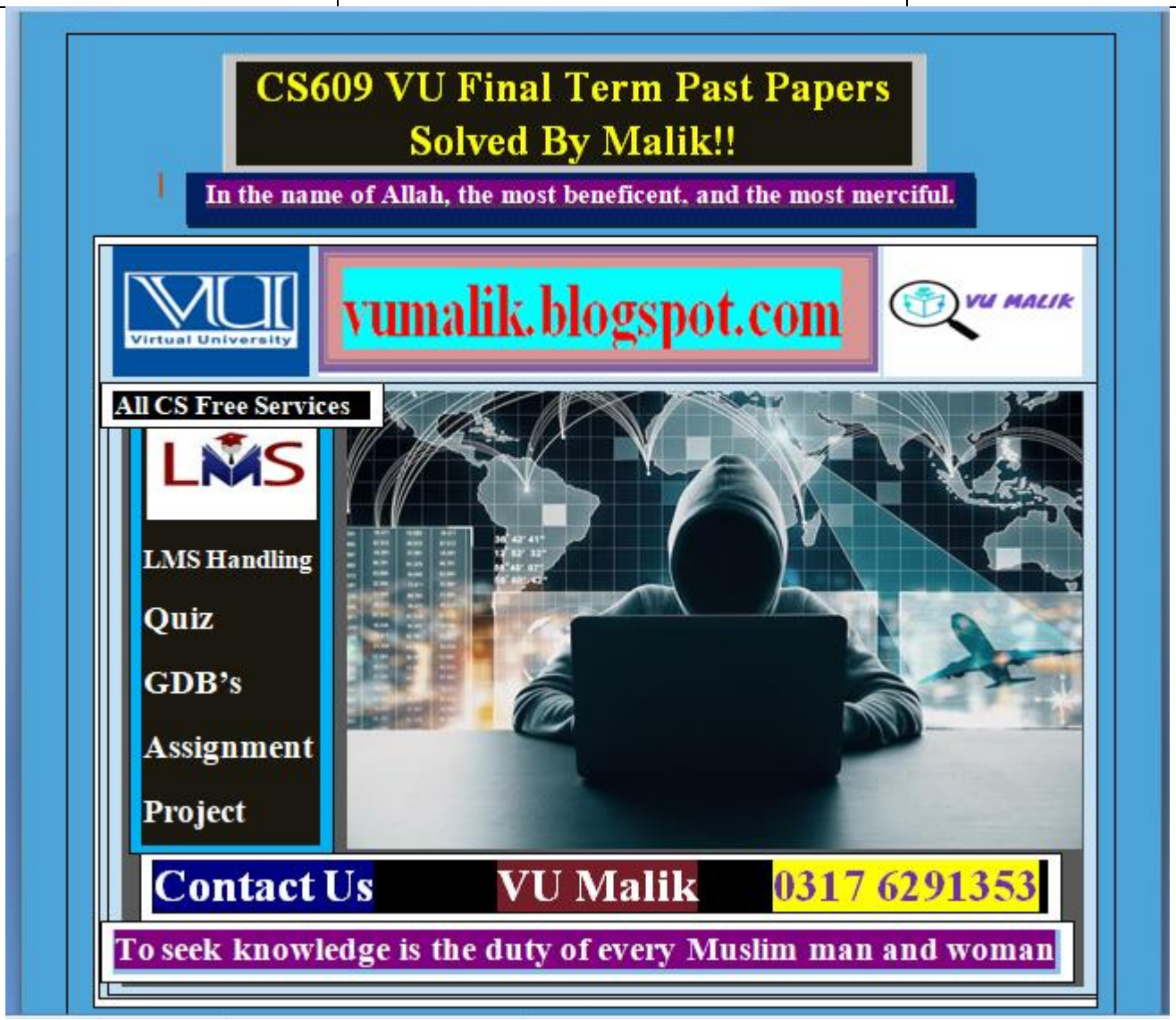
Question No: 40 (Marks: 01) - Please choose the correct option

0. Total number of parameters required for **CreateNamedPipe()** API is _____.
A. 8
B. 9
C. 10
D. 11

Faisal (Jc FIA)

soprotection.com

Virtual University	CS609-System Programming (Solved Macq's) For Final Term TOPIC 93 to 190	VU MALIK
vumalik.blogspot.com	FOR MORE VUMALIK.BLOGSPOT.COM	VU MALIK 03176291353



VU MALIK

1. The GetCommandLine() function returns the command line as a single_____
   - Float variable
   - Integer Array
   - Integer Variable
   - Character string Topic 101
2. Simple job management shell will allow _____commands to run.
   - Two
   - Five
   - Three Topic 104 (jobbg,jobs,kill)
   - Four
3. In job management shell the shell uses_____specific file keeping track of process ID and other related information
   - Shell
   - Process
   - System
   - User Topic 104
4. Process identify can be obtained from the_____ structure.
   - CreateProcess()
   - CREATE_PROCESS()
   - PROCESS_INFORMATION Topic 93
   - GetProcessInfo()
5. If the system is_____time is multiplexed among multiple processes in an interleaved manner.
   - Uniprocessor system Topic 99
   - Linear processor system
   - Multicore processor system
   - Multiprocessor system
6. If a system is_____then windows scheduler can run process threads on separate processors.
   - Uni processor
   - Linear processor
   - Single core processor
   - Multiprocessor Topic 99
7. FILETIME parameter of GetThreadTime() function is a_____bit value.
   - 32
   - 8
   - 16

- 64 Topic100

8. Closing the child process handle_____the process.
   - Does not create
   - Create
   - Destroy
   - Does not destroy Topic 93
9. Closing the child process only closes the access of the _____.
   - All parent process
   - Parent process Topic 93
   - All children processes of parent process
   - Child process
10. In windows, there are_____ways to get command line parameter for a process.
   - Two Topic 96
   - Four
   - Five
   - Three
11. Ip ApplicationName handle's value_____be NULL.
   - Should
   - Should not According to me
   - May
   - May not
12. Control of a running fiber can be given to another fiber by using----------function.
   - GetCurrentFiber()
   - SwitchToFiber() Topic 129
   - CreateFiber()
   - ConvertThreadToFiberEx()
13. A thread can enable fiber operation by calling_____function.
   - GetCurrentFiber()
   - SwitchToFiber()
   - CreateFiber()
   - ConvertThreadToFiberEx() Topic 129
14. Inherited handles are_____ copies that a parent and child might be accessing.
   - Connected
   - Similar
   - Distinct Topic 118
   - Related
15. A fiber can obtain its identity by calling_____function.
   - GetFiberData()
   - GetCurrentFiber() Topic 129
   - GetFiberId()

- GetFiberIdentify()

16. The_____function is used to obtain the process handle using the process ID.
    - openProcesses()
    - GetCurrentProcess()
    - OpenProcess() Topic 93
    - OpenprocessID()
17. Parent process usually creates a_____handle if parent and child process require _____ access rights.
    - Duplication, same
    - Duplication, different Topic 94
    - New, same
    - New, different
18. There are------ APIs to manage fiber.
    - 10
    - 9
    - 7 Topic 129
    - 8
19. One process can finish another process using the function _____
    - FinishProcess()
    - CreateProcess()
    - ExistProcess()
    - TerminateProcess() Topic 95
20. The simplest form of synchronization can be achieved through___construct.
    - Run
    - Halt
    - Lock
    - Wait Topic 96
21. The single object for which the process waits is activated by_____.
    - IpHandle
    - bWaitAll
    - hHandle Topic 96
    - nCount
22. The array of handles for which the process wait is activated by _____
    - IpHandle Topic 96
    - hHandle
    - nCount
    - bWaitAll
23. _____is the number of objects in an array. Should not exceed MAXIMUM WAIT OBJECTS
    - IpHandle

- hHandle
- nCount Topic 96
- bWaitAll

24. _____is the timeout period for wait. 0 for no wait and INFINITE for indefinitewait.
    - IpHandle
    - hHandle
    - nCount
    - dwMilliseconds Topic 96
25. _____handle describes if it's necessary to wait for all the objects to get free
    - IpHandle
    - hHandle
    - nCount
    - bWaitAll Topic 96
26. How many The possible return values are___
    - 2
    - 3
    - 4
    - 5 Topic 95
27. The timeout period for wait() function is measured in_____.
    - Millisecond Topic 96
    - Kilo hertz
    - Mega hertz
    - Microsecond
28. Its not a good idea to use_____within the program as it will not give it a chance to release resources.
    - createProcess()
    - TerminateProcess()
    - ExitProcess() Topic 95
    - FinishProcess()
29. A new fiber can be created by using_____
    - ConvertThreadToFiberEx()
    - ConvertThreadToFiber()
    - CreateFiber() Topic 129
    - GetFiber()
30. In Environment Block, IpName is a_____name.
    - Stack
    - Array
    - Process
    - Variable Topic 97

VU MALIK

31. In Environment Block, GetEnvironmentVariable() function returns_____in case of failure.
    - 0 Topic 97
    - NULL
    - -1
    - UNDEFINED
32. In Environment Block, PATH is an example of an environment _____
    - Stack
    - Array
    - Process
    - String Topic 97
33. An Environment Block is associated with_____ process(es) in the system.
    - Some
    - No
    - Each and every Topic 97
    - Exactly one
34. In Environment Block (EB), each string is_____
    - Five character long
    - Null terminated Topic 97
    - Undefined
    - Empty
35. The wait() function is limited to_____handles.
    - 32
    - 8
    - 64 Topic 98
    - 16
36. A_____value will cause a thread to move from the running state to the ready state.
    - Negative
    - INFINITE
    - 1
    - 0 Topic 127
37. Fiber which is shared to different threads_____
    - Should not access private data
    - Should not access global data
    - Should access thread specific data
    - Should not access thread specific data Topi 128
38. Fibers are scheduled by the _____
    - Hardware
    - Operating system
    - Application Topic 128

- BIOS

39. IPC stands for
    - Inter process communication My Point of View
    - Inter privacy communication
    - Information process communication
    - Information and privacy communication
40. The sleep function allows a thread to move the_____to the_____state.
    - Running, Suspended
    - Running, terminated
    - Running, ready
    - Running,wait Topi 127
41. The function SwitchToThread() provides a way for a thread to yield its processor to another_____thread if there is one that is_____ to run.
    - Ready, Running
    - Running, Running
    - Ready, Ready Topic 127
    - Running, Ready
42. The time period in sleep function is specified in _____.
    - Seconds
    - Milliseconds My point of View
    - Nanoseconds
    - Microseconds
43. Default stack size for a thread is _____.
    - 1MB Topic 126
    - 1 Byte
    - 1 Bit
    - 1KB
44. A scheduler will move a_____threads to the _____state if the threads time slice expires without the thread waiting.
    - Waiting, Waiting
    - Running, Ready Topic 125
    - Running, Waiting
    - Waiting, Ready
45. When a thread is suspended it goes into _____
    - Terminated state
    - Waiting state
    - Suspended state
    - Ready state Topic 125
46. A thread that is ready but do not have required resource will go into _____
    - Running state

- Suspended state
- Waiting state My point of View
- Execution state

47. The scheduler will run the_____priority_____thread when a processor becomes available.
    - Highest, ready Topic 125
    - Lowest, ready
    - Highest, running
    - Lowest, running
48. If there are no THREAD_PRIORTY_TIME_CRITICAL thread, the process will run _____first.
    - THREAD_PRIORTY_LOWEST
    - THREAD_PRIORTY_NORMAL My point of View
    - THREAD_ORIORITY_BELOW_NORMAL
    - THREAD_PRIORTY_IDLE
49. Which of the following priority is the highest class in the thread?
    - THREAD_PRIORTY_ABOVE HIGHEST
    - THREAD_PRIORTY_TIME_CRITICAL My point of View
    - REALTIME_ORIORITY_CLASS
    - THREAD_PRIORTY_HIGHEST
50. The range of relative thread priorities is between_____.
    - 0 to 1
    - 0 to 4
    - -2 to +2 (+-2) Topic 123
    - -1 to +1
51. Which of the following thread priorities will run first?
    - Thread with THREAD_PRIORTY_ABOVE_NORMA My point of View
    - Thread with THREAD_PRIORTY_BELOW_NORMAL
    - Thread with THREAD_PRIORTY_ _IDEAL
    - Thread with THREAD_PRIORTY_NORMAL
52. In windows, most common processes have_____ priority class
    - HIGH_PRIORTY_CLASS
    - IDLE_PRIORTY_CLASS
    - NORMAL_PRIORTY_CLASS My Point View
    - REALTIME_PRIORTY_CLASS
53. The value returned using TIsSetValue() function is in the form of _____
    - BOOL My Point of View
    - INT
    - LPVOID
    - DWORD

54. In Thread Local Storage (TLS) arrangement, row represents_____while column represent _____
    - Thread TLS Index Topic 123
    - Process Number, TLS Index not sure
    - TLS index, Thread
    - Thread, process Number
55. Which of the following function frees the specified index numbers?
    - TIsClear()
    - TIsFree() Topic 123
    - TIsReSet()
    - TIsDelete()
56. API is used to allocate the index and it returns the TLS index in theform of the double word
    - TIsClear()
    - TIsAlloc() Topic 123
    - TIsReSet()
    - TIsDelete()
57. _____Provided valid indexes are used, The programmer can access TLS spaceusing these simple GET/SET APIs
    - TlsGetValue ()
    - Both A and C Topic 123
    - TlsSetValue ()
    - TIsDelete()
58. In the form of failure, TIsAlloc() function returns_____.
    - DWORD
    - 0
    - -1 Topic 123
    - BOOL
59. Which of the following is a correct statement?
    - Every worker work with multiple processors on multiple processor
    - Every worker work as a separate thread on a separate processor Topic 122
    - Every worker work as a separate thread on a single processor
    - Every worker work with multiple processors on single processor
60. Which of the following is an optimal situation?
    - After certain limit processor speed can also be enhanced
    - After certain limit processor speed cannot be enhanced Topic 122
    - Multiprocessing is not responsible for multiple flows of execution
    - Output of parallel program should not be same when it is serialized
61. Which of following statement is incorrect?
    - Program performance can be scaled without any certain limit.

- Program performance can be enhanced by using multithreading.
- Program performance can be enhanced with Parallelism.
- Program performance scales automatically, up to some limit.

62. Four threads are created i.e. thread 0, thread 1, thread 2, and thread 3 and are running to sort a large file, select the most appropriate statement?
    - Wait for thread 1 to complete and merge it with thread 2.
    - Wait for thread 0 to complete and merge it with thread 2.
    - Wait for thread 1 to complete and merge it with thread 0. My point of view
    - Wait for thread 2 to complete and merge it with thread 0.
63. Why a file is always mapped before accessing it?
    - To access it just like accessing some data from main memory My point of view
    - To make it secure
    - To reuse it any time
    - To access it in the form of LinkedList
64. Once all the threads are created, then can be run using_____function.
    - ResumeThread()
    - ReadyThread()
    - RunThread() Topic 121
    - StartThread()
65. The basic difference between boss-worker thread model and client-server model is _____
    - In boss worker all the thread are run at the boss's end, but in client-server model, each client run a different thread
    - In the client-server all the thread are run at the server end, but in the boss worker model, each worker runs different thread Conceptual
66. Which of the following is not a thread model?
    - Peer-to-peer Topic 119
    - Pipeline
    - Boss-worker
    - Client-server
67. In client server model, rather than_____work is done _____
    - Concurrently, sequentially
    - Problematic, Efficiency
    - Sequentially, Concurrently Topic 119
    - Linear, Straight
68. In which of the following models, work moves from one thread to the next thread?
    - Peer-to-peer Model
    - Pipeline Model Topic 119
    - Boss-worker Model
    - Client-server Model

VU MALIK

69. The_____flag is set to be_____in the CreateProcess() function, which will determine whether child process will inherit copies of parent open handles.
    - bInheritFlag, FALSE
    - bInheritHandle, TRUE My point of View
    - bInheritFlag, TRUE
    - bInheritHandles, FALSE
70. Inherited handles are_____copies that a parent and child might be accessing
    - Similar
    - Connected
    - Distinct Wikipedia
    - Related
71. The get.JobNumber() function looks into the file for a vacant place. If no place is available, It appends a new record at the_____
    - Next
    - Start
    - Middle
    - End My point of View
72. In getting a job number, the job Management Shell uses a number of job management functions to manage jobs information within the_____life.
    - Process
    - User Topic 103
    - Shell
    - System
73. Fiber management occurs at the_____level.
    - Hardware
    - User space Topic 128
    - Kernel
    - BIOS
74. In Asynchronous input/ Output_____.
    - Input from the keyboard will be taken after the song is ended.
    - Song cannot be played until the other operations are not completed.
    - Song can be played with other operations being executed same time.

      Topic 119 also Conceptual (correct Accordingly)

    - Two operations cannot be run at the same time.
75. The first two parameters in argc will be_____and_____respectively.
    - Process name, pattern which is to be searched Topic 118
    - Inputted file names, pattern to be searched
    - Pattern to be searched, process name
    - Inputted file names, process name

VU MALIK

76. Total number of files inputted, can be obtained with_____-
    - Argc-1
    - Argc-2 Topic 118 (Confusion kindly Correct Accordingly)
    - Argc-3
    - Argc
77. C Library Threading functions are_____than windows library functions but not _____
    - Simpler, Hard
    - Hard, Simpler
    - Diverse, Simpler
    - Simpler, Diverse Topic 118
78. LIBCMT is a _____
    - BIOS Library for thread
    - Java Library for threads
    - C Library for threads Topic 118
    - Windows Library for threads
79. _____function is used to extract_____from a string.
    - gettok(), Token
    - getTok, String
    - strtok(), String
    - strtok(), Token Topic 118
80. The return type of _beingthreadex() function _____
    - is HANDLE, but we need to type caste it for further processing.
    - Is not HANDLE, but we cannot type caste it for further processing.
    - is not HANDLE, we need to type caste it for further processing.

      Topic 118 Conceptual
    - is HANDLE, but we don't need to type caste for further processing.
81. _____library function are thread safe than_____library functions.
    - C, java
    - Java, C
    - C, Windows Topic 118
    - Windows, C
82. Windows treats threads as_____.
    - Objects Topic 116
    - Threads
    - Processes
    - Classes
83. When a thread exits, the thread_____is deallocated and the handle referring to the thread in invalidated.
    - Stack Topic 113
    - List

- Queue
- Array

84. ResumeThread() function will_____.
    - Decrease the value of suspend count Topic 112
    - Reset the value of suspend count
    - Increase the value of suspend count
    - Not affect the value of suspend count
85. By default, the value of Suspend count, when creating a thread, is_____.
    - 2
    - 1
    - 0 Topic 112
    - -1
86. Which of the following version of windows was not compatible with GetProcessIdOfThread() function?
    - Windows 2003
    - Windows 7
    - Windows NT
    - Windows XP Topic 115
87. Which of the following function can be used to map a process with a thread?
    - GetProcessIdOfThread() My Point of View
    - DWORD ResumeThread (HANDLE hThread)
    - DWORD SuspendThread (HANDLE hThread)
    - GetThreadOPendingFlag()
88. A thread will execute if and only if its Suspend Count value is _____
    - Non zero
    - Above 1
    - 0 Topic 115
    - 1
89. Threads can also be treated as parent and child although the_____is unaware of that.
    - Kernel
    - Thread
    - Program
    - Operating system Topic 112
90. Threads share resources within a _____
    - Code
    - Program
    - Thread
    - Process Topic 110
91. Threads uses the space assigned to a _____
    - Thread

- Process Topic 111
- Code
- Program

92. In a multi-threading system, multiple threads may exist within a single _____
    - Thread
    - Process Topic 109
    - Code
    - Program
93. _____is an independent unit of execution within a process.
    - Thread Topic 109
    - Process
    - Code
    - Program
94. A job object is used to_____process execution time and obtain user time statistics.
    - Limit Topic 108
    - Write
    - Read
    - Open
95. The getJobNumber() function looks into the file for a vacant place. If no place is available, it appends a new recover of the file.
    - Start
    - Next
    - Middle
    - End My Point of View
96. In job Management Shell, the shell uses_____specific file keeping track of process ID and other related information.
    - User Topic 104
    - Process
    - Shell
    - System
97. In finding a Process ID, the FindProcessId() obtains the process ID of the given job number. It simply looks on job number and_____the record at the specific location.
    - Sums
    - Executes
    - Reads Topic 106
    - Writes
98. Simple job management shell will allow _____commands to run.
    - Two
    - Three Topic 104 (Jobbg, jobs, kill)
    - Four

- Five

99. In listing background jobs, the job management function uses display jobs () function. This function_____a file
    - Run
    - Start
    - Process
    - Open Topic 105

100. In Job Objects, firstly a job object is created using CreateJobObject(). It _____ a name and security attributes.
    - Writes
    - Closes
    - Uses Topic 107
    - Throws

101. Control of a running fiber can be given to another fiber by using_____function.
    - GetCurrentFiber()
    - SwitchToFiber() Topic 129
    - CreateFiber()
    - ConvertThreadToFiberEx()

102. A thread can enable fiber operation by calling_____function.
    - GetCurrentFiber()
    - SwitchToFiber()
    - CreateFiber()
    - ConvertThreadToFiberEx() OR ConvertThreadToFiber() Topic 129

103. Fiber can obtain its identity by calling_____function.
    - GetFiberData()
    - GetCurrentFiber() Topic 129
    - GetFiberId()
    - GetFiberIdentity()

104. There are_____APIs to manage fibers.
    - 10
    - 9
    - 7 Topic 129
    - 8

105. A new fiber can be created by using_____
    - ConvertThreadToFiberEx()
    - ConvertThreadToFiber()
    - CreateFiber() Topic 129
    - GetFiber()

106. The wait() function is limited to_____handles
    - 32

- 16
- 8
- 64 Topic 98

107. Windows OS does not have any structure that keeps track record of the _____ processes.
- Child
- Grand-child
- Parent
- Parent-Child My point of View

108. In comparison of DLL, executable library files are linked at_____time.
- Link
- Run
- Compiler My Point of View and GOOGLE
- Load

109. Interlocked functions are_____to use.
- Simpler, Faster but hard
- Simpler, Faster and easy Topic 135
- Not simpler but faster and easy
- Simpler easy but slow

110. Thread should not change the_____environment.
- Process Topic 137
- Hard disk
- Integer
- Ram

111. Which of the following function can be used to map the process with thread?
- DWORD Resume Thread (HANDLE hThread)
- DWORD Suspend Thread (HANDLE hThread)
- GetProcessIdOfThread() My Point of View
- GetThreadIOPendingFlag()

112. Mutexes, Semaphores, Events, and Critical Section are the four synchronization objects provided by_____.
- Software
- RAM
- Windows Topic 138
- processor

113. Like every other resource, threads are also treated as_____
- Code
- Object Topic 112
- Program
- Thread

114. The process Execution times uses the API GetCommandLine() to get the command line as a single_____.

- Bool
- Float
- String Topic 102
- None

115. lpApplicationName_____be NULL.

- May
- Should
- Should not My point of View
- May not

116. Volatile stage is a_____level facility.

- Compiler or windows Topic 133
- Programmer
- BIOS
- Hardware

117. CRITICAL_SECTION Objects do not have_____.

- Loops
- Handler Topic 139
- Variables
- Process

118. CRITICAL_SECTION Objects do not have__and are not shared among the____.

- Handler
- Both A and C Topic 139
- process
- Process

119. _____variables should not be accessible globally.

- String
- Locally required Topic 139
- Long
- Integer

120. Thread IDs and handles can be obtained using functions quite similar to the ones used with_____

- Process Topic 114
- Program
- Thread
- Object

121. A thread can be signaled from which of the following?

- ResumeThread()

- CloseThread()
- CreateThread()
- ExitThread() OR TerminateThread() Topic 117

122. If there are 129 objects, how many times we will have to call the wait () function?
- 4
- 6
- 3 My Point of View
- 2

123. ResumeThread() function will ____
- Increase
- Decrease Conceptual
- None of given
- Both

124. If you want to create 4 thread i.e, thread 0, thread 1, thread 2, and thread 3, they all must be created at_____state.
- Suspended Topic 120
- Blocked
- Running
- Ready

125. It must be ensured that _____thread(s) is/ are not modifying_____data at same time.
- Many, Many
- 1, Many
- Many,1 Topic 131 Conceptual
- 1,1

126. The result of concurrent processing and normal processing _____
- Yield same output
- Is always different
- Is always same
- May differ in some cases My point of View

127. In memory architecture and barriers, before writing the value s back to memory, the processor usually keeps them in _____
- Hard disk
- Memory card
- Processor
- Cache Topic 134

128. _____are used to ensure that memory is accessed in the desired order.
- Memory barriers or memory fences Topic 134
- Memory paths or memory houses
- Multi processors

- Memory blocks or memory chains

129. Turning off_____may adversely affect the performance but sometimes may _____the program.
- Usability, faster
- Optimization, slow Topic 133
- Performance, slow
- Security, faster

130. Turning off Optimization may adversely affect the performance but sometimes may slow the program. ANSI C provides a qualifier_____for this purpose.
- Volatile Topic 131
- Non-volatile
- Both A and B
- None of Given

131. The volatile qualifier does not guarantee that the modifications will be visible to _____in a desired order.
- Processors Topic 134
- User
- Ram
- Hard disk

132. The_____functions provide memory barriers.
- Interlocked Topic 139
- User define
- Interchange
- Intermediate

133. Interlocked increment () takes a 32-bit signed variable as an arguments, which should be placed in memory at the_____boundary.
- 4-byte Topic 135
- 8-byte
- 16-byte
- 2-byte

134. If a variable with a volatile scope only needs to be incremented, decremented, or exchanged, interlocked functions are the_____choice.
- Last
- Best Topic 135
- 2nd last
- Worst

135. _____are most suited if variable with volatile scope only need to be incremented,decremented and exchanges
- Interlocked functions Topic 135
- Local

- Global
- None

136. The most basic of Interlocked Functions
- Interlockedincrement()
- InterlockedDecrement()
- Both A and B Topic 135
- None of Given

137. How many Basic Function of Interlocked()
- 1
- 2 Topic 135
- 3
- 4

138. If any needed variables are not initialized, then create_____threads until variables are initializes.
- Suspended Topic 137
- Double
- Slow
- Fast

139. In thread safe code,_____ conditions are avoided.
- Race Topic 137
- Loop
- Local
- Global

140. When more than one thread can run the same code without introducing synchronization issues, it is said to be _____
- Thread safe Topic 137
- Local
- Threaded
- Initialized properly

141. InterlockedIncremen64() and interlockedDecrement64() can be utilized on 64-bit systems if the addend is paced at the_____boundary.
- 8-byte Topic 135
- 4-byte
- 2-byte
- 16-byte

142. _____have a performance disadvantage since they create a memory barrier.
- Interchanged function
- Interlocked function Topic 135
- Intermediate function
- Interlocked memories

143. If a_____is used to store thread-specific data, other threads will also use it.
- Small variable
- Large variable
- Global variable topic 136
- Local variable

144. Make a variable_____if we know it includes thread-specific information.
- Static
- Local Topic 136
- Global
- Random

145. According to the criterion for good thread code, _____storage should not be used for the purpose of local storage.
- Hard disk
- Global Topic 136
- Cache
- Ram

146. If we know the information in a variable will be used by other threads, we can make it_____.
- Global Topic 136
- Local
- Main
- Auto

147. When using thread synchronization objects, there are always inherent risks associated with them such as _____
- Lower power
- Multi clocks
- Load
- Deadlocks Topic 137

148. A thread waits for other threads to_____using the waitForString0bject() or waitForMultipleObjects() methods.
- Wait
- Terminate Topic 138
- Start
- Update

149. The part of the code that can only be executed by one thread at a time is referred to as the _____
- Process section
- Loop section
- Top section
- Critical section Topic 139

150. Only_____can be in the CRITICAL_SECTION variable at a time
- Odd threads
- Many threads
- One thread Topic 139
- Even threads

151. Which function is used to initialized the CRITICAL_SECTION.
- InitializeCriticalSection() Topic 139
- DeleteCriticalSection()
- Both A and B
- None of Given

152. Which function is used to Delee the CRITICAL_SECTION.
- InitializeCriticalSection()
- DeleteCriticalSection() Topic 139
- Both A and B
- None of Given

153. Multiple threads may call EnterCriticalSection() but only_____thread is allowed to enter the critical section while the rest areblocked.
- One Topic 139
- Two
- Three
- Foure

154. A call to LeaveCriticalSection() must match a/ an____
- EnterSection()
- Variables
- EnterCriticalSection() Topic 139
- CriticalSectionPart()

155. CRITICAL_SECTION object do not have _____
- Process
- Variables
- Handles Topic 139
- Loops

156. EnterCriticalSection() can be called by_____but only one of them is allowed to enter the critical section, while the others are blocked.
- Two threads
- One thread Topic 139
- Half thread
- Many threads

157. Using the critical section construct is_____and intuitive.
- Easy Topic 140
- Difficult

- Short
- Long

158. It would be_____to use different objects in the same thread or across numerous threads that share the same data.

- Easy
- Incorrect Topic 141
- Correct
- Difficult

159. For mutual exclusion to work,_____variables must be protected by a single object across all threads.

- Integer
- Important
- Shared Topic 141
- specific

160. Within the critical section, all the variables must be guarded by _____

- Critical object
- All objects
- A single object Topic 141
- Double object

161. What is the limit for object to be waited for waitForMultiple Objects () function in window?

- 63
- 65
- 62
- 64 Topic 141

162. Producer consumer problem is a classical problem in_____

- Critical Section
- Conical collection
- Mutual Exclusion Topic 142
- None of given

163. The producer periodically_____a message.

- Creates Topic 142
- Delete
- Update
- None

164. The producer also computes a simple_____of the message

- Checksum Topic 142
- Contents
- Update
- None

165. _____is a short form of Mutual Exclusion
- Mechanism
- Checksum
- Mutex Topic 143
- All of the given

166. Which of the following Windows functions used to manage mutexes.
- CreateMutex()
- ReleaseMutex()
- OpenMutex()
- All of the given Topic 144

167. Home Many windows Function are used to manage mutex.
- 1
- 2
- 3 Topic 144
- 4

168. lpMutexName is the name of the mutex.
- True Topic 144
- False

169. OpenMutex() is used to _____and_____named mutex
- Existing,open
- Open,Existing Topic 144
- Open,Delete
- Exsiting,Delete

170. However, abrupt termination of a thread indicates a serious programming flaw. Mutex waits can_____
- Time In
- Time Out Topic 146
- Has
- Does Not

171. If NULL is returned it indicates dose not failure.
- True
- False Topic 144

172. Semaphore maintains a count
- True Topic 147
- False

173. A semaphore is in a signaled state when the count is greater than.
- 1
- -1
- 0 Topic 147
- None of given

174. Semaphore is_____when the count is 0.
- Signaled
- Un-signaled Topic 147
- Release
- None of the given

175. Which of the following are used in semaphore Count?
- CreateSemaphore()
- CreateSemaphoreEx()
- OpenSemaphore()
- All of the Given Topic 147

176. cReleaseCount gives the count after the release and must be greater than____.
- 1
- -1
- 0 Topic 147
- None of given

177. Events are classified as either
- manual-reset
- auto-reset
- Both A and B Topic 150

178. A_____event can signal several waiting threads simultaneously and can be reset.
- manual-reset Topic 150
- auto-reset

179. An_____event signals a single waiting thread, and the event is reset automatically.
- manual-reset
- auto-reset Topic 150

180. If bInitialState is TRUE, the event is set to a_____state
- Signaled Topic 150
- Un-Signaled

181. If the event is manual-reset, it remains signaled until a thread explicitly calls ResetEvent().
- True Topic 150
- False

182. How many ways to use Events?
- 1
- 2
- 3
- 4 Topic 151

VU MALIK

183. Windows operating system is a multithreaded_____that provides support for real time applications and multiprocessors.
- Hardware
- User space
- Kernel Topic 153
- BIOS

184. Program design and performance can be simplified and improved by
- Odd threads
- Many threads
- Threads Topic 153
- Even threads

185. _____Synchronization is a way to coordinate processes that use shared data
- Shell
- Process Topic 153
- System
- User

186. Which of the following Synchronization objects.
- Critical Section
- Semaphore
- Mutex
- All of the Given Topic 153

187. The_____functions provide a simple mechanism for synchronizing access to a variable that is shared by multiple threads.
- Interlocked Topic 155
- Local
- Global
- None

188. Memory allocation is then performed using using_____rather than using__
- HeapAlloc() and HeapFree(),malloc() and free() Topic 56
- malloc() and free() and HeapAlloc() and HeapFree(),

189. Each thread that performs memory management can create a Handle to its own heap using _____
- HeapCreate() Topic 156
- HeapAlloc()
- HeapFree()
- None of the given

190. Threads allocate memory and free memory using_____functions respectively
- malloc () and free () Topic 156
- free () and malloc ()

VU MALIK

191. How may aspects compare the performance on the Basis.
- 1
- 2
- 3 Topic 157
- 4

192. Which of the Following aspects that are used to compare the Performance.
- Real-Time
- User-Time
- System-time
- All of the given Topic 157

193. MX(Mutexes) version costs more than_____times than IN
- 2 times
- 2 to 30 times Topic 158
- 1 to 30 times
- 3 times

194. CS (Critical Section) version costs_____times more than IN
- 2 to 30 times
- 2 times Topic 158
- 1 to 30 times
- 3 times

195. For older version of windows CS was not scalable.
- True Topic 159
- False

196. Critical Section works in_____
- Hardware
- User space Topic 161
- Kernel
- BIOS

197. How many methods of Lightweight Reader Writer Locks.
- 1
- 2 Topic 163
- 3
- 4

198. Which of the following methods of Lightweight Reader Writer Locks
- Exclusive Mode
- Shared Mode
- Both A and B Topic 163
- None of given

199. How many APIs which are used to access SRWs in Exclusive mode.
- 1

- 2 Topic 164
- 3
- 4

200. When a thread is created almost_____space is reserved for that thread and with increasing threads memory area is piled up.
- 1 MB Topic 166
- 1 KB
- 1 Bit
- 1 Byte

201. If you_____the threads the you can lose the benefits of parallelism and synchronization
- Minimize Topic 166
- Maximize

202. Which of the following optimization technique?
- Use of semaphore throttles
- Asynchronous I/O
- Using I/O completion ports
- All of the Given Topic 166

203. Every process has a dedicated_____pool.
- Odd threads
- Many threads
- Threads Topic 172
- Even threads

204. The thread pool is used to_____the number of application threads and provide management of the worker threads
- Increased
- Reduce Topic 172
- Constant
- Equal

205. How many types of call back Function.
- 1
- 2 Topic 173
- 3
- 4

206. How many sorts of parallelism?
- 1
- 2 Topic 177
- 3

- 4

207. Which of the following parallelism define every loop iteration can execute concurrently.

- Loop parallelism Topic 177
- Fork-join parallelism

208. _____the control flow divides (like the shape of the fork) into multiple flows that join later

- Loop parallelism
- Fork-join parallelism Topic 177

209. Each process has its own process affinity mask and a system affinity mask.

- True Topic 180
- False

210. How many Bits vector of MASK?

- 1
- 2
- 3 Topic 180
- 4

211. Which of the follwing Bits vector of MASK?

- System Affinity Mask
- Process Affinity Mask
- Thread Affinity Mask
- All of the Given Topic 180

212. IPC stand for_____

- Internet process control
- Inter-process Communication Topic 180
- Inter-parameter control
- None of the Given

213. How many Types of Pipe?

- 1
- 2 Topic 181 (Anonymous Pipes, Named Pipes)
- 3
- 4

214. File-like object called Pipe can be used for IPC.

- True Topic 181
- False

215. Which of the following are correct statement of Anonymous Pipes?

- Simple anonymous pipes are character based and half duplex. Topic 181
- They allow network wide communication.

216. Which of the following are correct statement of Named Pipes?

- There can be multiple open handles for a pipe
- They allow network wide communication.
- Much more powerful than anonymous pipes
- All of the Given Topic 181

217. Anonymous pipes allow_____(half-duplex) communication

- one-way Topic 182
- two-way

218. The default set of cbPipe is_____

- 0 Topic 182
- 1
- 2
- 3

219. Named Pipe is_____

- Directional
- Bi-Directional Topic 184
- Tri-Directional
- None of the Given

220. The pipe name would be like this.

- \\.\pipe\[path]pipename Topic 187
- \\servername\pipe\pipename

221. Which of the following connection sequences for the Client.

- The client connects with a server
- Communicates with the server using CreateFile()
- Performs Read and Write operations and ultimately disconnects
- All of the given Topic 190

222. Which of the following connection sequences for the Server.

- Communicates with the client.
- As a result ReadFile() returns FALSE
- The server-side connection is disconnected
- All of the given Topic 190

223. Which of the following Returns information whether the pipe is in blocking or non-blocking mode, message oriented or byte oriented, number of pipe instances, and so on.

- SetNamedPipeHandleState()
- GetNamedPipeHandleState() Topic 188
- GetNamedPipeInfo()
- None of the given

224. Which of the following Allows the program to set the same state attributes. Mode and other values are passed as reference so that NULL can also be passed indicating no change is desired.

- SetNamedPipeHandleState() Topic 188
- GetNamedPipeHandleState()
- GetNamedPipeInfo()
- None of the given

225. Which of the following Determines whether the handle is for client or a server, buffer sizes, and so on.
- SetNamedPipeHandleState()
- GetNamedPipeHandleState()
- GetNamedPipeInfo() Topic 188
- None of the given

226. Call_____to disconnect from the handle
- DisconnectNamedPipe() Topic 189
- WaitNamedPipe()
- ConnectNamedPipe()
- All of the given

227. _____is used to synchronize connections to the server
- DisconnectNamedPipe()
- WaitNamedPipe() Topic 189
- ConnectNamedPipe()
- All of the given

228. _____are the security attributes as discussed previously
- lpSecurityAttributes Topic 186
- nOutBufferSize
- dwOpenMode
- nMaxInstance

229. _____and_____give the size in bytes of input and output buffer
- nOutBufferSize and nInBufferSize Topic 186
- nInBufferSize and nOutBufferSize

230. which of the following determines maximum number of pipe instances?
- lpSecurityAttributes
- nOutBufferSize
- dwOpenMode
- nMaxInstance Topic 186

231. which of the following indicates whether writing is message oriented or byte oriented?
- dwPipeMode Topic 186
- nOutBufferSize
- dwOpenMode
- nMaxInstance

232. _____is the default timeout period.

- nDefaultTimeOut Topic 186
- nOutBufferSize
- dwOpenMode
- nMaxInstance

233. The period (.) stands for local machine.
- True Topic 186
- False

234. In mutex which type of data structure that stores the resource should also be used to store the mutexes because mutexes correspond to the resources.
- Different Topic 154
- Same
- Equal
- None of the Given

235. Only the_____Scheduler decides which thread has the priority according to its scheduling policy.
- Hardware
- Application
- OS Topic 154
- BIOS

236. The shell uses a ..... file keeping track of process ID and other related information?
- **User-specific**

237. The array of handles for which the process waits.
- **lpHandle**

Which header file includes all the Unicode macros for setting environment of a program?

- <everthing.h>
- **<environment.h>**
- <tchar.h>
- <Windows.h>

In NTFS based system _____ is the maximum allowed size for a single file

- 2^8
- 2^{32}
- 2^{16}
- **2^{64}**

Which major theme in windows evolution focuses on running on enterprise servers with multiple processors?

- Performance
- Ease of use
- Integration
- **Scalability**

RegSetvalueEx() function is used to set the data of a ____ value.

- Final
- Fixed
- **Named**
- Static

Which of the following function takes variable list of parameters?

- PrintMsg()
- ConsolePrompt()
- WriteConsole()
- **PrintString()**

TextOut() API has _____ variants.

- TextOutA and TextOutB
- TextOutA and TextOutX
- TextOutW and TextOutX
- **TextOutA and TextOutW**

Which of the following function uses the va start, va_arg, and va end library functions?

- PrintMsg()
- **PrintString()**
- ConsolePrompt()
- WriteConsole()

What is the return type of Options() function

- VOID
- BOOL
- **DWORD**
- HANDLE

Catfile() function takes _____ parameters.

- 3
- 5
- 4
- **2**

Which statement is true about hard copy function?

- Both the files must not be on same system volume.
- Both the files must be in encrypted form
- **Security attributes will apply on new file name**
- Creates a hard link for copy file

The filter-expression in the _____ clause is evaluated immediately after the exception occurs.

- -try
- -catch
- **-except**
- -finally

Which of the following Windows API is used to invalidate kernel handles and to release system resources?

- ExitFile()
- **CloseFile()**
- TerminateFile()
- ReleaseFile()

The name of windows API used for opening and creating a new file is

- **CreateFile()**
- OpenFile()
- CreateopenFile()
- ReopenFile()

What is the maximum length for file names used as API function arguments?

- 128 characters
- **255 characters**
- 512 characters
- Unlimited length

A Unicode word consists of _____ bits

- 24
- 32
- **16**
- 8

To reduce the filesize, we use _____ windows API.

- Setfilesize()
- Changefilesize()
- **SetendoffileEx()**
- SetfileEx()

Which macros should be used before including <windows.h> to work with Unicode in Windows?

- **#define UNICODE and #define _UNICODE**
- #define ASCII and #define _ANSI
- #define UNICODE and #define _ASCII
- #define ASCII and #define _UNICODE

setfilePointerEx() AP has _____ parameters

- **Four**
- Two
- Five
- Three

SEH is not supported through..

- Compiler supported language extensions
- Run-time support
- **Windows Registry**
- Windows functions

Choose the correct option for a source code that uses only windows API instead of C library functions.

- Source code can only access some features of windows
- **Source code will not remain portable**
- Source code cannot be complied
- Source code is portable

#include is used for _____

- **Input output operation**
- Working in CLI
- Memory allocation
- Multitasking

Every lockfileEx() function that is successful must be followed by a call to _____

- DeletelockEx()
- RemovelockEx()
- UnlatchlockEx()
- **UnclockfileEx()**

Correct syntax of GetCurrentDirectory() function is

- lenCurDir = GetCurrentDirectory (pwdBuffer, DIRNAME_LEN)
- lenCurDir = GetCurrentDirectory (DIRNAME_LEN);
- **lenCurDir = GetCurrentDirectory [DIRNAME_LEN, pwdbuffer)**
- lenCurDir = GetCurrentDirectory (pwdbuffer)

How many parameter takes functions createDirectory()

- 4
- 3
- 1
- **2**

Which of the following is not a Standard C function?

- atoi

- strcmp
- Print
- **CopyFile**

If invalid file handle is passed as a parameter to the closeFile API, then it will return_____

- Empty string
- **False value**
- 1
- File handle

Roman Empire use _____ algorithm to encrypt secret conversation.

- CTR
- **Ceaser cipher**
- Brute force
- Cryto graph

According to the Unicode and 8-bit strategy, the program includes both ____ and ____ code.

- ASCII, Generic
- Generic, windows specific
- **Unicode, ASCII**
- Unicode Generic

A uniform extension is software components are possible in ___ software.

- **Vender specific**
- Open source
- Community source
- Both open and closed

Return type of printString() function is _____

- DWORD
- CHAR
- **BOOL**
- WORD

Each windows API has a _____ number of parameters.

- Fixed
- Two

- **Variable**
- Four

In lockFileEx() function, the OVERLAPPED data structure contains _____ data members.

- **3**
- 5
- 2
- 4

The number of arguments required for Findclose()API is ____

- 3
- **1**
- 2
- 0

Which of the following parameter of CreateFile() API is used to access to the file or device?

- Templaterie
- dwCreationDisposition
- **dwDesiredAccess**
- dwshareMode

One of the major causes of windows dominance in PC's market is its

- **User-friendly GUI**
- Best security features
- Feature of multitasking
- Best marketing strategy

The return type of readfile() function is_____.

- DROWD
- **BOOL**
- HANDLE
- INT

If process-A has a shared lock on a file, and process-B tries to read without a shared lock then the read attempt will _____

- Return exception
- **Succeed**
- Return a shared lock

- Fail

The options function has _____ arguments

- 5
- 4
- **7**
- 6

The formula of ceaser chopper is _____

- C = (E + W) mod26
- E = (P +n) mod27
- E = (D +n) mod27
- **C = (P + n) mod26**

PLONG is a _____

- **Pointer to a long variable**
- String
- Variable
- Pointer to a string

Which of the following desktop operating systems was widely used in the world market?

- **Windows**
- Fedora
- Salaris
- Linux

In the RECORD structure, data type of numNonemptyRecord is _____

- **DWORD**
- BOOL
- Double
- Int

NTFS stand for _____

- New trend file system
- **New technology file system**
- New trend for system
- New technology for system

Overlap structure is a structure which is defined in the _____ header file

- **Window.h**
- Stdary.h
- Everything.h
- Stdio.h

What is the return type of GetTempFileName() function?

- HANDLE
- BOOL
- VOID
- **UINT**

DWORD data type represents _____

- **32 bit unsigned integer**
- 32 bit signed integer
- 16 bit unsigned integer
- 16 bit signed integer

API stands for _____.

- **Application programming interface**
- Advance programming interface
- Application programming integrity
- Application programs interoperability.

The termination handler cannot execute the____ statement

- Break
- **Report**
- Continue
- Return

_tcscmp() is _____ function to compare the string

- An ASCII
- A Unicode
- Not a generic
- **A generic**

what is the return type of FindFirstFile() function?

- BOOL
- **HANDLE**
- DWORD
- VOID

Return type of setFilePointerEx() is _____

- String
- **BOOL**
- Word
- DWORD

How can you ensure that your programs run under both Win32 and Win64 in Windows?

- Carefully manage virtual address spaces
- **Write separate code for 32-bit and 64-bit versions**
- Test programs on 64-bit computers only
- Avoid using pointers altogether

Windows provides convenience functions that:

- Implement object-oriented programming principles
- **Combine common sequences of function calls into one function**
- Create back doors for easy access
- Manipulate kernel objects directly

GetFileAttribute() API need ____ argument(s) to return the attributes of a file or directory

- **1**
- 2
- 3
- 0

Every subsequent read/write operation is performed at the file pointer position.

- **Current**
- Start
- End
- Middle

While using CreateFileMapping(),setting lpMapName to _____ disables the map sharing.

- -1

- **NULL**
- 0
- 1

_____ stores the root address.

- RootHeap
- RecHeap
- **ProcHeap**
- NodHeap

In order to deallocate a block of heap,_____ is used.

- HeapTruncate()
- HeapDelete()
- **HeapFree()**
- HeapDestroy()

Windows mainly uses _____ API.

- 4
- **2**
- 3
- 1

For sequential files, file mapping may reduce the processing time by folds as compared to conventional

- Nine
- Six
- **Three**
- Ten

Pages are swapped in and out when a_____ occurs.

- **Page fault**
- Frame fault
- Page error
- Dirty frame

The _____ exception are enabled with the help of controlfp() function

- **Floating point**
- String

- Mutex
- Integer

Dynamic memory is allocated from the

- Cache
- **Paging file**
- Stack
- Static memory

Which of the following instruction is used to suspend the execution of a program for 5 milliseconds?

- Sleep(500)
- Sleep(5000)
- **Sleep(5)**
- Sleep(0.5)

The heapAlloc() API has _____ parameter(s).

- **3**
- 4
- 2
- 1

To start the mapping from the start of a file, set both dwFileOffsetHigh and dwFileOffsetlow as _____

- -1
- NULL
- **0**
- 1

Which of the following are the number of parameters taken by OpenFileMapping()?

- 4
- 2
- **3**
- 1

_____ API is used to create a new heap.

- createHeap()
- **HeapCreate()**

- BuildHeap()
- NewHeap()

If threads have separate memory space, then it will reduce_____

- **Memory contention**
- Access speed
- Direct memory access
- Memory density

Try and catch keywords _____ required for vectored exception handlers.

- **Are not**
- Are
- Must be
- Are occasionally

Console control handlers are similar to _____

- Interrupts
- Parameters
- Function
- **Exception handlers**

In DDLs the executable library files are linked at ____ time.

- Link
- Run
- Compile
- **Load**

Serialization is necessary when concurrent threads use _____ resource(s).

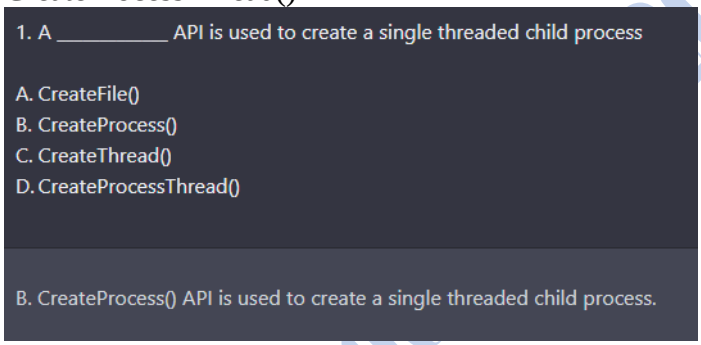
- The remaining
- Different
- Extra
- **A common**

Sample Paper solved by Nasir Mehmood
FINALTERM EXAMINATION
Fall 2022
CS609 – System Programming

Time: 90 min
Marks: 60

Question No: 1 (Marks: 01) - Please choose the correct option

1. A _____ API is used to create a single threaded child process.
   A. CreateFile()
   B. CreateProcess()
   C. CreateThread()
   D. CreateProcessThread()



Question No: 2 (Marks: 01) - Please choose the correct option

2. Total number of parameters required for **CreateProcess()** API is_____.
   A. 9
   B. 10
   C. 11
   D. 12

Question No: 3 (Marks: 01) - Please choose the correct option

3. From which structure, we obtain the newly created child process's handle and ID?
   A. LPSTARTUPINFO structure
   B. LPPROCESS_INFORMATION structure
   C. lpEnvironment Variable
   D. LPSECURITY_ATTRIBUTES

Question No: 4 (Marks: 01) - Please choose the correct option

4. _____API is used to map a view of a file-mapping object into the address

Nasir Mehmood 03040403630

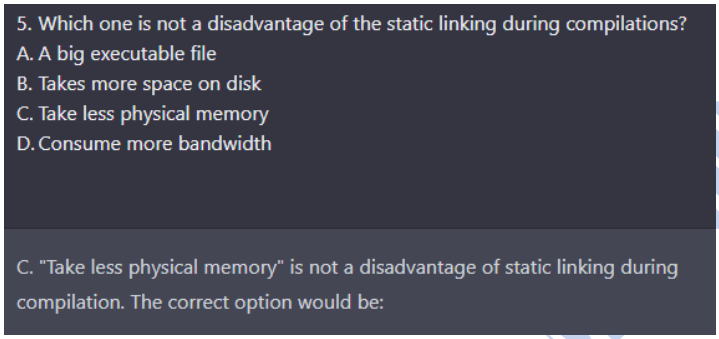
space of a calling process.

A. CreateViewOfFile()
B. **MapViewOfFile**()
C. CreateFileMapping()
D. OpenFileMapping()

Question No: 5 (Marks: 01) - Please choose the correct option

---

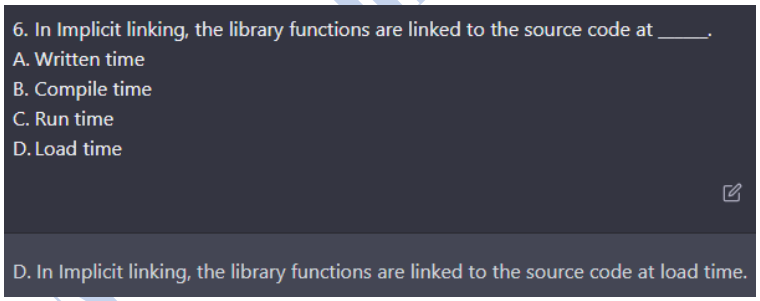
5. Which one is not a disadvantage of the static linking during compilations?
   A. A big executable file
   B. Takes more space on disk
   C. Take less physical memory
   D. Consume more bandwidth



Question No: 6 (Marks: 01) - Please choose the correct option

---

6. In Implicit linking, the library functions are linked to the source code at _____.
   A. Written time
   B. Compile time
   C. Run time
   D. Load time



Question No: 7 (Marks: 01) - Please choose the correct option

---

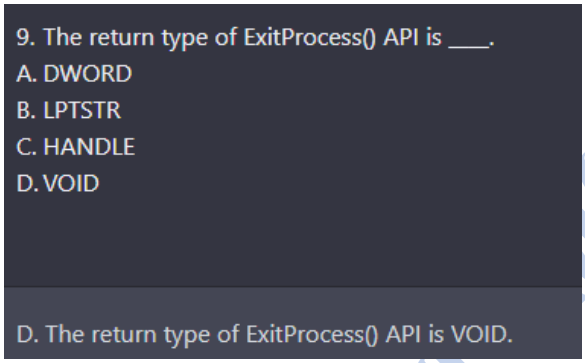
7. In which type of linking, the DLLs are not used?
   A. Static linking
   B. Explicit Linking only
   C. Implicit linking only
   D. Both Implicit and Explicit linking

Question No: 8 (Marks: 01) - Please choose the correct option

8. The **CreateProcess()** API returns _____ handle(s) after successful execution.
   A. Zero
   B. One
   C. Two
   D. Three

Question No: 9 (Marks: 01) - Please choose the correct option

9. The return type of **ExitProcess()** API is ____.
   A. DWORD
   B. LPTSTR
   C. HANDLE
   D. VOID



Question No: 10 (Marks: 01) - Please choose the correct option

10. The number of parameters required for **TerminateProcess()** API is _____.
    A. 1
    B. 2
    C. 3
    D. 4

Question No: 11 (Marks: 01) - Please choose the correct option

11. After loading a DLL explicitly, we can obtain the address of any entry point in DLL using the function_____.
    A. GetModuleHandle
    B. GetProcAddress()
    C. LoadLibrary()
    D. LoadLibraryEx()

Question No: 12 (Marks: 01) - Please choose the correct option

12. The number of parameters required for **CreateJobObjectA()** API is _____.
    A. 0

Nasir Mehmood BC200403630

B. 1
C. 2
D. 3

Question No: 13 (Marks: 01) - Please choose the correct option

13. The return type of **CreateThread()** API is **_____**.
A. DWORD
B. LPDWOR
C. HANDLE
D. LPTSTR

Question No: 14 (Marks: 01) - Please choose the correct option

14. The default stack size for a thread is _____.
A. 1 MB
B. 1 KB
C. 1 Byte
D. 1 bit

Question No: 15 (Marks: 01) - Please choose the correct option

15. The number of parameters required for **GetExitCodeThread()** API is _____.
A. 0
B. 1
C. 2
D. 3

Question No: 16 (Marks: 01) - Please choose the correct option

16. TLS stands for _____.
A. Thread Local Storage
B. Thread Last State
C. Thread Local Size
D. Timestamp Local Storage

Question No: 17 (Marks: 01) - Please choose the correct option

17. A unit of execution that is scheduled by the application rather than by the OS Kernel, is called as _____.
A. Fiber
B. Thread
C. Process
D. Mutex

Nasir Mehmood MC200403630

Question No: 18 (Marks: 01) - Please choose the correct option

18. A fiber can obtain its starting address by using the API _____.
    A. CreateFiber()
    B. GetFiberData()
    C. GetCurrrentFiber()
    D. ConvertThreadToFiber()

Question No: 19 (Marks: 01) - Please choose the correct option

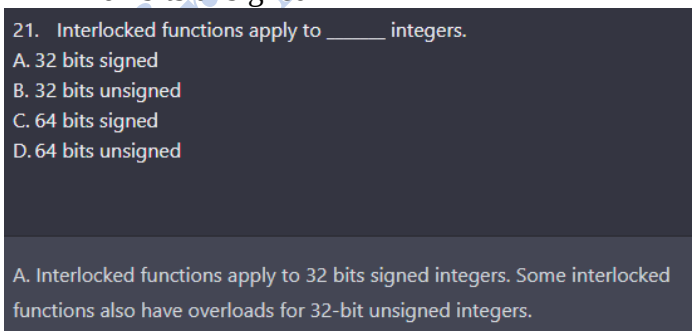
19. The return type of **CreateFiber()** API is **_____**.
    A. VOID
    B. LPVOID
    C. HANDLE
    D. BOOL

Question No: 20 (Marks: 01) - Please choose the correct option

20. Which one of the following is not a Process or thread synchronization object?
    A. Fiber
    B. Critical Section
    C. Mutex
    D. Semaphore

Question No: 21 (Marks: 01) - Please choose the correct option

21. Interlocked functions apply to _____ integers.
    A. 32 bits signed
    B. 32 bits unsigned
    C. 64 bits signed
    D. 64 bits unsigned



Question No: 22 (Marks: 01) - Please choose the correct option

22. Which one of the following is not a requirement for correct thread code?
    A. Global Storage

Waqar Mehmood MC200403630

B. Volatile Storage
C. Memory Barrier
D. Critical Code Region

Question No: 23 (Marks: 01) - Please choose the correct option

---

23. The return type of **CreateSemaphore()** API is _____.
A. DWORD
B. HANDLE
C. BOOL
D. VOID

Question No: 24 (Marks: 01) - Please choose the correct option

---

24. The code region where only one thread can execute at a time is known as _____.
A. program
B. critical section
C. non – critical section
D. Exception Handler

Question No: 25 (Marks: 01) - Please choose the correct option

---

25. Which one of the following APIs block a thread if another thread is in the section?
A. InitializeCriticalSection
B. EnterCriticalSection
C. DeleteCriticalSection
D. LeaveCriticalSection

Question No: 26 (Marks: 01) - Please choose the correct option

---

26. Which one of the following numbers is good spin count for heap management according to MSDN?
A. 1000
B. 2000
C. 3000
D. 4000

Question No: 27 (Marks: 01) - Please choose the correct option

---

27. Which one of the following concepts limit the number of active, contending threads?
A. Mutual Exclusion
B. Thread Contention
C. Semaphore Throttle
D. Parallelism

Question No: 28 (Marks: 01) - Please choose the correct option

Nasir Mehmood MC200403630

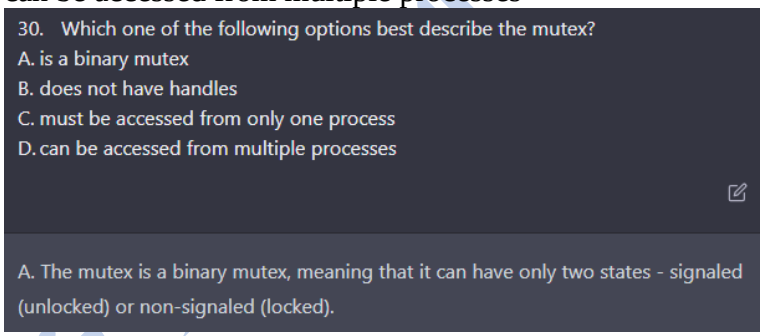
28. Which one of the following APIs returns a pointer to a TP_WORK structure?
    A. TrySubmitThreadpoolCallback
    B. WaitForThreadpoolWorkCallbacks
    C. CreateThreadpoolWork
    D. SubmitThreadpoolWork

Question No: 29 (Marks: 01) - Please choose the correct option

29. Thread synchronization is a process of handling situation when _____ or more threads need access to a shared resource.
    A. One
    B. Two
    C. Three
    D. Four

Question No: 30 (Marks: 01) - Please choose the correct option

30. Which one of the following options best describe the mutex?
    A. is a binary mutex
    B. does not have handles
    C. must be accessed from only one process
    D. can be accessed from multiple processes



Question No: 31 (Marks: 01) - Please choose the correct option

31. What will happen if a non-recursive mutex is locked more than once?
    A. Starvation
    B. Deadlock
    C. Critical Section
    D. Semaphore

Question No: 32 (Marks: 01) - Please choose the correct option

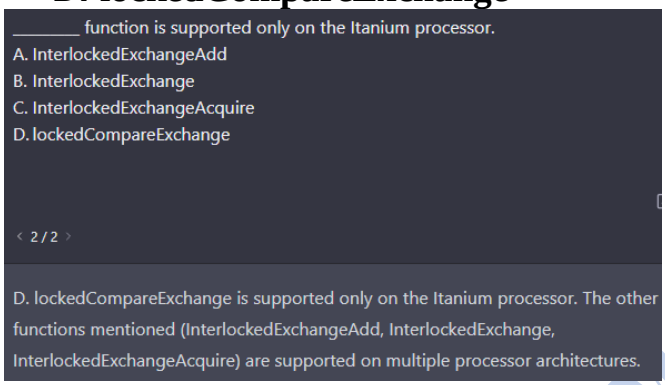
32. What are the two kinds of semaphores?
    A. mutex & counting
    B. binary & counting

MC200403630

C. counting & decimal
D. decimal & binary

Question No: 33 (Marks: 01) - Please choose the correct option

33. _____ function is supported only on the Itanium processor.
A. InterlockedExchangeAdd
B. InterlockedExchange
C. InterlockedExchangeAcquire
D. lockedCompareExchange



Question No: 34 (Marks: 01) - Please choose the correct option

34. Instead of starting a new thread for every task to execute concurrently, the task can be passed to a _____.
A. process
B. thread pool
C. thread queue
D. CPU

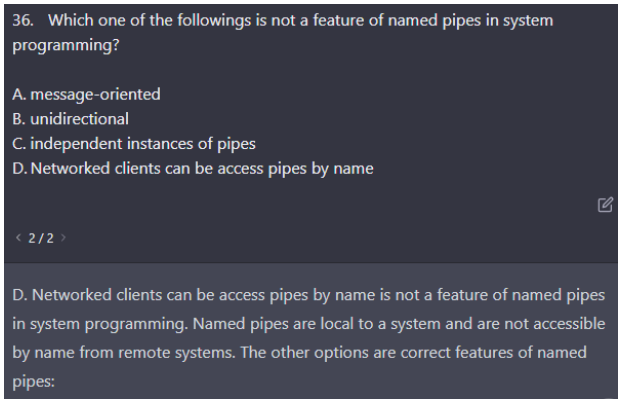
Question No: 35 (Marks: 01) - Please choose the correct option

35. Each process has its own process affinity mask, which is a _____ vector.
A. bit
B. byte
C. Nibble
D. Word

Question No: 36 (Marks: 01) - Please choose the correct option

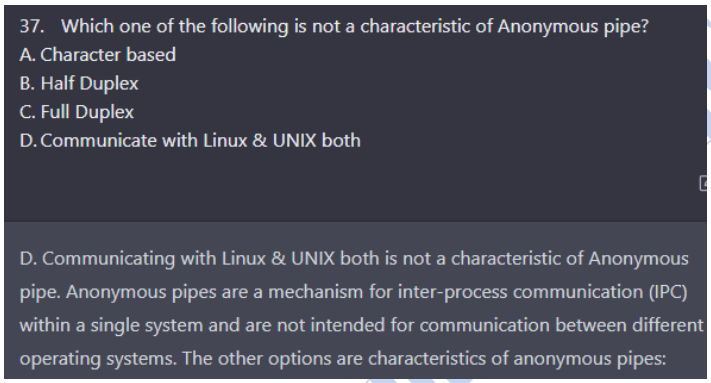
36. Which one of the followings is not a feature of named pipes?
A. message-oriented
B. unidirectional
C. independent instances of pipes
D. Networked clients can be access pipes by name

Nasir Mehmood MC200403630



Question No: 37 (Marks: 01) - Please choose the correct option

37. Which one of the following is not a characteristic of Anonymous pipe?
    A. Character based
    B. Half Duplex
    C. Full Duplex
    D. Communicate with Linux & UNIX both



Question No: 38 (Marks: 01) - Please choose the correct option

38. Which one of the followings has two handles: Read handle and Write handle?
    A. Named Pipes
    B. Anonymous Pipes
    C. Unnamed Pipes
    D. Interprocess Pipes

Question No: 39 (Marks: 01) - Please choose the correct option

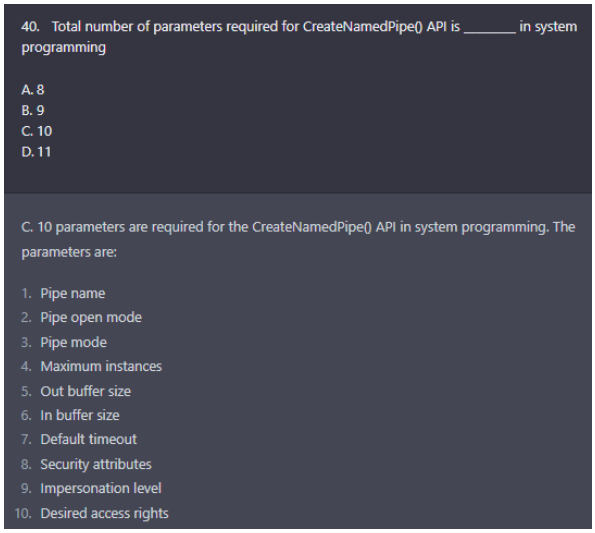
39. The return type of **CreatePipe()** API is _____.
    E. DWORD
    F. HANDLE
    G. BOOL
    H. VOID

Nasir Mehmood 2004403630

Question No: 40 (Marks: 01) - Please choose the correct option

40. Total number of parameters required for **CreateNamedPipe()** API is _____.

A. 8
B. 9
C. 10
D. 11



Question No: 41 (Marks: 03)

What is Explicit Linking and which Windows API is used to load a DLL explicitly inside a program?

Answer

Explicit linking is a technique used to dynamically load a DLL at runtime and call its functions using function pointers. This is different from implicit linking, where the DLL is linked at compile time.

The Windows API used to load a DLL explicitly inside a program is LoadLibrary().

Question No: 42 (Marks: 03)

Name the two Windows APIs that are used to return the pseudo and real handle of a process.

Answer

i) GetCurrentProcess():
ii) OpenProcess()

Nasir Mahmood MC200403630

Question No: 43 (Marks: 03)

Briefly describe the two parameters associated with the following Windows API.
FARPROC **GetProcAddress**(HMODULE hModule, LPCSTR lpProcName)
Answer

The GetProcAddress function is a Windows API used to retrieve the address of an exported function or variable from a dynamic-link library (DLL) specified by the hModule parameter.

hModule: is a handle to the DLL module that contains the function or variable. This handle is obtained by calling the LoadLibrary function or one of its variations.

lpProcName; is a pointer to a null-terminated string that represents the name of the function or variable. If this parameter is set to NULL, the function returns the base address of the module specified by the hModule parameter.

The GetProcAddress function returns a pointer to the exported function or variable specified by lpProcName, or NULL if the function or variable does not exist in the DLL module. The return type of GetProcAddress is FARPROC, which is a pointer to a function that has no specified calling convention. It is up to the caller to cast the return value to the appropriate function pointer type before calling the function.

Question No: 44 (Marks: 03)

Mention the names of two type of events and which function is used to set these types?
Answer

Two types of events in Windows are Manual Reset Event and Auto Reset Event.

The `SetEvent()` function is used to set these types of events.

Question No: 45 (Marks: 03)

Briefly explain system mask and process mask.

Answer

Nasir Mehmood MC200403530

In Windows operating system, the system mask and process mask are used to control the processor affinity of threads.

The system mask is a bit vector that determines the processors that the operating system can use. The processor affinity of the system threads is controlled by the system mask. The system mask is controlled by the operating system and cannot be changed by the application.

On the other hand, the process mask is a bit vector that determines the processors that a specific process can use. The processor affinity of the threads in a process is controlled by the process mask. By default, the process mask is set to the system mask, which means that the threads in the process can run on any available processor. However, the process mask can be changed using the SetProcessAffinityMask function.

Question No: 46 (Marks: 03)

When did the mutex become the abandoned mutex?

Answer

The abandoned mutex functionality was introduced in Windows NT 3.1. Prior to that, if a thread exited while owning a mutex, the mutex would remain locked indefinitely, causing other threads to block indefinitely. The abandoned mutex functionality allows a mutex to be released if the thread owning it terminates unexpectedly, thus preventing other threads from being blocked indefinitely.

Question No: 47 (Marks: 05)

Briefly describe each parameter in the following Widows API.

```
BOOL GetProcessTime(HANDLE hProcess,
                    LPFILETIME lpCreationTime,
                    LPFILETIME lpExitTime,
                    LPFILETIME lpKernelTime,
                    LPFILETIME lpUserTime );
```

Answer

The `GetProcessTime` Windows API is used to retrieve the timing information about a process. Here is the description of the parameters:

- `HANDLE hProcess`: This parameter is the handle to the process for which timing information is to be retrieved.
- `LPFILETIME lpCreationTime`: This parameter is a pointer to a `FILETIME` structure that receives the creation time of the process.
- `LPFILETIME lpExitTime`: This parameter is a pointer to a `FILETIME` structure that receives the exit time of the process. If the process has not exited, the content of this structure is undefined.
- `LPFILETIME lpKernelTime`: This parameter is a pointer to a `FILETIME` structure that receives the amount of time the process has executed in kernel mode.
- `LPFILETIME lpUserTime`: This parameter is a pointer to a `FILETIME` structure that receives the amount of time the process has executed in user mode.

Question No: 48 (Marks: 05)

Write the name of functions or APIs used for the following tasks.

1. Creating a job object
2. Opening a named job object
3. Adding a process to a job object
4. To destroy or close job object

Answer

---

1. CreateJobObject()
2. OpenJobObject()
3. AssignProcessToJobObject()
4. CloseHandle() or TerminateJobObject()

Question No: 49 (Marks: 05)

Which Windows API is used to release the ownership of a mutex? Also write its signature.

Answer

The Windows API used to release the ownership of a mutex is `ReleaseMutex`.

Signature:

Nasir Mehmood MC200403

```
BOOL ReleaseMutex(
  HANDLE hMutex
);
```

This API takes one parameter, `hMutex`, which is a handle to the mutex object. It releases ownership of the specified mutex object. If the function succeeds, the return value is `TRUE`. If the function fails, the return value is `FALSE`.

Question No: 50 (Marks: 05)

What is the purpose of dwPipeMode parameter in CreateNamedPipe() API and mention the names of its two commonly used values.

Answer

The `dwPipeMode` parameter in the `CreateNamedPipe()` API is used to specify the pipe mode and the type of the pipe.

Two commonly used values for the `dwPipeMode` parameter are:

- `PIPE_TYPE_BYTE`: The pipe is a byte-mode pipe. Data is written to the pipe as a stream of bytes, and read from the pipe as a stream of bytes.
- `PIPE_READMODE_MESSAGE`: The pipe is a message-mode pipe. Data is written to the pipe as a stream of messages, and read from the pipe as a stream of messages.

The `dwPipeMode` parameter can be combined with other flags to specify additional pipe modes and options.

Nasir Iqbal 03004403630

Virtual University	**CS609** **SYSTEM PROGRAMMING** **Update (Topics)** **Current Final Term Subjective 2023**	02/03/2023
100% Correct	**Solved by Faisal (J.c FIA)**	**VU**

Question# 1: What is Explicit Linking and which Windows API is used to load a DLL explicitly inside a program?

Answer: Explicit linking is a method of loading dynamic-link libraries (DLLs) in Windows programs. When a program uses explicit linking, it loads the DLL manually at runtime and explicitly calls the functions within the DLL that it needs.

The Windows API used to load a DLL explicitly inside a program is called "LoadLibrary". The LoadLibrary function loads the specified DLL into the address space of the calling process and returns a handle to the loaded module. Once a module is loaded, the program can use the GetProcAddress function to obtain the address of a function within the DLL, which can then be called like any other function.

Explicit linking is useful when a program only needs to use certain functions within a DLL and doesn't want to incur the overhead of loading the entire DLL into memory.

Question#2: Name the two Windows APIs that are used to return the pseudo and real handle of a process.

Answer: The two Windows APIs used to return the pseudo and real handle of a process are:

1.**GetCurrentProcess**(): This function returns a pseudo handle to the current process. A pseudo handle is a special constant that represents the current process and does not need to be closed.

2. OpenProcess(): This function opens a real handle to a specified process. It takes two parameters: the first parameter is the desired access to the process (such as read, write, or execute), and the second parameter is a Boolean value indicating whether the handle should be inheritable by child processes. The function returns a real handle to the specified process if successful, or NULL if an error occurred.

Question#3: Briefly describe the two parameters associated with the following Windows API. FARPROC GetProcAddress(HMODULE hModule, LPCSTR lpProcName).

Answer: The Windows API "GetProcAddress" is used to retrieve the address of an exported function or variable from a specified dynamic-link library (DLL). It takes two parameters:

HMODULE hModule: This parameter is a handle to the loaded module (DLL) that contains the function or variable to be retrieved. If this parameter is NULL, the function retrieves the address of the function or variable from the current process's executable file.

LPCSTR lpProcName: This parameter is a pointer to a null-terminated string that specifies the name of the function or variable to be retrieved. If this parameter is an ordinal value, rather than a string, it represents the numeric value that corresponds to the function's position in the DLL's export table.

Faisal (J.c FIA)

Question#4: Briefly explain system mask and process mask.

Answer:

System mask and process mask refer to sets of processor affinity masks that determine which processors a system or a process is allowed to use.

System mask: The system mask is a bit mask that specifies which processors are available for use by the entire system. By default, all processors are included in the system mask. However, system administrators can modify the system mask to exclude certain processors from use.

Process mask: The process mask is a bit mask that specifies which processors a specific process is allowed to use. By default, a process is allowed to run on any processor in the system. However, the process mask can be set using the SetProcessAffinityMask function to restrict a process to a subset of the available processors.

Restricting the processor affinity of a process can have performance implications. On one hand, restricting a process to a subset of available processors can reduce contention for shared system resources such as memory and I/O. On the other hand, it can also result in decreased performance if the process is CPU-bound and the restricted set of processors is insufficient to handle its workload.

Question#5: Briefly describe each parameter in the following Widows API.

```
BOOL GetProcessTime(HANDLE hProcess,
                    LPFILETIME lpCreationTime,
                    LPFILETIME lpExitTime,
                    LPFILETIME lpKernelTime,
                    LPFILETIME lpUserTime );
```

Answer: The Windows API "GetProcessTimes" is used to retrieve timing information about a specified process. It takes five parameters:

HANDLE hProcess: This parameter is a handle to the process for which timing information is being retrieved. This handle must have the PROCESS_QUERY_INFORMATION or PROCESS_QUERY_LIMITED_INFORMATION access right.

LPFILETIME lpCreationTime: This parameter is a pointer to a FILETIME structure that receives the creation time of the process.

LPFILETIME lpExitTime: This parameter is a pointer to a FILETIME structure that receives the exit time of the process. If the process has not exited, this parameter receives a NULL value.

LPFILETIME lpKernelTime: This parameter is a pointer to a FILETIME structure that receives the amount of time that the process has spent executing in kernel mode.

LPFILETIME lpUserTime: This parameter is a pointer to a FILETIME structure that receives the amount of time that the process has spent executing in user mode.

Question#6: What is the purpose of dwPipeMode parameter in CreateNamedPipe() API and mention the names of its two commonly used values.

Answer: The dwPipeMode parameter in the CreateNamedPipe() API function specifies the mode of the named pipe being created. It determines how data can be transferred through the pipe and how the pipe behaves in different situations.

The dwPipeMode parameter can be set to one or more of the following flags:

PIPE_WAIT: This flag specifies that the pipe is in blocking mode, which means that ReadFile and WriteFile operations will block until data is available or can be written.

PIPE_NOWAIT: This flag specifies that the pipe is in non-blocking mode, which means that ReadFile and WriteFile operations will return immediately, even if there is no data available or the pipe is full.

Faisal (J.c FIA)

PIPE_READMODE_BYTE: This flag specifies that the pipe operates in byte mode, which means that data is read and written in byte-sized blocks.
PIPE_READMODE_MESSAGE: This flag specifies that the pipe operates in message mode, which means that data is read and written in message-sized blocks.
PIPE_TYPE_BYTE: This flag specifies that the pipe is a byte-type pipe, which means that the data is transferred as a stream of bytes.
PIPE_TYPE_MESSAGE: This flag specifies that the pipe is a message-type pipe, which means that the data is transferred as a series of messages.

Question#7: Which Windows API is used to release the ownership of a mutex? Also write its signature.

Answer: The Windows API used to release the ownership of a mutex is called ReleaseMutex().
The signature of the ReleaseMutex() API is as follows:

```
BOOL ReleaseMutex(
HANDLE hMutex
);
```

HANDLE hMutex: This parameter is a handle to the mutex object. The calling thread must be the owner of the mutex. If the calling thread is not the owner of the mutex, the function will fail and return zero.

Question#8: Write the four function names of semaphores?

Answer:

1. CreateSemaphore()
2. CreateSemaphoreEx()
3. OpenSemaphore()
4. ReleaseSemaphore()

Question#9: **Write the types of Semaphores?**

Answer: **There are four types of Semaphores:**

1. Creating a semaphore
2. Deleting a semaphore
3. Waiting a semaphore
4. Releasing a semaphore

Question#10: This Answer is important for mcqs and short question?

Answer:

GetCurrentThreadId()
Obtains thread Id rather than the handle.

GetThreadId()
Obtains thread Id from thread handle.

GetCurrentThread()
Non-inheritable pseudohandle to the calling thread.

Faisal (J.c FIA)

OpenThread()
Creates Thread handle from the thread Id.

Thread Identity
Thread Ids and handles can be obtained using functions quite similar to one used with processes.

Question#11: Name any thread party frameworks that are used for thread optimization.
Answer: OpenMP and cilk++

Question#12: **Synchronization mutex thread synchronization mutex thread model?**
Answer: Synchronization refers to the coordination of multiple threads or processes to ensure that they access shared resources in a mutually exclusive and coordinated manner. The **mutex model** is widely used in multithreaded programming to prevent race conditions and ensure data consistency in shared resources.

Question#12: Syntax allocate and deallocate function?
Answer: allocate and deallocate are functions used in programming languages to dynamically allocate and deallocate memory during runtime.

syntax for allocate and deallocate functions in C++
Allocate Function
pointer_variable = new data_type [size];
Deallocate Function
delete [] pointer_variable;

Question#13: Interlocked Functions?
Answer: The Interlocked functions are a set of functions provided by the Windows operating system to provide atomic operations on shared variables that are accessed concurrently by multiple threads.
Interlocked functions include
InterlockedIncreament()
InterlockedDecreament().
InterlockedExchange
InterlockedCompareExchange

Question#14: Types of callback functions?
Answer:
1. CreateThreadpoolTimer()
2. CreateThreadpoolIO
3. CreateThreadpoolWait()

Question#15: **Critical section?**
Answer: Critical section is the part of the code that can only executed by one thread at a time. Windows provides the CRITICAL_SECTION objects as a simple lock mechanism for solving critical section problem.

Question#15: When thread Object is create, the space stack memory?
Answer: When a thread is created, the operating system allocates memory to hold the thread's execution

Faisal (J.c FIA)

context, which includes the thread's stack space. The stack is a portion of memory that is used to store local variables, function parameters, and return addresses for the functions called by the thread.

Question#16: **ASCII encoded or Unicode format.**

Answer: ASCII and Unicode are two different character encoding schemes used to represent text in computers. ASCII is the most common character-encoding format for text data in computers and on the internet. In standard ASCII-encoded data, there are unique values for 128 alphabetic, numeric or special additional characters.

Question#17: InterlockedExchangeAdd write the parameter or return type.

Answer:

```
Long InterlockedExchangeAdd(
LONG volatile *Addend
LONG Increment )
```

Question#18: Write the GetProcessAffinityMask parameters, API etc.

Answer: most important question.

```
BOOL GetProcessAffinityMask (
    HANDLE hProcess,
    LPDWORD lpProcessAffinityMask,
    LPDWORD lpSystemAffinityMask)
```

Question#19: SetThreadAffinityMask?

Answer: very important

```
DWORD_PTR SetThreadAffinityMask (
    HANDLE hThread,
    DWORD dwThreadAffinityMask)
```

Question#20: CreatePipe parameters Api.

Answer:

```
BOOL CreatePipe (
    PHANDLE phRead,
    PHANDLE phWrite,
    LPSECURITY_ATTRIBUTES lpsa,
    DWORD cbPipe)
```

Question#21: General syntax of Interlockedincrement and InterlockedDecrement.

Answer:

Long InterlockedIncrement(LONG volatile *Addend)
Long InterlockedDecrement(LONG volatile *Addend)

Question#22: Write the name of API that is used to remove the lock from the locked file area?

Answer: The API used to remove a lock from a locked file area depends on the operating system.
In C/C++, the **flock** function can be used to lock and unlock a file region on Unix-based systems.

Question#23: In what condition, the termination Handler is ignored by a process?

Answer: Some conditions under which the termination handler may be ignored by a process.
The process has installed a signal handler that overrides the default termination handler for the SIGTERM signal. **The process** is unable to execute the termination handler due to resource constraints, such as a lack of available memory or CPU time.

Question#24: Which window function is used to convert thread into fiber? (3 marks)

Answer: Converting a thread into a fiber is a programming concept that involves creating a lightweight thread that can be cooperatively scheduled within a single operating system thread. the concept of converting a thread into a fiber is not directly related to window functions, and there is no specific window function that is used for this purpose.

Question#25: Write the name of the API which is used to map the view of a file mapping into the address space of a process?

Answer: **MapViewOfFile** is a function in the Windows API that is used to map a view of a file mapping object into the address space of a calling process. **MapViewOfFile** is typically used in conjunction with other file mapping APIs, such as CreateFileMapping and OpenFileMapping, to implement efficient file I/O operations in Windows applications.

Question#26: Suppose we have a semaphore named as hThrottleSem objects current count is to be increased by 1 and the previous count is not required. you need to write the code to release this semaphore using the above mentioned information. Note: Make use of Windows API to release semaphore.

Answer: ReleaseSemaphore function. code snippet to release the hThrottleSem semaphore and increase its count by 1:

ReleaseSemaphore(hThrottleSem, 1, NULL);

Question#27: Pointer decleration Windows: LPTSTR pFile = Null; define two based ponters on the above ponter. take pSize and pIn as their names .their types should be DWORD and TCHAR respectively.

Answer: Assuming that TCHAR is defined as a wide-character (Unicode) character type, here's how you can define two base pointers pSize and pIn based on the LPTSTR pointer pFile:

Code:

```
DWORD* pSize = NULL;
TCHAR** pIn = NULL;
pSize = (DWORD*)pFile;
pIn = (TCHAR**)(pSize + 1);
```

In the code above, pSize is a pointer to a DWORD (i.e., a 32-bit unsigned integer) and pIn is a pointer to a pointer to a wide-character string (TCHAR*).

Question#28: What is the purpose of *Addend and Value parameter used in InterlockedExchangeAdd() API? (5 Marks)

Answer: **InterlockedExchangeAdd** is a function provided by the Windows API for performing an atomic addition operation on a variable.

LONG InterlockedExchangeAdd(LONG volatile *Addend, LONG Value);

The Addend parameter is a pointer to the variable to which the atomic addition operation should be applied, and Value is the value to be added to the variable. The **InterlockedExchangeAdd** function adds the value specified by Value to the variable pointed to by Addend, and returns the original value of the variable before the addition took place.

Question#29: VOID sleep(DWORD dwMilliseconds); i. In above API if the parameter dwMilliseconds is specified as INFINITE then what happened with threads. ii. If the parameter dwMilliseconds is specified as 0 then what happened with threads? (5 Marks)

Answer: The sleep function is a Windows API function that suspends the execution of the calling thread for a specified number of milliseconds. Here are the answers to your questions:

If the dwMilliseconds parameter is specified as INFINITE, the calling thread will be suspended indefinitely, which means that it will not be scheduled for execution by the operating system until one of the following occurs:

The thread is unblocked by a call to a synchronization object such as WaitForSingleObject or WaitForMultipleObjects. The thread is terminated by another thread. In other words, setting dwMilliseconds to INFINITE will cause the calling thread to enter a wait state that can only be exited by an external event.

Faisal (J.c FIA)

Q.30: How many values can be passing to the nSTDhandle. Write their names?

Ans: Three types of values can be pass to the nSTDHandle

1. STD_INPUT_HANDLE
2. STD_OUPUT_HANDLE
3. STD_ERROR_HANDLE

Q.31: Pre-defined data types required for Windows APIs?

Ans:

BOOL (for storing a single logical value)
HANDLE (a handle for object)
LPTSTR (a string pointer)
DWORD (32-bit unsigned integer)

Q.32: Write the types of floating-point exception?

Ans: Five types of floating-point exception

1. Invalid operation
2. Division by zero
3. Overflow
4. Underflow
5. Inexact calculation.

Q.33: Write names of setting pointer?

Ans: SetFilePointer() SetFilePointerEx()

Q.34: Write the names of five predefined root keys?

Ans:

1. HKEY_LOCAL_MACHINE or HKLM
2. HKEY_CURRENT_CONFIG or HKCC
3. HKEY_CLASSES_ROOT or HKCR
4. HKEY_CURRENT_USER or HKCU
5. HKEY_USERS or HKU

Q.35: How many standard IO devices used in Operating system.

Ans: There are three standard IO devices used in Operating system.

1. Input
2. Output
3. Error

Faisal (J.c FIA)

inprotected.com

Virtual University	**CS609** **SYSTEM PROGRAMMING** **Update (Topics)** **Current Final Term Subjective 2023**	01/03/2023
100% Correct	**Solved by Faisal (J.c FIA)**	**VU**

Question# 1: What is Explicit Linking and which Windows API is used to load a DLL explicitly inside a program?

Answer: Explicit linking is a method of loading dynamic-link libraries (DLLs) in Windows programs. When a program uses explicit linking, it loads the DLL manually at runtime and explicitly calls the functions within the DLL that it needs.

The Windows API used to load a DLL explicitly inside a program is called "LoadLibrary". The LoadLibrary function loads the specified DLL into the address space of the calling process and returns a handle to the loaded module. Once a module is loaded, the program can use the GetProcAddress function to obtain the address of a function within the DLL, which can then be called like any other function.

Explicit linking is useful when a program only needs to use certain functions within a DLL and doesn't want to incur the overhead of loading the entire DLL into memory.

Question#2: Name the two Windows APIs that are used to return the pseudo and real handle of a process.

Answer: The two Windows APIs used to return the pseudo and real handle of a process are:

1.**GetCurrentProcess**(): This function returns a pseudo handle to the current process. A pseudo handle is a special constant that represents the current process and does not need to be closed.

2. OpenProcess(): This function opens a real handle to a specified process. It takes two parameters: the first parameter is the desired access to the process (such as read, write, or execute), and the second parameter is a Boolean value indicating whether the handle should be inheritable by child processes. The function returns a real handle to the specified process if successful, or NULL if an error occurred.

Question#3: Briefly describe the two parameters associated with the following Windows API. FARPROC GetProcAddress(HMODULE hModule, LPCSTR lpProcName).

Answer: The Windows API "GetProcAddress" is used to retrieve the address of an exported function or variable from a specified dynamic-link library (DLL). It takes two parameters:

HMODULE hModule: This parameter is a handle to the loaded module (DLL) that contains the function or variable to be retrieved. If this parameter is NULL, the function retrieves the address of the function or variable from the current process's executable file.

LPCSTR lpProcName: This parameter is a pointer to a null-terminated string that specifies the name of the function or variable to be retrieved. If this parameter is an ordinal value, rather than a string, it represents the numeric value that corresponds to the function's position in the DLL's export table.

Faisal (J.c FIA)

Question#4: Briefly explain system mask and process mask.

Answer:

System mask and process mask refer to sets of processor affinity masks that determine which processors a system or a process is allowed to use.

System mask: The system mask is a bit mask that specifies which processors are available for use by the entire system. By default, all processors are included in the system mask. However, system administrators can modify the system mask to exclude certain processors from use.

Process mask: The process mask is a bit mask that specifies which processors a specific process is allowed to use. By default, a process is allowed to run on any processor in the system. However, the process mask can be set using the SetProcessAffinityMask function to restrict a process to a subset of the available processors.

Restricting the processor affinity of a process can have performance implications. On one hand, restricting a process to a subset of available processors can reduce contention for shared system resources such as memory and I/O. On the other hand, it can also result in decreased performance if the process is CPU-bound and the restricted set of processors is insufficient to handle its workload.

Question#5: Briefly describe each parameter in the following Widows API.

```
BOOL GetProcessTime(HANDLE hProcess,
                    LPFILETIME lpCreationTime,
                    LPFILETIME lpExitTime,
                    LPFILETIME lpKernelTime,
                    LPFILETIME lpUserTime );
```

Answer: The Windows API "GetProcessTimes" is used to retrieve timing information about a specified process. It takes five parameters:

HANDLE hProcess: This parameter is a handle to the process for which timing information is being retrieved. This handle must have the PROCESS_QUERY_INFORMATION or PROCESS_QUERY_LIMITED_INFORMATION access right.

LPFILETIME lpCreationTime: This parameter is a pointer to a FILETIME structure that receives the creation time of the process.

LPFILETIME lpExitTime: This parameter is a pointer to a FILETIME structure that receives the exit time of the process. If the process has not exited, this parameter receives a NULL value.

LPFILETIME lpKernelTime: This parameter is a pointer to a FILETIME structure that receives the amount of time that the process has spent executing in kernel mode.

LPFILETIME lpUserTime: This parameter is a pointer to a FILETIME structure that receives the amount of time that the process has spent executing in user mode.

Question#6: What is the purpose of dwPipeMode parameter in CreateNamedPipe() API and mention the names of its two commonly used values.

Answer: The dwPipeMode parameter in the CreateNamedPipe() API function specifies the mode of the named pipe being created. It determines how data can be transferred through the pipe and how the pipe behaves in different situations.

The dwPipeMode parameter can be set to one or more of the following flags:

PIPE_WAIT: This flag specifies that the pipe is in blocking mode, which means that ReadFile and WriteFile operations will block until data is available or can be written.

PIPE_NOWAIT: This flag specifies that the pipe is in non-blocking mode, which means that ReadFile and WriteFile operations will return immediately, even if there is no data available or the pipe is full.

Faisal (J.c FIA)

PIPE_READMODE_BYTE: This flag specifies that the pipe operates in byte mode, which means that data is read and written in byte-sized blocks.
PIPE_READMODE_MESSAGE: This flag specifies that the pipe operates in message mode, which means that data is read and written in message-sized blocks.
PIPE_TYPE_BYTE: This flag specifies that the pipe is a byte-type pipe, which means that the data is transferred as a stream of bytes.
PIPE_TYPE_MESSAGE: This flag specifies that the pipe is a message-type pipe, which means that the data is transferred as a series of messages.

Question#7: Which Windows API is used to release the ownership of a mutex? Also write its signature.

Answer: The Windows API used to release the ownership of a mutex is called ReleaseMutex().
The signature of the ReleaseMutex() API is as follows:

```
BOOL ReleaseMutex(
HANDLE hMutex
);
```

HANDLE hMutex: This parameter is a handle to the mutex object. The calling thread must be the owner of the mutex. If the calling thread is not the owner of the mutex, the function will fail and return zero.

Question#8: Write the four function names of semaphores?

Answer:

1. CreateSemaphore()
2. CreateSemaphoreEx()
3. OpenSemaphore()
4. ReleaseSemaphore()

Question#9: **Write the types of Semaphores?**

Answer: **There are four types of Semaphores:**

1. Creating a semaphore
2. Deleting a semaphore
3. Waiting a semaphore
4. Releasing a semaphore

Question#10: This Answer is important for mcqs and short question?

Answer:

GetCurrentThreadId()
Obtains thread Id rather than the handle.

GetThreadId()
Obtains thread Id from thread handle.

GetCurrentThread()
Non-inheritable pseudohandle to the calling thread.

Faisal (J.c FIA)

OpenThread()
Creates Thread handle from the thread Id.

Thread Identity
Thread Ids and handles can be obtained using functions quite similar to one used with processes.

Question#11: Name any thread party frameworks that are used for thread optimization.
Answer: OpenMP and cilk++

Question#12: **Synchronization mutex thread synchronization mutex thread model?**
Answer: Synchronization refers to the coordination of multiple threads or processes to ensure that they access shared resources in a mutually exclusive and coordinated manner. The **mutex model** is widely used in multithreaded programming to prevent race conditions and ensure data consistency in shared resources.

Question#12: Syntax allocate and deallocate function?
Answer: allocate and deallocate are functions used in programming languages to dynamically allocate and deallocate memory during runtime.

syntax for allocate and deallocate functions in C++
Allocate Function
pointer_variable = new data_type [size];
Deallocate Function
delete [] pointer_variable;

Question#13: Interlocked Functions?
Answer: The Interlocked functions are a set of functions provided by the Windows operating system to provide atomic operations on shared variables that are accessed concurrently by multiple threads.
Interlocked functions include
InterlockedIncreament()
InterlockedDecreament().
InterlockedExchange
InterlockedCompareExchange

Question#14: Types of callback functions?
Answer:
1. CreateThreadpoolTimer()
2. CreateThreadpoolIO
3. CreateThreadpoolWait()

Question#15: **Critical section?**
Answer: Critical section is the part of the code that can only executed by one thread at a time. Windows provides the CRITICAL_SECTION objects as a simple lock mechanism for solving critical section problem.

Question#15: When thread Object is create, the space stack memory?
Answer: When a thread is created, the operating system allocates memory to hold the thread's execution

Faisal (J.c FIA)

context, which includes the thread's stack space. The stack is a portion of memory that is used to store local variables, function parameters, and return addresses for the functions called by the thread.

Question#16: **ASCII encoded or Unicode format.**

Answer: ASCII and Unicode are two different character encoding schemes used to represent text in computers. ASCII is the most common character-encoding format for text data in computers and on the internet. In standard ASCII-encoded data, there are unique values for 128 alphabetic, numeric or special additional characters.

Question#17: InterlockedExchangeAdd write the parameter or return type.

Answer:

```
Long InterlockedExchangeAdd(
LONG volatile *Addend
LONG Increment )
```

Question#18: Write the GetProcessAffinityMask parameters, API etc.

Answer: most important question.

```
BOOL GetProcessAffinityMask (
    HANDLE hProcess,
    LPDWORD lpProcessAffinityMask,
    LPDWORD lpSystemAffinityMask)
```

Question#19: SetThreadAffinityMask?

Answer: very important

```
DWORD_PTR SetThreadAffinityMask (
    HANDLE hThread,
    DWORD dwThreadAffinityMask)
```

Question#20: CreatePipe parameters Api.

Answer:

```
BOOL CreatePipe (
    PHANDLE phRead,
    PHANDLE phWrite,
    LPSECURITY_ATTRIBUTES lpsa,
    DWORD cbPipe)
```

Question#21: General syntax of Interlockedincrement and InterlockedDecrement.

Answer:

Long InterlockedIncrement(LONG volatile *Addend)
Long InterlockedDecrement(LONG volatile *Addend)

Question#22: Write the name of API that is used to remove the lock from the locked file area?

Answer: The API used to remove a lock from a locked file area depends on the operating system.
In C/C++, the **flock** function can be used to lock and unlock a file region on Unix-based systems.

Question#23: In what condition, the termination Handler is ignored by a process?

Answer: Some conditions under which the termination handler may be ignored by a process.
The process has installed a signal handler that overrides the default termination handler for the SIGTERM signal. **The process** is unable to execute the termination handler due to resource constraints, such as a lack of available memory or CPU time.

Faisal (J.c FIA)

soprotection.com

FINALTERM EXAMINATION
Spring 2012
CS609- System Programming

Question No: 1 (Marks: 1) - Please choose one
Bit # _____ of Eflag is used for alignment check

►12
►14
►15
►**18 (page 164)**

Question No: 2 (Marks: 1) - Please choose one
Each addressable unit has a unique combination of sec#, head #, track # as its --------- address.

►**Physical (page 202)**
►Logical
►Both
►None

Question No: 3 (Marks: 1) - Please choose one
First cluster in user data is numbered in a FAT based system.

►0
►1
►**2 (page 258)**
►3

دنیا میں سب سے مشکل کام اپنی اصلاح اور سب سے آسان کام دوسروں پر نکتہ چینی کرنا ہے

Muhammad Moaaz Siddiq – MCS(4th)
Moaaz.pk@gmail.com
Campus:- Institute of E-Learning & Moderen Studies (IEMS) Samundari

Question No: 4 (Marks: 1) - Please choose one
BIOS services understand --------.

►**LBA (page 212)**
►LSN
►Cluster #
►None

Question No: 5 (Marks: 1) - Please choose one
The first cluster number of a file can be found in---------

►BPB
►DPB
►**FCB(page 265)**
►None

Question No: 6 (Marks: 1) - Please choose one
The size of FS Info block is

►64byte
►128 byte
►256 byte
►**512 byte(page 300)**

Question No: 7 (Marks: 1) - Please choose one
In NTFS first -------- entries are reserved.

►4
►6
►**16 (page 303)**
►32

Question No: 8 (Marks: 1) - Please choose one
In memory map of first 1 MB of ram ,the first -------- is called conventional RAM.

►64kb
►384kb
►**640kb (page 317)**
►None

بری صحبت سے تنہائی بہتر ہے اور تنہائی سے نیک صحبت بہتر ہے

Muhammad Moaaz Siddiq – MCS(4th)
Moaaz.pk@gmail.com
Campus:- Institute of E-Learning & Moderen Studies (IEMS) Samundari

Question No: 9 (Marks: 1) - Please choose one
In memory map of first 1 MB of ram ,the higher -------- is called system memory.

►64kb
►**384kb (page 317)**
►640kb
►None

Question No: 10 (Marks: 1) - Please choose one
The -------- of boot block constitutes of BPB.

►Code part
►**Data part (page 242)**
►Both
►None

Question No: 11 (Marks: 1) - Please choose one
Extended BIOS function make use of --------- address

►**LBA (Page 212)**
►CHS
►LSN
►None

Question No: 12 (Marks: 1) - Please choose one
LBA address can be used in place of the CHS address.
►**True (Page 235)**
►**False**

Question No: 13 (Marks: 1) - Please choose one
In FAT12, the maximum range of clusters is ……..

►0 ~ FEFH
►1~ FEFH
►**2 ~ FEFH (Page 266)**
►3 ~ FEFH

Question : 14 (Marks: 1) - Please choose one
NTFS volume can be accessed directly in DOS.

►True
►**False (Page 310)**

Muhammad Moaaz Siddiq – MCS(4th)
Moaaz.pk@gmail.com
Campus:- Institute of E-Learning & Moderen Studies (IEMS) Samundari

Question No: 15 (Marks: 1) - Please choose one
Each partition information chunk is 16 bytes long and the last two bytes at the end of the partition table data part is the partition table signature whose value should be _____ indicating that the code part contains valid executable code.

►00AA
►0055
►050A
►AA55 (Page 219)

Question No: 16 (Marks: 1) - Please choose one
Service 21H/52H service returns the address of DOS internal data structures in ES: BX _____behind the address returned lies the far address of the first MCB in memory.

►2-bytes
►4-bytes (Page 322)
►6-bytes
►8-bytes

Question No: 17 (Marks: 1) - Please choose one
80386 can have _____ control registers.

►2
►5
►3
►4 (Page 331)

Question No: 18 (Marks: 1) - Please choose one
The partition table uses the extended _____ service.
►13H (p234)
►14H
►15H
►16H

Question No: 19 (Marks: 1) - Please choose one
The entry point of execution in EXE File can be

►Start of the first instruction
►Start of the last instruction
►Anywhere in the Program (Page 335)
►Can be in the middle of the program

اللہ کا خوف سب سے بڑی دانائی ہے

Muhammad Moaaz Siddiq – MCS(4th)
Moaaz.pk@gmail.com
Campus:- Institute of E-Learning & Moderen Studies (IEMS) Samundari

Question : 20 (Marks: 1) - Please choose one
Using the____ entry and the FAT we can access the contents of file.

►Reserved blocks
►**Root Directory (Page 269)**
►Number of FAT copies
►None of the given

Question : 21 (Marks: 1) - Please choose one
Control information in files is maintained using
►BPB
►DPB
►**FCB (Page 256)**
►FPB

Question : 22 (Marks: 1) - Please choose one
What will happen if NTFS volume is accessed in DOS?

►Convert it to FAT volume
►Nothing will happen
►**Error of invalid media (Page 310)**
►None of the given

Question : 23 (Marks: 1) - Please choose one
LSN of FS Info block is available at

►BPB
►**FAT**
►Root Directory
►None of the given

Question : 24 (Marks: 1) - Please choose one
DOS device drivers do not understand the_____ data structures.

►FAT12
►FAT16
►FAT32
►**NTFS (Page 310)**

Muhammad Moaaz Siddiq – MCS(4th)
Moaaz.pk@gmail.com
Campus:- Institute of E-Learning & Moderen Studies (IEMS) Samundari

Question : 25 (Marks: 1) - Please choose one
_____ is a collection of contagious blocks.

►**Cluster (Page 242)**
►Sector
►Byte
►None of Given

Question No: 26 (Marks: 1) - Please choose one
_____ used to determine the amount of conventional memory interfaced with the processor in kilobytes.

►INT 10 H
►INT 11 H
►**INT 12 H (Page 162)**
►INT 13 H

Question No: 27 (Marks: 1) - Please choose one
Bit number _____ of coprocessor control word is the Interrupt Enable Flag.
►**7 (Page 168)**
►8
►9
►10

Question No: 28 (Marks: 1) - Please choose one
To distinguish 486 with Pentium CPUID Test is used.
►**True (Page 166)**
►False

Question : 29 (Marks: 1) - Please choose one
Practically _____ entries are there in FAT 32.
►2^26
►2^28
►2^30
►**2^32 (Page 265)**

Question No: 30 (Marks: 1) - Please choose one
BPB stands for _____.

►**BIOS parameter block (Page 243)**
►BIOS processing block
►Base processing block
►BIOS partition block

Muhammad Moaaz Siddiq – MCS(4th)
Moaaz.pk@gmail.com
Campus:- Institute of E-Learning & Moderen Studies (IEMS) Samundari

Question No: 31 (Marks: 1) - Please choose one
The keyboard input character scan code is received at ___ port.

►**60H (Page 179)**
►61H
►62H
►63H

Question No: 32 (Marks: 1) - Please choose one
_____ is LED control byte.

►0xFD
►**0xED (Page 181)**
►0xFF
►0xEE

Question : 33 (Marks: 1) - Please choose one
_____means typematic rate will be sent in next byte.

►**0xF3 (Page 180)**
►0xF4
►0xF5
►0xF6

Question : 34 (Marks: 1) - Please choose one
Keyboard uses port ____ as status port.

►**64H (Page 177)**
►66H
►67H
►69H

Question : 35 (Marks: 1) - Please choose one
The keyboard can perform _____ serial I/O.

►asynchronous
►**synchronous**
►Multiple
►Single

ایماندار کو غصہ دیر سے آتا ہے اور جلدی دور ہو جاتا ہے

Muhammad Moaaz Siddiq – MCS(4th)
Moaaz.pk@gmail.com
Campus:- Institute of E-Learning & Moderen Studies (IEMS) Samundari

Question : 36 (Marks: 1) - Please choose one
Bit number 2 of port 64H Status register used for output buffer full.

►True
►**False**

Question : 37 (Marks: 1) - Please choose one
Bit number _____ can declares the parity error of port 64H Status register.

►4
►5
►6
►**7**

Question : 38 (Marks: 1) - Please choose one
Bit number _____ of port 64H Status register used for input buffer full.

►**0**
►1
►2
►3

زندگی میں کامیابی کا یہی راز ہے کہ پریشانیوں سے پریشان مت بنو

Muhammad Moaaz Siddiq – MCS(4th)
Moaaz.pk@gmail.com
Campus:- Institute of E-Learning & Moderen Studies (IEMS) Samundari

FINALTERM EXAMINATION
Spring 2010
CS609- System Programming

Question No: 1 (Marks: 1) - Please choose one
Maximum possible entries in FAT12 are _____.

►1024
►2048
►4096 (Page 264)
►65536

Question No: 2 (Marks: 1) - Please choose one
Disadvantage of FAT32 is _____.

►Large disk size can be managed in FAT32
►Cluster size is reduced
►Internal fragmentation is reduced
►Very large table (Page 299)

Question No: 3 (Marks: 1) - Please choose one
What will be the value of the word located at 1Fh in DPB when number of free clusters on drive is not known?

►0000H
►1111H
►FFFFH (Page 250)
►None of the given.

Question No: 4 (Marks: 1) - Please choose one
Jump code part contains ____ bytes in boot block.

►3 (Page 302)
►5
►8
►11



Muhammad Moaaz Siddiq – MCS(4th)
Moaaz.pk@gmail.com
Campus:- Institute of E-Learning & Moderen Studies (IEMS) Samundari

Question : 5 (Marks: 1) - Please choose one
Operating system name contains ____ bytes in boot block.

►3
►5
►8 (Page 257)
►11

Question No: 6 (Marks: 1) - Please choose one
File can be _____ viewed as organization of data.

►Physically
►Logically (Page 256)
►Both logically and physically
►None of the give

Question No: 7 (Marks: 1) - Please choose one
_____ is used to read a block against its LSN.

►absread() (Page 247)
►abswrite()
►lsnread()
►None of the given

Question : 8 (Marks: 1) - Please choose one
File can be_____ viewed as collection of clusters or blocks.

►Physically (Page 256)
►Logically
►Both physically and logically
►None

Question No: 9 (Marks: 1) - Please choose one
When we talk about FAT based file system, in user data area first cluster number is _____.

►0
►1
►2 (Page 258)
►None of the given

جھوٹ انسان اور ایمان دونوں کا دشمن ہے

Muhammad Moaaz Siddiq – MCS(4th)
Moaaz.pk@gmail.com
Campus:- Institute of E-Learning & Moderen Studies (IEMS) Samundari

Question No: 10 (Marks: 1) - Please choose one
Cluster number can also be referred as block number.
►True
►**False (Page 258)**

Question : 11 (Marks: 1) - Please choose one
To access the block within cluster using BIOS services the cluster number should be converted into _____.

►CHS
►LBA
►**LSN (Page 258)**
►None of the given

Question : 12 (Marks: 1) - Please choose one
What will be the value of DL register when we are accessing C drive using undocumented service 21H/32H?

►0
►1
►2
►**3 (Page 249)**

Question No: 13 (Marks: 1) - Please choose one
The directory structure of DOS is like _____.

►Array
►**Tree (Page 256)**
►Linked list
►None of the given

Question No: 14 (Marks: 1) - Please choose one
Control information about files is maintained using _____.
►BPB
►DPB
►**FCB (Page 256)**
►FPB

Question No: 15 (Marks: 1) - Please choose one
When LSN is equal to zero (0), it means _____.

► First block of the disk
► **First block of the logical drive (Page 240)**
►First block of hidden blocks
► None of the given

Muhammad Moaaz Siddiq – MCS(4th)
Moaaz.pk@gmail.com
Campus:- Institute of E-Learning & Moderen Studies (IEMS) Samundari

Question No: 16 (Marks: 1) - Please choose one
In FAT32, lower _____ bits are used.
► 26
► 28 (Page 292)
►30
►32

Question No: 17 (Marks: 1) - Please choose one
_____ is relative address with respect to the start of Logical Drive.
►LBA
►LSN (Page 240)
►CHS
►None of the given

Question No: 18 (Marks: 1) - Please choose one
The practical limit of blocks per cluster is _____.

►32 blocks per cluster
►64 blocks per cluster (Page 242)
►128 blocks per cluster
►256 blocks per cluster

Question No: 19 (Marks: 1) - Please choose one
In dos we have limit of _____ .
►128 blocks per cluster (Page 242)
►256 blocks per cluster
►32 blocks per cluster
►64 blocks per cluster

Question No: 20 (Marks: 1) - Please choose one
Highest capacity physical capacity of the disk according to the IDE interface is _____.
►127 GB (Page 212)
►100 GB
►80 GB
►300 GB

Question No: 21 (Marks: 1) - Please choose one
Partition Table can be read using the extended _____ Services.

►13 H (Page 234)
►14 H
►15 H
►None of given

Muhammad Moaaz Siddiq – MCS(4th)
Moaaz.pk@gmail.com
Campus:- Institute of E-Learning & Moderen Studies (IEMS) Samundari

Question No: 22 (Marks: 1) - Please choose one
In Protected Mode, the segment registers are used as _____

►Descriptor
►**Selector (Page 326)**
►All of the given choices
►None of the given choices

Question No: 23 (Marks: 1) - Please choose one
To access drive parameter block we use undocumented service _____

►09H/32H
►11H/32H
►17H/32H
►**21H/32H (Page 249)**

Question No: 24 (Marks: 1) - Please choose one
_____ is an absolute address relative to the start of physical drive.

►**LBA (Page 240)**
►LSN
►CHS
►None of the above

Question No: 25 (Marks: 1) - Please choose one
Boot block consists of _____ bytes.

►64
►128
►256
►**512 (Page 242)**

Question No: 26 (Marks: 1) - Please choose one
The DMA requests to acquire buses through the _____ signal.

►**HOLD (Page 186)**
►ACR
►ACK
►None of Given

عقل مند کہتا ہے میں کچھ نہیں جانتا جبکہ بے وقوف کہتا ہے کہ میں سب کچھ جانتا ہوں

Muhammad Moaaz Siddiq – MCS(4th)
Moaaz.pk@gmail.com
Campus:- Institute of E-Learning & Moderen Studies (IEMS) Samundari

Question No: 27 (Marks: 1) - Please choose one
The keyboard device writes a code 0xFA on the port 60H to indicate that the _____.

►Input buffer is full
►**Byte has been received properly (Page 179)**
►Output buffer is full
►None of the given

Question No: 28 (Marks: 1) - Please choose one
A single DMA can transfer _____ operands to and from memory in a single a bus cycle.

►**8-bits (Page 186)**
►16-bits
►32-bits
►12-bits

Question No: 29 (Marks: 1) - Please choose one
In FAT12, to calculate the address or offset from index, we need to multiply it with ___.

►1/2
►**3/2 (Page 267)**
►5/7
►7/2

Question No: 30 (Marks: 1) - Please choose one
_____ Register can be used to show that the channel is single transfer, block transfer or demand transfer mode.

►DMA Command register
►DMA Request Register
►DMA Mode Register
►**DMA controller Register (Page 187-188)**

خود کو تمھیں سے بڑھ کر کوئی اچھا مشورہ نہیں دے سکتا

Muhammad Moaaz Siddiq – MCS(4th)
Moaaz.pk@gmail.com
Campus:- Institute of E-Learning & Moderen Studies (IEMS) Samundari

CS609 – Solved Quizzes (Quiz No.3 & 4)

Question : 1 of 10 (Marks: 1) - Please choose one
When we mark a file as deleted by placing 0xE5 then the chain of clusters in FAT is also replaced by _____.

►E5
►1
►0 (Page 79)
►N

Question : 2 of 10 (Marks: 1) - Please choose one
Cluster size is reduced in _____.

►FAT12
►FAT16
►FAT32 Click here for detail
►None of the given

Question : 3 of 10 (Marks: 1) - Please choose one
In FAT32 _____ root directory entries are there.

►128
►256
►512
►None of the given Click here for detail

Question : 4 of 10 (Marks: 1) - Please choose one
If a file is having more than one cluster then it will be managed by _____.

►FAT
►BPB
►DPB
►None of the above

Question : 5 of 10 (Marks: 1) - Please choose one
Internal fragmentation is reduced in _____.

►FAT12
►FAT16 Click here for detail
►FAT32
►None of the given

Muhammad Moaaz Siddiq – MCS(4th)
Moaaz.pk@gmail.com
Campus:- Institute of E-Learning & Moderen Studies (IEMS) Samundari

Question : 6 of 10 (Marks: 1) - Please choose one
For supporting long file names, _____ fragments can be supported.

►12
►20
►26
►32

Question : 7 of 10 (Marks: 1) - Please choose one
To store a cluster in FAT 32 _____ is/are needed.

►Nibble
►Byte
►2 Bytes
►4 Bytes **Click here for detail**

Question : 8 of 10 (Marks: 1) - Please choose one
If a file size is 12K and the size of the cluster is 4K then _____ clusters are used for the file.

►2
►3
►4
►5

Question : 9 of 10 (Marks: 1) - Please choose one
We can access the contents of File by using the root directory entry and _____.

►Reserved Blocks
►Number of FAT copies
►File Allocation Table (FAT) (Page 269)
►None of the given

Question : 10 of 10 (Marks: 1) - Please choose one
FAT based file system can store file name in _____ form.

►ASCII
►UNICODE
►Both ASCII and UNICODE
►None of the given

جو شخص ناکامیوں سے ڈر کر بھاگتا ہے کامیابی اُس سے ڈر کر بھاگتی ہے

Muhammad Moaaz Siddiq – MCS(4th)
Moaaz.pk@gmail.com
Campus:- Institute of E-Learning & Moderen Studies (IEMS) Samundari

Question : 1 of 10 (Marks: 1) - Please choose one
Drive parameter block is derived from _____.

►FCB
►FAT
►**BPB (Page 249)**
►CPB

Question : 2 of 10 (Marks: 1) - Please choose one
We can access Blocks for FAT using _____.
►BPB
►DPB
►**FCB**
►Both BPB and DPB

Question : 3 of 10 (Marks: 1) - Please choose one
If we know the cluster number, we can access the blocks within the cluster using BIOS services directly.

►**True (Page 258)**
►False

Question : 4 of 10 (Marks: 1) - Please choose one
_____ is an internal data structure of DOS and resides in main memory.

►BPB
►DPB
►CPB
►None of the given. Click here for detail

Question : 5 of 10 (Marks: 1) - Please choose one
The size of DPB data structure is _____ bytes.
►16
►32
►64
►**128** click here for detail

Question : 6 of 10 (Marks: 1) - Please choose one
The size of FCB data structure is _____ bytes.
►**16** Click here for detail
►32
►64
►128

Muhammad Moaaz Siddiq – MCS(4th)
Moaaz.pk@gmail.com
Campus:- Institute of E-Learning & Moderen Studies (IEMS) Samundari

Question : 7of 10 (Marks: 1) - Please choose one
Advantages of FAT32 is/are _____.

►Large disk size can be managed in FAT32
►**Cluster size is reduced** Click here for detail
►Internal fragmentation is reduced
►All of the given

Question : 8 of 10 (Marks: 1) - Please choose one
_____ file system keeps the backup of its boot block.

►FAT12
►FAT16
►**FAT32 Click here for detail**
►None of the given

Question : 9 of 10 (Marks: 1) - Please choose one
To store a UNICODE character ____ is/are needed.

►Nibble
►Byte
►**2 Bytes Click here for detail**
►4 Bytes

Question : 10 of 10 (Marks: 1) - Please choose one
_____ is the first block on disk.

►LSN =0
►**LBA=0 (Page 240)**
►LBA=1
►Both LBA=0 and LSN=0

Question : 1 of 10 (Marks: 1) - Please choose one
If FAT entry is between FFF0H to FFF6H in FAT16 then _____.

►Cluster is available
►**It is a Reserved cluster (Page 272)**
►It is next file cluster
►It is a last file cluster

جو لوگوں کے سامنے فخر کرتا ہے وہ لوگوں کی نظروں سے گر جاتا ہے

Muhammad Moaaz Siddiq – MCS(4th)
Moaaz.pk@gmail.com
Campus:- Institute of E-Learning & Moderen Studies (IEMS) Samundari

Question : 2 of 10 (Marks: 1) - Please choose one
File system used in CD's is_____ file system

►**Contiguous** **Click here for detail**
►Chained
►Indexed
►None

Question : 3 of 10 (Marks: 1) - Please choose one
A file has 2 clusters and the size of cluster is 4K. What will be the size of file?

►2K
►**8K**
►16K
►32K

Question : 4 of 10 (Marks: 1) - Please choose one
In NTFS, Backup of boot block is stored at block # _____.

►2
►6
►**8**
►10

Question : 5 of 10 (Marks: 1) - Please choose one
The interval timer can operate in _____ modes.

►Five
►Seven
►Four
►**Six (Page 72)**

Question : 6 of 10 (Marks: 1) - Please choose one
File control block (FCB) is _____ byte long.

►**32** Click here for detail
►64
►16
►128

عقل مند اپنے عیب خود دیکھتا ہے اور بیوقوفوں کے عیب دنیا دیکھتی ہے

Muhammad Moaaz Siddiq – MCS(4th)
Moaaz.pk@gmail.com
Campus:- Institute of E-Learning & Moderen Studies (IEMS) Samundari

Question : 7 of 10 (Marks: 1) - Please choose one
On the execution of IRET instruction, number of bytes popped from stack is

► 4 bytes
► 6 bytes
► **8 bytes** **Click here for detail**
► 10 bytes

Question : 8 of 10 (Marks: 1) - Please choose one
If CPUID instruction is not present then the processor can be a

► **486 processor (Page 166)**
► 386 processor
► 286 processor
► All of the above

Question : 9 of 10 (Marks: 1) - Please choose one
Extended memory is available if the processor is of the type _____

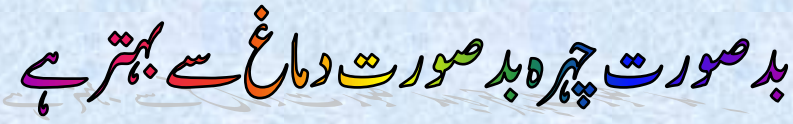
►**AT (Page 171)**
►XT
►All of the given choices
►None of them

Question : 10 of 10 (Marks: 1) - Please choose one
The built in mechanism within the UART for error detection is_____

►hamming code
►**parity (Page 107)**
►CRC16 (cyclic redundancy check 16 bit)
►CRC32 (cyclic redundancy check 32 bit)

Question : 1 of 10 (Marks: 1) - Please choose one
If three Programmable interrupt controllers are cascaded then how many interrupt driven hardware IO devices can be attached _____

►12
►18
►23
►**24 (Page 48)**



Muhammad Moaaz Siddiq – MCS(4th)
Moaaz.pk@gmail.com
Campus:- Institute of E-Learning & Moderen Studies (IEMS) Samundari

Question : 2 of 10 (Marks: 1) - Please choose one
Int 14H_____ can be used to send a byte

►Service#0
►**Service#1 (Page 121)**
►Service#2
►None of the given option.

Question : 3 of 10 (Marks: 1) - Please choose one
Int 14H _____ can be used to set the line parameter of the UART or COM port.

►**Service # 0 (Page 119)**
► Service # 1
►Service # 2
►None of the given options

Question : 4 of 10 (Marks: 1) - Please choose one
Int 14H _____ can be used to receive a byte.

►Service # 0
►Service # 1
►**Service # 2 (Page 121)**
►None of the given options

Question : 5 of 10 (Marks: 1) - Please choose one
The _____ function simply enables the self test facility within the modem control register

►STOn()
►SelfTest()
►**SelfTestOn() (Page 127)**
►None of these

Question : 6 of 10 (Marks: 1) - Please choose one
_____ is a device incorporated into the PC to update time even if the computer is off.

►Clock counter
►ROM
►Clock
►**Real time clock (Page 136)**

عقل مند آدمی اس وقت تک نہیں بولتا جب تک خاموشی نہیں ہو جاتی

Muhammad Moaaz Siddiq – MCS(4th)
Moaaz.pk@gmail.com
Campus:- Institute of E-Learning & Moderen Studies (IEMS) Samundari

Question : 7 of 10 (Marks: 1) - Please choose one
Interrupt _____ is used to get or set the time.

►0AH
►1AH (Page 136)
►2AH
►3AH

Question : 8 of 10 (Marks: 1) - Please choose one
_____ is used to set time.
►1A/02H
►1A/03H (Page 138)
►1A/04H
►1A/05H

Question : 9 of 10 (Marks: 1) - Please choose one
---------- is used to read date from RTC
►1A\02H
►1A\03H
►1A\04H (Page 138)
►1A\05H

Question : 10 of 10 (Marks: 1) - Please choose one
____whenever received indicates the start of communication and _____ whenever received indicates a temporary pause in the communication.

►XON & XOFF (Page 135)
►XOFF & XON
►XON & YOFF
►YON & XOFF

Question : 1 of 10 (Marks: 1) - Please choose one
The_____ function uses the COM port number to receive a byte from the COM port using BIOS services.
►recievebyte()
►initialize ()
►receive()
►recievechar() (Page 125)

Question : 2 of 10 (Marks: 1) - Please choose one
In self test mode the output of the UART is routed to its input.

►True (Page 117)
►False

Muhammad Moaaz Siddiq – MCS(4th)
Moaaz.pk@gmail.com
Campus:- Institute of E-Learning & Moderen Studies (IEMS) Samundari

Question : 3 of 10 (Marks: 1) - Please choose one
Only _____ ports are important from programming point of view.

► **70 and 71H (Page 141)**
►71 and 72H
►70 and 72H
►72 and 73H

Question : 4 of 10 (Marks: 1) - Please choose one
The BIOS interrupt 0x1AH can be used to configure real time clock

►**True (Page 136)**
►False

Question : 5 of 10 (Marks: 1) - Please choose one
DOS command _____ which gives the status of the memory and also points out which memory area occupied by which process.

►**mem/d (Page 13)**
►mem/e
►mem/m
►None of the given

Question : 6 of 10 (Marks: 1) - Please choose one
Each entry in the IVT is _____ in size.

►**4-bytes (Page 12)**
►6-bytes
►8-bytes
►2-bytes

Question : 7 of 10 (Marks: 1) - Please choose one
The tail value of the keyboard buffer should be examined to get to the _____ of the buffer.

►**Start (Page 56)**
►End
►Middle
►None of given)

Question : 8 of 10 (Marks: 1) - Please choose one
Usually interrupt procedures are reentrant procedures especially those interrupt procedure compiled using C language compiler are reentrant.

Muhammad Moaaz Siddiq – MCS(4th)
Moaaz.pk@gmail.com
Campus:- Institute of E-Learning & Moderen Studies (IEMS) Samundari

►**True (Page 38)**
►False

Question : 9 of 10 (Marks: 1) - Please choose one
_____ is Disk interrupt.

►10H
►11H
►**13H (Page 42)**
►14H

Question : 10 of 10 (Marks: 1) - Please choose one
In parallel communication, the maximum numbers of bits we can send between two computers are _____.

►2-bits
►4-bits
►6-bits
►**8-bits**

Question : 1 of 10 (Marks: 1) - Please choose one
14h include _____ which is used to send a byte.

►Service #0
►**Service #1 (Page 121)**
►Service #2
►Service #3

Question : 2 of 10 (Marks: 1) - Please choose one
The status register _____ is the main control register.

►**B (Page 146)**
►A
►C
►D

Question : 3 of 10 (Marks: 1) - Please choose one
_____ is used to identify the cause of interrupt.

►**Interrupt ID Register (Page 116)**
►PC Register
►AC Register
►None of All These

Muhammad Moaaz Siddiq – MCS(4th)
Moaaz.pk@gmail.com
Campus:- Institute of E-Learning & Moderen Studies (IEMS) Samundari

Question : 4 of 10 (Marks: 1) - Please choose one
In NTFS, up to _____ characters are used to store files names,

►30
►48
►255 (Page 283)
►510

Question : 5 of 10 (Marks: 1) - Please choose one
A cluster is a collection of contiguous _____.

►Blocks (Page 242)
►Sectors
►Bytes
►None of Given

Question : 6 of 10 (Marks: 1) - Please choose one
In BPB, root directory is saved in _____.

►Cluster#0
►Cluster#1
►Cluster#2
►Cluster#3

Question : 7 of 10 (Marks: 1) - Please choose one
In NTFS, total sizes of MFT entries are _____.

►16-bytes (Page 303)
►20-bytes
►26-bytes
►32-bytes

Question : 8 of 10 (Marks: 1) - Please choose one
In NTFS, _____ store the contents of file.

►Both small & large file Record
►Small record
►Large Record
►None of given

انسان دکھ نہیں دیتے بلکہ انسانوں سے وابستہ امیدیں دکھ دیتی ہیں

Muhammad Moaaz Siddiq – MCS(4th)
Moaaz.pk@gmail.com
Campus:- Institute of E-Learning & Moderen Studies (IEMS) Samundari

Question : 9 of 10 (Marks: 1) - Please choose one
In NTFS, contents and indexed of file is stored in _____.

►Small record
►Large Record
►Both small & large file Record
►None of given

Question : 10 of 10 (Marks: 1) - Please choose one
Total No. of bytes that can be stored in Keyboard Buffer is____.

►16
►32 (Page 54)
►64
►128

Question : 1 of 10 (Marks: 1) - Please choose one
BIOS support _____UARTS as COM ports.
►6
►4 (Page 113)
►3
►2

Question : 2 of 10 (Marks: 1) - Please choose one
DCE stands for _____.

►Data communication equipment (Page 109)
►Distributed Computing Environment
►Data Communications Equipment
►Data Carrier Equipment

Question : 3 of 10 (Marks: 1) - Please choose one
In counter register bit no. 3 changes its value between 0 and 1 with in ____clock cycles

►1
►2
►4
► 16 (Page 69)

بہترین تجربہ وہ ہے جس سے نصیحت حاصل ہو

Muhammad Moaaz Siddiq – MCS(4th)
Moaaz.pk@gmail.com
Campus:- Institute of E-Learning & Moderen Studies (IEMS) Samundari

Question : 4 of 10 (Marks: 1) - Please choose one
In _____each byte is needed to be encapsulated in start and end.

►Synchronous communication
►**Asynchronous communication (Page 106)**
►Both
►None of given

Question : 5 of 10 (Marks: 1) - Please choose one
The _____ service # is not used in any interrupt.

►01
►02
►03
►**FF**

Question : 6 of 10 (Marks: 1) - Please choose one
If we want to send printing on the printer then we have to perform following steps.

►Initialize printer
►Read Status
►Check Error
►**All of the given**

Question : 7 of 10 (Marks: 1) - Please choose one
DTE is _____.

►**Data terminal equipment (Page 109)**
►Data transmitting equipment
►Dual terminal equipment
►None of the given.

Question : 8 of 10 (Marks: 1) - Please choose one
If printer is _____ then printer sends back the ACK signal to the printer interface

►**idle (Page 97)**
►busy
►Out of paper
►None of the given

خوبصورتی علم وادب سے ہوتی ہے لباس و حسن سے نہیں

Muhammad Moaaz Siddiq – MCS(4th)
Moaaz.pk@gmail.com
Campus:- Institute of E-Learning & Moderen Studies (IEMS) Samundari

Question : 9 of 10 (Marks: 1) - Please choose one
DSR stands for _____ .

►**Data set ready (Page 111)**
►Data service ready
►Data stock ready
►None of the given

Question : 10 of 10 (Marks: 1) - Please choose one
At IRQ 7 Interrupt # ___ is used.

► 0x0A
►0x0B
► 0x0C
►**0x0F (Page 95)**

Question : 1 of 10 (Marks: 1) - Please choose one
The memory addresses of COM ports remain same for all computers
►True
►**False**

Question : 2 of 10 (Marks: 1) - Please choose one
In keyboard status byte bit no. 2 and 3 are used for ctrl and alt keys respectively. which of the following condition is used to check that Ctrl + Alt keys are pressed. Where: unsigned char far * scr = (unsigned char far *)(0x00400017);

►**if (((*scr)&12)==12)**
►if (((*scr)&8)==8)
►if (((*scr)&4)==4)
►if (((*scr)&2)==2)

Question : 3 of 10 (Marks: 1) - Please choose one
In case of synchronous communication a timing signal is required to identify the start and end of a bit.

►**True (Page 105)**
►False

Question : 4 of 10 (Marks: 1) - Please choose one
The baud rate is set in accordance with the divisor value loaded within the UART internal registers base +0 and base +1.

►**TRUE (Page 114)**
►FALSE

Muhammad Moaaz Siddiq – MCS(4th)
Moaaz.pk@gmail.com
Campus:- Institute of E-Learning & Moderen Studies (IEMS) Samundari

Question : 5 of 10 (Marks: 1) - Please choose one
Software based flow control make use of -------- control characters

►Xon
►XOFF
►**Both (Page 135)**
►None

Question : 6 of 10 (Marks: 1) - Please choose one
---------- is used to read time from RTC

►**1A\02H (Page 137)**
►1A\03H
►1A\04H
►1A\05H

Question : 7 of 10 (Marks: 1) - Please choose one
Int_____ service 0 can be used to set the line parameter of the UART or COM port.

►**14H (Page 119)**
►15H
►13H
►None of the given option

Question : 8 of 10 (Marks: 1) - Please choose one
When LBA is equal to zero (0), it means _____.

►**First block of the disk (Page 240)**
►First block of the logical drive
►First block of the hidden block
►None of the given

Question : 9 of 10 (Marks: 1) - Please choose one
In IRQ2 and IRQ3 which one has the highest priority?

►Can't be determined
►Both have same priority
►IRQ3
►**IRQ2 (Page 47)**

تم اچھا کرو زمانہ تم کو برا سمجھے یہ اس سے بہتر ہے کہ تم برا کرو اور زمانہ تم کو اچھا سمجھے

Muhammad Moaaz Siddiq – MCS(4th)
Moaaz.pk@gmail.com
Campus:- Institute of E-Learning & Moderen Studies (IEMS) Samundari

Question : 10 of 10 (Marks: 1) - Please choose one
Following is not a method of I/O

►Programmed I/O
► Interrupt driven I/O
►Hardware Based I/O (Page 4)
►None of given

Question : 1 of 10 (Marks: 1) - Please choose one
It is possible to perform I/O operations from three different methods.

►True (Page 7)
►False

Question : 2 of 10 (Marks: 1) - Please choose one
The Function of I/O controller is to provide _____.

►I/O control signals
►Buffering
►Error Correction and Detection
►All of given (Page 5)

Question : 3 of 10 (Marks: 1) - Please choose one
Which of the following are types of ISR _____.

►BIOS (Basic I/O service) ISR
►DOS ISR
►ISR provided by third party device drivers
►All of the given (Page 13)

Question : 4 of 10 (Marks: 1) - Please choose one
Interrupt service number is usually placed in _____ register.

►CH
►CL
►AH (Page 26)
►AL

انسان کے لئے بری صحبت سے بڑھ کر بری کوئی چیز نہیں

Muhammad Moaaz Siddiq – MCS(4th)
Moaaz.pk@gmail.com
Campus:- Institute of E-Learning & Moderen Studies (IEMS) Samundari

Question : 5 of 10 (Marks: 1) - Please choose one
NMI Stand for

►**Non Maskable Interrupt (Page 46)**
►Non Multitude Interrupt
►Non Maskable Instruction
►None of Given

Question : 6 of 10 (Marks: 1) - Please choose one
A single interrupt controller can arbitrate among ____ different devices.
►4
► 6
►**8 (Page 47)**
►10

Question : 7 of 10 (Marks: 1) - Please choose one
The microprocessor package has many signals for data. Below are some incorrect priority order (Higher to Lower).

►**Reset,Hold,NMI,INTR (Page 46)**
►NMI, INTR,Hold,Reset
►INTR,NMI,Reset,Hold
►None of the Given

Question : 8 of 10 (Marks: 1) - Please choose one
The following command "outportb (0x61,inportb(0x61) & 0xFC);" will

►Turn on the speaker
►**Turn off the speaker (Page 75)**
►Toggle the speaker
►None of the given

Question : 9 of 10 (Marks: 1) - Please choose one
The following command "outportb (0x61,inportb(0x61) | 3);" will _____ .
►**Turn on the speaker (Page 74)**
►Turn off the speaker
►Toggle the speaker
►None of the above

Muhammad Moaaz Siddiq – MCS(4th)
Moaaz.pk@gmail.com
Campus:- Institute of E-Learning & Moderen Studies (IEMS) Samundari

Question : 10 of 10 (Marks: 1) - Please choose one
The PPI acts as an interface between the CPU and a parallel _____ .

►**I/O device (Page 83)**
►CPU
►BUS
►None of Given

Question : 1 of 10 (Marks: 1) - Please choose one
BIOS DO NOT support _____.

►LPT1
►LPT2
►LPT3
►**LPT4 (Page 91)**

Question : 2 of 10 (Marks: 1) - Please choose one
_____ bit is cleared to indicate the low nibble is being sent.

►D1
►D2
►D3
►**D4 (Page 104)**

Question : 3 of 10 (Marks: 1) - Please choose one
The bit _____ of Line control register in UART, if cleared will indicate that DLL is the data register.

►1
►3
►5
►**7 (Page 114)**

Question : 4 of 10 (Marks: 1) - Please choose one
_____ file system is used in NTFS based systems.

Contiguous Chained
Indexed
None of the given

خاموشی غصے کا بہترین علاج ہے

Muhammad Moaaz Siddiq – MCS(4th)
Moaaz.pk@gmail.com
Campus:- Institute of E-Learning & Moderen Studies (IEMS) Samundari

Question : 5 of 10 (Marks: 1) - Please choose one
Communication between keyboard and keyboard controller is _____.

►Asynchronous serial
►**Synchronous serial (P 77)**
►Parallel communication
►None of the given

Question : 6 of 10 (Marks: 1) - Please choose one
In NTFS, boot sector is stored at

►First and 6th sector
►First and Last sector
►Only at Last sector
►**Only at First sector**

Question : 7 of 10 (Marks: 1) - Please choose one
Standard PC operates in two modes in terms of memory which are

►Real mode and Extended Mode
►Base mode and Memory Mode
►None of the given
►**Real mode and protected mode (Page 6)**

Question : 8 of 10 (Marks: 1) - Please choose one
IVT is a table containing _____ byte entries each of which is a far address of an interrupt service routine.

►2
►**4 (Page 20)**
►8
►16

Question : 9 of 10 (Marks: 1) - Please choose one
Each paragraph in keep function is ____ bytes in size.

►4
►8
►**16 (Page 24)**
►32



Muhammad Moaaz Siddiq – MCS(4th)
Moaaz.pk@gmail.com
Campus:- Institute of E-Learning & Moderen Studies (IEMS) Samundari

Question : 10 of 10 (Marks: 1) - Please choose one
A software interrupt does not require EOI (End of interrupt).

►**True (Page 49)**
►False

Question : 1 of 10 (Marks: 1) - Please choose one
To store each character in keyboard buffer ____ bytes are required.
►**2 (Page 54)**
►4
►6
►8

Question : 2 of 10 (Marks: 1) - Please choose one
Interrupt _____ is empty; we can use its vector as a flag.

►9H
►13H
►15H
►**65H (Page 65)**

Question : 3 of 10 (Marks: 1) - Please choose one
Command register is an _____ bit register
►4
►**8 (Page 71)**
►16
►32

Question : 4 of 10 (Marks: 1) - Please choose one
How many bytes can be used to store a file name in NTFS?
►128
►**255**
►510
►1024

Question : 5 of 10 (Marks: 1) - Please choose one
____ is the first logical sector of NTFS partition.

►DPB
►MFT
►**Boot sector**
►None

Muhammad Moaaz Siddiq – MCS(4th)
Moaaz.pk@gmail.com
Campus:- Institute of E-Learning & Moderen Studies (IEMS) Samundari

Question : 6 of 10 (Marks: 1) - Please choose one
In boot block BIOS parameter block starts from 03H

►05H
►08H
►0BH (Page 302)

Question : 7 of 10 (Marks: 1) - Please choose one
IN NTFS, FAT and root directory is replaced by

►FCB
►MFT (Page 301)
►Hidden blocks
►Boot sector

Question : 8 of 10 (Marks: 1) - Please choose one
Block # 2 is the safest block to store the backup of boot block.

►True
►False

Question : 2 of 10 (Marks: 1) - Please choose one
The keyboards interface as discussed earlier uses the IRQ0 and the port 64H as data port.

►True
► False

Question : 3 of 10 (Marks: 1) - Please choose one
FAT12 will have 12-bit wide entries and can have 2^12=4096 entries maximum

►True
► False

Question : 4 of 10 (Marks: 1) - Please choose one
In order to produce the sound from PC internal Speaker we have to load the___bit divisor value at the ___port.

►8, 0x21
►16, 0x42
►32, 0x22
►64, 0x32



Muhammad Moaaz Siddiq – MCS(4th)
Moaaz.pk@gmail.com
Campus:- Institute of E-Learning & Moderen Studies (IEMS) Samundari

Some More MCQs and Quizzes

Question : 1 of 10 (Marks: 1) - Please choose one
DMA stands for_____

►**Direct Memory Access (Page 4)**
►Distinct Memory Access
►Direct Module Access
►Direct Memory Allocation

Question : 2 of 10 (Marks: 1) - Please choose one
UART stands for_____

►**Universal Asynchronous Receiver Transmitter (Page 107)**
►Universal Adjustment and Realigning Tool
►Unconventional Assisted Recovery Team
►None of these

Question : 3 of 10 (Marks: 1) - Please choose one
Interrupt Vector Table (IVT) in short is a _____ bytes sized table.

►**1024 (Page 10)**
►2048
►3072
►4096

Question : 4 of 10 (Marks: 1) - Please choose one
Hardware Interrupts are _____.

►Preemptive
►**Non-Preemptive (Page 48)**
►Both Preemptive and Non-Preemptive
►None of Given

Question : 5 of 10 (Marks: 1) - Please choose one
Timer interrupt is a _____.

►**Hardware Interrupt (Page 28)**
►Software Interrupt
►Both of these
►None of These

Muhammad Moaaz Siddiq – MCS(4th)
Moaaz.pk@gmail.com
Campus:- Institute of E-Learning & Moderen Studies (IEMS) Samundari

Question : 6 of 10 (Marks: 1) - Please choose one
The keyboard makes use of interrupt number _____ for its input operations.
►**9 (Page 34)**
►10
►11
►12

Question : 7 of 10 (Marks: 1) - Please choose one
Register can be used to divide frequency is _____

►**Counter Register (Page 69)**
►Accumulator Register
►None of these

Question : 8 of 10 (Marks: 1) - Please choose one
Which port is known as Data Port_____
►**60H (Page 177)**
►61H
►64H
►69H

Question : 9 of 10 (Marks: 1) - Please choose one
LPTs can be swapped.
►**True (Page 92)**
►False

Question : 10 of 10 (Marks: 1) - Please choose one
PPI is used to perform parallel communication
►**True (Page 81)**
►False

Question : 1 of 10 (Marks: 1) - Please choose one
_____is used to control the printer via the BIOS

►Int 16H
►**Int 17H (Page 84)**
►Int 18H
►Int 19H

اطمینان قلب چاہتے ہو تو حسد سے دور رہو

Muhammad Moaaz Siddiq – MCS(4th)
Moaaz.pk@gmail.com
Campus:- Institute of E-Learning & Moderen Studies (IEMS) Samundari

Question : 2 of 10 (Marks: 1) - Please choose one
There are two main types of interrupts namely _____.

►PC based and Window based
►Hardware based and Kernal based
►**Hardware interrupts and Software interrupts (Page 10)**
►None of the given

Question : 3 of 10 (Marks: 1) - Please choose one
To set the interrupt vector means is to change the double word sized interrupt vector within the IVT.

►**True (Page 22)**
►False

Question : 4 of 10 (Marks: 1) - Please choose one
The service number is usually placed in the _____ register.
►AL
►CL
►**AH (Page 26)**
►AX

Question : 5 of 10 (Marks: 1) - Please choose one
The keyboard makes use of interrupt number _____ for its input operations.
►**9 (Page 34)**
►10
►11
►12

Question : 6 of 10 (Marks: 1) - Please choose one
The service _____ is called the keyboard hook service.

►15H/2FH
►**15H/4FH (Page 44)**
►15H/FFH

Question : 7 of 10 (Marks: 1) - Please choose one
The BIOS interrupt _____ can be used to configure RTC.

►**1AH (Page 136)**
►2AH
►3AH
►4AH

Muhammad Moaaz Siddiq – MCS(4th)
Moaaz.pk@gmail.com
Campus:- Institute of E-Learning & Moderen Studies (IEMS) Samundari

Question : 8 of 10 (Marks: 1) - Please choose one
PPI stands for
►Parallel Programmable interface
►Peripheral Programmable interface (Page 76)
►Port Programmable interface
►None of the given

Question : 9 of 10 (Marks: 1) - Please choose one
Int _____ is used to control the printer via the BIOS.
►17H (Page 84)
►18H
►20H
►21H

Question : 10 of 10 (Marks: 1) - Please choose one
Counter register can be used to divide clock signal.
►True (Page 69)
►False

Question : 1 of 10 (Marks: 1) - Please choose one
The bit number _____ of the coprocessor control word is the interrupt enable flag.

►7 (Page 168)
►8
►9
►6

Question : 2 of 10 (Marks: 1) - Please choose one
There are _____ kinds of serial communication.

►2 (Page 105)
►3
►4
►5

Question : 3 of 10 (Marks: 1) - Please choose one
_____ store the base address for LPT1.

►40:00H
►40:02H
►40:08H (Page 92)
►40:1AH

Muhammad Moaaz Siddiq – MCS(4th)
Moaaz.pk@gmail.com
Campus:- Institute of E-Learning & Moderen Studies (IEMS) Samundari

Question : 4 of 10 (Marks: 1) - Please choose one
The amount of memory above conventional memory (extended memory) can be determined using the service _____.

►**15H/88H (Page 162)**
►16H/88H
►17H/88H
►21H/88H

Question : 5 of 10 (Marks: 1) - Please choose one
The output on the monitor is controlled by a controller called _____within the PC.

►**Video controller (Page 30)**
►Bus controller
►Ram controller
►None of the given

Question : 6 of 10 (Marks: 1) - Please choose one
Interrupt 9 usually reads the _____ from keyboard.

►ASCII code
►**Scan code (Page 34)**
►Both ASCII and Scan code
►None of the above

Question : 7 of 10 (Marks: 1) - Please choose one
NMI Stand for

►**Non Maskable Interrupt (Page 46)**
►Non Multitude Interrupt
►Non Maskable Instruction
►None of Given

Question : 8 of 10 (Marks: 1) - Please choose one
A single interrupt controller can arbitrate among ____ different devices.

►4
►6
►**8 (Page 47)**
►10

اس سے پہلے کہ تمہیں شہوت فتنے میں ڈالے نکاح کرلو

Muhammad Moaaz Siddiq – MCS(4th)
Moaaz.pk@gmail.com
Campus:- Institute of E-Learning & Moderen Studies (IEMS) Samundari

Question : 9 of 10 (Marks: 1) - Please choose one
The microprocessor package has many signals for data. Below are some in Correct priority order (Higher to Lower).

►**Reset,Hold,NMI,INTR (Page 46)**
►NMI, INTR,Hold,Reset
►INTR,NMI,Reset,Hold
►None of the Given

Question : 10 of 10 (Marks: 1) - Please choose one
The _____function initialize the COM port whose number is passed as parameter using BIOS services.

►Initializecom()
►**Initialize() (Page 125)**
►Recievechar()
►None of these option

Question : 1 of 10 (Marks: 1) - Please choose one
There are two type of communication synchronous and Anti Synchronous
►True
►**False (Page 105)**

Question : 2 of 10 (Marks: 1) - Please choose one
REGS is a Union

►**True**
►False

Question : 3 of 10 (Marks: 1) - Please choose one
Keyboard Status Byte is located at the address

► 0040:0000H
► 0040:0013H
► 0040:0015H
► **0040:0017H (Page 29)**

Question : 4 of 10 (Marks: 1) - Please choose one
If we use keep (0, 1000) in a TSR program, the memory allocated to it is

► 64000 bytes
► 32000 bytes
► **16000 bytes**
► 80000 bytes

Muhammad Moaaz Siddiq – MCS(4th)
Moaaz.pk@gmail.com
Campus:- Institute of E-Learning & Moderen Studies (IEMS) Samundari

Question : 5 of 10 (Marks: 1) - Please choose one
Maximum number of interrupts in a standard PC is

► 64
► 128
► **256 (Page 10)**
► 512

Question : 6 of 10 (Marks: 1) - Please choose one
The -------- function receive a byte and COM port number is passed as parameter using BIOS service

►Receivebyte ();
►Receive ();
►**Receivechar (); (Page 125)**
►None of the given option

Question : 7 of 10 (Marks: 1) - Please choose one
---------- whenever receive indicates the start of communication …….. whenever receive indicates the end of communication

►**XON\XOFF (Page 135)**
►XOFF\XON
►XON\YOFF
►YON\XOFF

Question : 8 of 10 (Marks: 1) - Please choose one
---------- is used to set time from RTC

►1A\02H
►**1A\03H (Page 138)**
►1A\04H
►1A\05H

Question : 9 of 10 (Marks: 1) - Please choose one
Set the Interrupt vector means to change the double word sized interrupt vector within IVT.
►**True (Page 22)**
►False

Question : 10 of 10 (Marks: 1) - Please choose one
If keyboard buffer is empty the head and tail points at the same location.
►**True (Page 55)**
►False

Muhammad Moaaz Siddiq – MCS(4th)
Moaaz.pk@gmail.com
Campus:- Institute of E-Learning & Moderen Studies (IEMS) Samundari

Question : 1 of 10 (Marks: 1) - Please choose one
Standard PC can have _____ PPI.
►1
►4 **(Page 84)**
►8
►16

Question : 2 of 10 (Marks: 1) - Please choose one
By cascading two DMAs ____ bits can be transferred.
►4
►8
►16 **(Page 186)**
►32

Question : 3 of 10 (Marks: 1) - Please choose one
PPI interconnection _____ bits is cleared to indicate low nibble is being sent.
►D1
►D2
► D3
► D4 **(Page 101)**

Question : 4 of 10 (Marks: 1) - Please choose one
Display device (Monitor) performs _____ I/O.

►memory mapped **(Page 30)**
►Isolated
►Both of above
►None of these

Question : 5 of 10 (Marks: 1) - Please choose one
Timer interrupt occurs _____ times every second by means of hardware.

►18.2 **(Page 28)**
►16.2
►15.2
►14.2

Question : 6 of 10 (Marks: 1) - Please choose one
An I/O device cannot be directly connected to the busses so controller is placed between CPU and I/O.
►True **(Page 83)**
►False

Muhammad Moaaz Siddiq – MCS(4th)
Moaaz.pk@gmail.com
Campus:- Institute of E-Learning & Moderen Studies (IEMS) Samundari

Question : 7 of 10 (Marks: 1) - Please choose one
Tail of keyboard should get to get the start of buffer.
►**True (Page 55)**
►False

Question : 8 of 10 (Marks: 1) - Please choose one
____ No. of bytes are used to store the character in the keyboard buffer.
► 1
►**2 (Page 54)**
►4
► 8

Question : 9 of 10 (Marks: 1) - Please choose one
We have set the bit No. 7 of IMR(Interrupt Mask Register) to unmask the Interrupt so that interrupt _____ can occur at ____ line.

►**0xf ,IRQ 7**
► 0xa, IRQ 6
► 0x8, IRQ 5
►0x6, IRQ 2

Question : 10 of 10 (Marks: 1) - Please choose one
If we want to produce the grave voice from speaker phone then we have to load the ____ divisor values at Port ____.

►high, 0x42
►low, 0x22
►high, 0x22
►**low, 0x42**

اپنی مرضی اور اللہ کی مرضی میں فرق کا نام غم ہے
وہ لوگ مبارک ہیں جو الفاظ سے نصیحت نہیں کرتے بلکہ عمل سے کرتے ہیں
ہر چیز کی ایک پہچان ہوتی ہے اور عقلمند کی پہچان غور و فکر کرنا ہے اور غور و فکر کی پہچان خاموشی ہے

Muhammad Moaaz Siddiq – MCS(4th)
Moaaz.pk@gmail.com
Campus:- Institute of E-Learning & Moderen Studies (IEMS) Samundari

CS609-System Programming
FINAL TERM MCQS
Prepared by: JUNAID MALIK

AL-JUNAID TECH INSTITUTE



www.vulmshelp.com



Language Courses Training Available

I'm providing paid courses in different languages within 3 Months,Certificate will be awarded after completion.

- HTML
- CSS
- JAVASCRIPT
- BOOTSTRAPS
- JQUERY
- PHP MYSQL
- NODES.JS
- REACT JS

LMS Handling Services

LMS Activities Paid Task

Assignments 95% Results

Quizes 95% Results

GDB 95% Results

For CS619 Project Feel Free To Contact With Me

Ph# 0304-1659294

Email: junaidfazal08@gmail.com

AL-JUNAID TECH INSTITUTE

1. The_____signals are used by various devices to request a DMA operation.

- **DREQ** **PG #186**
- REQ
- DRE
- None of the given

2.DMA works in__modes.

- 2
- **3** **PG #188**
- 5
- 7

3. DMA Request register can be used to simulate a DMA request through software in case of___.

- I/O to memory transfer
- None of the given
- **Memory to memory transfer** **PG #192**
- Memory to I/O transfer

4. _____are the circular division of the disk.

- **Tracks** **PG #201**
- Sectors
- Blocks
- None of given

5. Blocks are_____.

- **Sectors per track** **PG #202**
- Track per sector
- Cylinder per track
- None of the given

6. Last___bytes of data part in partitioning table are the partition table signature.

- **2** **PG #218**
- 4
- 6
- 8

AL-JUNAID TECH INSTITUTE
0304-1659294
www.vulmshelp.com

7. Information about further partitioned drives is available at_physical block of extended partition.

- **First** **PG #222**
- Second
- Third
- Fourth

8. _____will be used to get drive parameters.

- Int 13H/42H
- **Int 13H/48H** **PG #238**
- Int 13H/66H
- Int 13H/66H

9. Inside a boot block operating system name occupies_bytes.

- 3
- 5
- **8** **Not sure**
- 11

10. A programmer wants to access the DPB(Drive Parameter Block) for C drive using undocumented service 21H/32H, whathe should place in DL register_____?

- **0** **PG #249**
- 1
- 2
- 3

20. Blocks for FAT can be accessed using__.

- BPB
- DPB
- **FCB**
- Both BPB and DPB.

21. In main memory smallest addressable unit is__.

- **Byte**
- Nibble

0304-1659294
www.vulmshelp.com

- Word
- None of the given

22. Using the root directory entry and the___, we can access the contents of File.

- Reserved Blocks
- Number of FAT copies
- **File Allocation Table(FAT) PG # 241**
- None of the given

23. If a file is having more than one cluster then it will be managed by _____.

- FAT
- BPB
- DPB
- **None of the given**

24. Practically in FAT 32, total number of entries are___.

- 2^{26}
- 2^{28}
- 2^{30}
- **2^{32}**

25. _____keeps the back up of its boot block.

- FAT 12
- FAT 16
- **FAT 32**
- None of the given

26. NTFS can store file name in_form.

- **UNICODE**
- Both ASCII and UNICODE
- None of the given
- ASCII

0304-1659294 www.vulmshelp.com

27. FAT based file system can store file name in__form.

- ASCII
- UNICODE
- **Both ASCII and UNICODE**
- None of the given

28. The LSN of first logical sector of NTFS partition is_.

- **0** **PG # 240**
- 1
- 2
- 3

29. To minimize_____defragmenter is used.

- **Seek time** **PG # 316**
- Access time
- Rotational delay
- Both seek and access time

30. In memory map of first 1 MB of RAM, the higher__is called system memory.

- 64 KB
- **384 KB** **PG # 317**
- 640 KB
- None of the given

31. Extended memory can be accessed in___mode.

- Real
- **Protected**
- Both real and protected
- None of the given

32. Extended memory can be accessed only after installing the driver.

- EMM386.EXE
- **HIMEM.SYS** **PG # 318**
- IO.SYS
- None of the given

33. DOS uses_____memory allocation system.

0304-1659294
www.vulmshelp.com

- **Contiguous PG # 325**
- Non Contiguous
- Both Contiguous and Non Contiguous
- None of the given

34. In protected mode segment register will be called___.

- **Selector PG # 326**
- Descriptor
- IDT
- Both Selector and Descriptor

35. There are___Control registers in 80386 and above systems.

- 2
- **4 PG # 331**
- 8
- 16

36. If the CHS (cylinder, head, and sector) address of a disk is known it can be translated in to the_____address and viceversa.

- Logical
- **LBA PG # 213**
- Physical
- None of the Given

37. _____is a data structure maintained by DOS in the boot block for each drive.

- DPB
- **BPB PG # 242**
- FCB
- None of the Given

38. System Stores the DPBs (Drive Parameter Block) for different drives in the form of_____.

- Chain / link list
- Array
- **Table Not sure**
- None of Given

39. The extension of a compiled TSR (Terminate and Study Resident)

0304-1659294 www.vulmshelp.com

program should be _____

- .EXE
- **.COM**
- .CPP
- .C

44. _____is LED control byte.

- 0xF3
- **0xED PG # 181**
- 0xE5
- 0xFF

45. The DMA requests to acquire buses through the____signal.

- **OLD PG # 186**
- ACR
- ACK
- All of the given

46. _____will specify if the next DMA transfer will happen as a single transfer, block transfer or demand transfer

- DMA Request register
- DMA Mask register
- **DMA Mode register**
- DMA Command register

47. _____will specify if the channel is to cascade two DMA controllers.

- DMA Request register
- DMA Mask register
- **DMA Mode register**
- DMA Command register

48. _____are the longitudinal division of the disk.

- Tracks
- **Sectors PG # 201**
- Blocks

0304-1639294
www.vulmshelp.com

- None of given

51. Highest physical capacity of the disk

according to the IDE interface is

- 504 MB
- 80 GB
- **127 GB PG # 212**
- 300 GB

52. In partition table, first_bytes contain code which loads the boot block of active partition.

- 128
- 286
- **446 PG # 218**
- 500

53. On a single disk there can be__different Operating Systems.

- 2
- **4 PG # 219**
- 6

54. _____is physical or absolute address.

- **LBA PG # 240**
- LSN
- CHS
- None of the given

55. The_____block is the first block on disk.

- LSN =0
- **LBA=0 PG # 240**
- Both LBA=0 and LSN=0
- None of the given

56. A directory is collection of_.

- Bios Parameter Blocks
- Drive Parameter Blocks

0304-1659294
www.vulmshelp.com

- **File Control Blocks**
- None of the given

57. First cluster in user data area is numbered____in a FAT based systems.

- 0
- 1
- **2 PG # 258**
- 3

58. Blocks for FAT can be accessed using__.

- BPB
- DPB
- **FCB**
- Both BPB and DPB.

59. In FAT12 to calculate the address or offset from index, we need to multiply it with_____.

- 1/2
- **3/2 PG 267**
- 5/2
- 7/2

60. Using the root directory entry and the___, we can access the contents of File.

- Reserved Blocks
- Number of FAT copies
- **File Allocation Table(FAT)PG #269**
- None of the given

61. In FAT16 if FAT entry is between FFF0H to FFF6H then_.

- Cluster is available
- **It is a Reserved cluster**
- It is next file cluster
- It is a last file cluster

62. _____needed to store a UNICODE character.

0304-1659294
www.vulmshelp.com

- Nibble
- **2 Bytes**
- 4 Bytes
- Byte

63. _____used for FCB in FAT 12 and FAT 16.

- Nibble
- Byte
- **2 Bytes**
- 4 Bytes

64. _____file system is used in CD's.

- None of the given
- **Contiguous**
- Chained
- Indexed

65. FAT based file system can store file name in__form.

- ASCII
- UNICODE
- **Both ASCII and UNICODE**
- None of the given

66. In case of_____contents, the indexes of clusters of file are stored in MFT.

- Small files
- Large files
- **Both small and large files**
- None of the given

67. PC's operated on 20 bit address bus can operate in__mode.

- **Real** **PG # 319**
- Protected
- Both real and protected
- None of the given

68. Int 21H/52H returns the address of DOS internal data structures in ES: BX, the far address of the first MCB in memory lies

0304-1659294
www.vulmshelp.com

_____behind the address returned.

- 2-bytes
- **4-bytes**
- 6-bytes
- 8-bytes

69. In memory management, descriptor entry is of bytes.

- **64**
- 32
- 8
- 16

70. There are___Control registers in 80386 and above systems.

- 2
- **4 PG # 331**
- 8
- 16

71. Control Registers are used for conveying certain control information for_____Addressing and Co-Processors.

- Real Mode
- **Protected Mode PG # 331**
- Virtual Mode
- None of the given

72. The least significant bit of Control register___is PE-bit which can be set to enable Protected Mode Addressing.

- **CR0**
- CR1
- CR2
- CR3

73. CD.. command gives the of directory.

- None of the given
- One previous level
- One next level
- **Current path PG #282**

AL-JUNAID TECH INSTITUTE 0345 4059294 www.vulmshelp.com

74. Cluster size is variable and can be measured in power of__.

- **2**
- 16
- 4
- 8

75. Who notifies the end of interrupt (EOI) to the programmable interrupt controller?

- Programmer
- **Microprocessor**
- Operating system
- None of the given

76. Choose the correct statement to notify the interrupt controller about end of interrupt (EOI).

- Inportb(0x20,0x20)
- **Outportb(0x20,0x20)**
- Outportb(0x30,0x22)
- None of the given

77. Information about Pending interrupts are held by___.

- Interrupt Service Register
- Interrupt Mask Register
- None of the Given
- **Interrupt Request Register**

78. BPB stands for____.

- **BIOS parameter block**
- BIOS processing block
- Base processing block
- BIOS partition block

79. The input frequency used by the interval timer is the signal generated by the clock generator.

- None of Given
- **PCLK**
- ACL

0304-1659294 www.vulmshelp.com

- ICP

80. BIOS DO NOT support____.

- LPT1
- LPT2
- LPT3
- **LPT4**

81. DTE is_____.

- **Data terminal equipment PG #109**
- Data transmitting equipment
- Dual terminal equipment
- None of the given.

82. Int 14H_____can be used to send a byte.

- Service # 0
- **Service # 1**
- Service # 2
- Service # 3

83. A single DMA can transfer_operands to and from memory in a single a bus cycle.

- **8 bit**
- 16 bit
- 32 bit
- 64 bbit

85. DMA does not work in_mode.

- **Double Mode**
- Block Transfer
- Demand Transfer
- None of the given

86.

An addressable unit on disk can be addressed by three parameters____

- **head #, sector # and track #**
- HD #, AH# and track #
- Hd0 #, sector # and track #

AL-JUNAID TECH INSTITUTE 03041659294 www.vulmshelp.com

- None of the given

87.

Each addressable unit has a unique combination of sec#, head# and track# as its____address.

- Both Physical and Logical
- None of the given
- **Physical**
- Logical

88. In partition table, first___bytes contain code which loads the boot block of active partition.

- 128
- 286
- **446**
- 500

89.

On a single disk there can be____different Operating Systems.

- 2
- **4**
- 6
- 8

90.

The last partition table within the chain contains_entry/entries about the logical drive.

- **Single**
- Double
- Three
- Four

91.

Partition Table can be read using the extended__Services.

- 13H

AL-JUNAID TECH INSTITUTE 0304-1659294 www.vulmshelp.com

- **14H**
- 15H
- 17H

92.

File control block (FCB) is_____byte long.

- 8
- 16
- **32**
- 64

93. __is collection of contagious blocks.

- **Cluster**
- Sector
- File
- Directory

94.A FAT 12 table can have maximum___values.

- 1024
- 2048
- **4096**
- None of given

95.

The first cluster number of file can be found in_____.

- BPB
- DPB
- **FCB** **PG # 265**
- None of the given

96.

In FAT12 if FAT entry is 000H then.

- It is a last file cluster
- Cluster is available
- **It is a Reserved cluster**
- It is next file cluster

0304-1659294
www.vulmshelp.com

97.

Total number of clusters of FAT12 are____.

- FF0 H
- FFF H
- FEF H
- **FEE H PG # 266**

98.

When we mark a file as deleted by placing 0xE5 then the chain of clusters in FAT is also replaced by___

- E5
- **0**
- 1
- N

99. keeps the back up of its boot block.

- FAT 12
- FAT 16
- **FAT 32**
- None of the given

100. Up tobytes can be used to store a file name in NTFS.

- 30
- 126
- **254**
- 510

101.

First logical sector of NTFS partition is_.

- DPB
- MFT
- **Boot sector**
- None of the given

102. In case of___the contents of file are stored in MFT.

0304-1659294
www.vulmshelp.com

- Small files
- Large files
- **Both small and large files**
- None of the given

103.

Interrupt Descriptor Table can have up to___entries.

- 64
- 128
- **256**
- 512

104. There can be__different descriptors privilege levels.

- 2
- **4**
- 8
- 10

105. In NTFS, Block # is the safest block to store the back up of boot block.

- 8
- **10** **Not sure**
- 20
- 6

106.

Size of single entry in Partition Table is____

- **512 bytes**
- 128 bytes
- 64 bytes
- 16 bytes

107.

MBR partition table can have_entries at maximum

- 1

0304-1659294
www.vulmshelp.com

- 2
- 3
- **4 Not sure**

108. __is used to identify the cause of interrupt.

- Interrupt Enable register
- **Interrupt ID register**
- Interrupt Status register
- None of the given

109.

To access bettery powered RAM, only_ports are important from programming point of view.

- **70 and 71H PG # 141**
- 71 and 72H
- 70 and 72H
- 72 and 73H

110.

The DMA requests to acquire buses through the___signal.

- **HOLD PG # 186**
- ACR
- ACK
- All of the given

111. ____used to program various common parameters of transfer for all the channels.

- None of the above
- DMA Status Register
- **DMA Command Register PG # 191**
- DMA Request Register

112. Bit #___of mode register in DMA determine the direction of a transfer.

- **2**
- 3

0304-1659294 www.vulmshelp.com

- 4
- 5

113.

Highest capacity of disk can be accessed using BIOS functions is____.

- 128 MB
- 256 MB
- **504 MB PG # 211**
- 127 GB

114.

DOS has built in limit of___blocks per cluster.

- 32
- 64
- **128 PG # 242**
- 256

115.

BIOS Parameter block is situated in_Block.

- **Boot PG # 242**
- Data
- Extended Data
- None of Given

116. The____block is the first block on disk.

- LSN =0
- **LBA=0 PG # 240**
- Both LBA=0 and LSN=0
- None of the given

117. Inside a boot block jump code part occupies_____byte.

- 5
- 8
- 11
- **3 PG # 244**

0304-1659294 www.vulmshelp.com

118.

In boot block BIOS parameter block starts from___.

- 03H
- 05H
- 08H
- **0BH**

119.

Total number of cluster of FAT16 is_

- FFF0 H
- FFFF H
- FFEF H
- **FFEE H Page 272**

120. Total____fragments can be supported for storing long file names.

- 26
- 28
- **32**
- 48

121. MCB is a bytes data structure.

- 8
- **16 PG # 321**
- 32
- 64

122.

Real mode does not support___memory allocation system.

- **Contiguous**
- Non Contiguous
- Both Contiguous and Non Contiguous
- None of the given

123.

0304-1659294
www.vulmshelp.com

For virus to propagate itself, it has to intercept interrupt__.

- 9H
- 11H
- **13H**
- 17H

124.

If virus wants to be in memory independently, it should have its own_.

- MCB
- PSP
- EB
- **Both MCB and PSP PG #334**

125. Bit #__of Eflag is used for alignment check

- 12
- 14
- 15
- **18 PG # 164**

126. First cluster in user data is numbered in a FAT based system.

- 0
- 1
- **2 PG # 258**
- 3

127. ----------BIOS services understand --.

- **LBA PG # 212**
- LSN
- Cluster #
- None

128. The first cluster number of a file can be found in---------

- BPB
- DPB
- **FCB PG # 265**
- None

0304-1659294
www.vulmshelp.com

129. The size of FS Info block is

- 64byte
- 128 byte
- 256 byte
- **512 byte PG # 300**

130. ----------In NTFS first entries are reserved.

- 4
- 6
- **16 PG # 303**
- 32

131. --------- The of boot block constitutes of BPB.

- Code part
- **Data part PG # 242**
- Both
- None

132. ----------Extended BIOS function make use of---------- address

- **LBA PG # 212**
- CHS
- LSN
- None

133. LBA address can be used in place of the CHS address.

- **True PG # 235**
- False

134. In FAT12, the maximum range of clusters is ……...

- 0 ~ FEFH
- 1~ FEFH

0304-1659294
www.vumshelp.com

- **2 ~ FEFH PG # 266**
- 3 ~ FEFH

135. NTFS volume can be accessed directly in DOS.

- True
- **False PG # 310**

136. Each partition information chunk is 16 bytes long and the last two bytes at the end of the partition tabledata part is the partition table signature whose value should be_____indicating that the code part contains valid executable code.

- 00AA
- 0055
- 050A
- **AA55 PG # 219**

137. ervice 21H/52H service returns the address of DOS internal data structures in ES: BX_____behindthe address returned lies the far address of the first MCB in memory.

- 2-bytes
- **4-bytes PG # 322**
- 6-bytes
- 8-bytes

138. 80386 can have_____control registers.

- 2
- 5
- 3
- **4 PG # 331**

139. The partition table uses the extended__service.

- **13H PG # 234**
- 14H

0304 1659294 www.vulmshelp.com

- 15H
- 16H

140. The entry point of execution in EXE File can be

- Start of the first instruction
- Start of the last instruction
- **Anywhere in the Program PG # 335**
- Can be in the middle of the program

141. Using the_entry and the FAT we can access the contents of file.

- Reserved blocks
- **Root Directory PG # 269**
- Number of FAT copies
- None of the given

142. Control information in files is maintained using

- BPB
- DPB
- **FCB PG # 256**
- FPB

143. What will happen if NTFS volume is accessed in DOS?

- Convert it to FAT volume
- Nothing will happen
- **Error of invalid media PG # 310**
- None of the given

144. LSN of FS Info block is available at

- BPB
- **FAT**
- Root Directory
- None of the given

145. DOS device drivers do not understand the_____data structures.

- FAT12

0304-1659294 www.vulmshelp.com

- FAT16
- FAT32
- **NTFS PG # 310**

146. __is a collection of contagious blocks.

- **Cluster PG # 242**
- Sector
- Byte
- None of Given

147. __used to determine the amount of conventional memory interfaced with the processor in kilobytes.

- INT 10 H
- INT 11 H
- **INT 12 H PG # 162**
- INT 13 H

148. Bit number_____of coprocessor control word is the Interrupt Enable Flag.

- **7 PG #168**
- 8
- 9
- 10

149. To distinguish 486 with Pentium CPUID Test is used.

- **True PG # 166**
- False

150. Practically____entries are there in FAT 32.

- 2^26
- 2^28
- 2^30
- **2^32 PG # 265**

151. BPB stands for____.

0304-1659294 www.vulmshelp.com

- **BIOS parameter block PG # 243**
- BIOS processing block
- Base processing block
- BIOS partition block

152. The keyboard input character scan code is received at_____port.

- **60H PG # 179**
- 61H
- 62H
- 63H

153. ___is LED control byte.

- 0xFD
- **0xED PG #181**
- 0xFF
- 0xEE

154. ___means typematic rate will be sent in next byte.

- **0xF3 PG # 180**
- 0xF4
- 0xF5
- 0xF6

155. Keyboard uses port_as status port.

- **64H PG # 177**
- 66H
- 67H
- 69H

156. The keyboard can perform___serial I/O.

- asynchronous
- **synchronous**
- Multiple

© 0304-1659294 www.vulmshelp.com

- Single

157. Bit number 2 of port 64H Status register used for output buffer full.

- True
- **False**

158. Bit number__can declares the parity error of port 64H Status register.

- 4
- 5
- 6
- **7**

159. Bit number__of port 64H Status register used for input buffer full.

- **0**
- 1
- 2
- 3

160. Disadvantage of FAT32 is__.

- Large disk size can be managed in FAT32
- Cluster size is reduced
- Internal fragmentation is reduced
- **Very large table PG # 299**

161. Maximum possible entries in FAT12 are_____.

- 1024
- 2048
- **4096 PG # 264**
- 65536

162. What will be the value of the word located at 1Fh in DPB when number of free clusters on drive is notknown?

- 0000H

0304-1659294 www.vulmshelp.com

- 1111H
- **FFFFH PG # 250**
- None of the given.

163. Jump code part contains____bytes in boot block.

- **3 PG # 302**
- 5
- 8
- 11

164. Operating system name contains_bytes in boot block.

- 3
- 5
- **8 PG # 257**
- 11

165. File can be_viewed as organization of data.

- Physically
- **Logically PG # 256**
- Both logically and physically
- None of the give

166. _is used to read a block against its LSN.

- **absread() PG # 247**
- abswrite()
- lsnread()
- None of the given

167. File can be__viewed as collection of clusters or blocks.

- **Physically PG # 256**
- Logically
- Both physically and logically

0304-1659294 www.vulmshelp.com

- None

168. When we talk about FAT based file system, in user data area first cluster number is.

- 0
- 1
- **2 PG # 258**
- None of the given

169. Cluster number can also be referred as block number.

- True
- **False PG # 258**

170. To access the block within cluster using BIOS services the cluster number should be converted into__.

- CHS
- LBA
- **LSN PG # 258**
- None of the given

171. What will be the value of DL register when we are accessing C drive using undocumented service21H/32H?

- 0
- 1
- 2
- **3 PG # 249**

176. The directory structure of DOS is like_.

- Array
- **Tree PG # 256**
- Linked list
- None of the given

177. Control information about files is maintained using__.

- BPB
- DPB
- **FCB PG # 256**

0304-1659294 www.vulmshelp.com

- FPB

178. When LSN is equal to zero (0), it means_.
- First block of the disk
- **First block of the logical drive PG # 240**
- First block of hidden blocks
- None of the given

179. In FAT32, lower___ bits are used.
- 26
- **28 PG # 292**
- 30
- 32

180. _____is relative address with respect to the start of Logical Drive.
- LBA
- **LSN PG #240**
- CHS
- None of the given

181. To access drive parameter block we use undocumented service _
- 09H/32H
- 11H/32H
- 17H/32H
- **21H/32H PG # 249**

182. _____is an absolute address relative to the start of physical drive.
- **LBA PG # 240**
- LSN
- CHS
- None of the above

183. Boot block consists of__bytes.
- 64
- 128
- 256

AL-JUNAID TECH INSTITUTE
www.vulmshelp.com

512 PG # 242

184. The DMA requests to acquire buses through the _____signal.

- **HOLD PG #186**
- ACR
- ACK
- None of Given

185. The keyboard device writes a code 0xFA on the port 60H to indicate that the_____.

- Input buffer is full
- **Byte has been received properly PG # 179**
- Output buffer is full
- None of the given

186. A single DMA can transfer_operands to and from memory in a single a bus cycle.

- **8-bits PG # 186**
- 16-bits
- 32-bits
- 12-bits

187. In FAT12, to calculate the address or offset from index, we need to multiply it with_____.

- 1/2
- **3/2 PG #267**
- 5/7
- 7/2

188. _____Register can be used to show that the channel is single transfer, block transfer or demandtransfer mode.

- DMA Command register
- DMA Request Register

0304-1659294 www.vulmshelp.com

- DMA Mode Register
- **DMA controller Register PG #187-188**

189. When we mark a file as deleted by placing 0xE5 then the chain of clusters in FAT is also replaced by____
   - E5
   - 1
   - **0 PG # 79**
   - N

190. Cluster size is reduced in_____.
   - FAT12
   - FAT16
   - **FAT32**
   - None of the given

191. In FAT32_____root directory entries are there.
   - 128
   - 256
   - 512
   - **None of the given**

192. If a file is having more than one cluster then it will be managed by
   - FAT
   - BPB
   - DPB
   - **None of the above**

193. Internal fragmentation is reduced in_____.
   - FAT12
   - **FAT16**
   - FAT32
   - None of the given

194. For supporting long file names,___fragments can be supported.
   - 12

AL-JUNAID TECH INSTITUTE 0304-1659294 www.vulmshelp.com

- 20
- 26
- **32**

195. To store a cluster in FAT 32 is/are needed.
- Nibble
- Byte
- 2 Bytes
- **4 Bytes**

196. If a file size is 12K and the size of the cluster is 4K then__clusters are used for the file.
- 2
- **3**
- 4
- 5

197. We can access the contents of File by using the root directory entry and_____.
- Reserved Blocks
- Number of FAT copies
- **File Allocation Table (FAT) PG # 269**
- None of the given

198. FAT based file system can store file name inform.
- ASCII
- UNICODE
- **Both ASCII and UNICODE**
- None of the given

199. Drive parameter block is derived from___.
- FCB
- FAT
- **BPB PG # 249**

0304-1639294
www.vulmshelp.com

- CPB

200. We can access Blocks for FAT using__.

- BPB
- DPB
- **FCB**
- Both BPB and DPB

201. If we know the cluster number, we can access the blocks within the cluster using BIOS services directly.

- **True PG # 258**
- False

202. _____is an internal data structure of DOS and resides in main memory.

- BPB
- DPB
- CPB
- **None of the given.**

203. The size of DPB data structure is_bytes.

- 16
- 32
- 64
- **128**

204. The size of FCB data structure is_bytes.

- **16**
- 32
- 64
- 128

205. Advantages of FAT32 is/are___.

0304-1659294
www.vulmshelp.com

- Large disk size can be managed in FAT32
- **Cluster size is reduced**
- Internal fragmentation is reduced
- All of the given

206. file system keeps the backup of its boot block.
- FAT12
- FAT16
- **FAT32**
- None of the given

207. To store a UNICODE character___is/are needed.
- Nibble
- Byte
- **2 Bytes**
- 4 Bytes

208. _____is the first block on disk.
- LSN =0
- **LBA=0 PG # 240**
- LBA=1
- Both LBA=0 and LSN=0

209. If FAT entry is between FFF0H to FFF6H in FAT16 then_.
- Cluster is available
- **It is a Reserved cluster PG # 272**
- It is next file cluster
- It is a last file cluster

210. File system used in CD's is__file system
- **Contiguous**
- Chained
- Indexed
- None

0304-1659294 www.vulmshelp.com

249.A file has 2 clusters and the size of cluster is 4K. What will be the size of file?

- 2K
- **8K**
- 16K
- 32K

250.In NTFS, up to____ characters are used to store files names,

- 30
- 48
- **255 PG # 283**
- 510

251.A cluster is a collection of contiguous__.

- **Blocks PG # 242**
- Sectors
- Bytes
- None of Given

252.In BPB, root directory is saved in__.

a. **Cluster#0**
b. Cluster#1
c. Cluster#2
d. Cluster#3

253.

254. In NTFS, total sizes of MFT entries are_____.

a. **16-bytes PG # 303**
b. 20-bytes
c. 26-bytes
d. 32-bytes

0304-1659294
www.vulmshelp.com

255. In NTFS,___store the contents of file.

a. Both small & large file Record

b. Small record

c. Large Record

d. None of given

256. In NTFS, contents and indexed of file is stored in_____.

a. Small record

b. Large Record

c. Both small & large file Record

d. None of given

257. Total No. of bytes that can be stored in Keyboard Buffer is_____.

a. 16

b. 32 PG #54

c. 64

d. 128

258. BIOS support__UARTS as COM ports.

a. 6

b. 4 PG #113

c. 3

d. 2

259. DCE stands for_____.

a. Data communication equipment PG # 109

b. Distributed Computing Environment

0304-1659294
www.vulmshelp.com

c. Data Communications Equipment

d. Data Carrier Equipment

260. In counter register bit no. 3 changes its value between 0 and 1 with in__clock cycles

a. 1

b. 2

c. 4

d. 16 PG #69

261. In_each byte is needed to be encapsulated in start and end.

a. Synchronous communication

b. Asynchronous communication PG # 106

c. Both

d. None of given

262. The__service # is not used in any interrupt.

a. 01

b. 02

c. 03

d. FF

263. If we want to send printing on the printer then we have to perform following steps.

a. Initialize printer

b. Read Status

c. Check Error

d. All of the given

264. If printer is__then printer sends back the ACK signal to the printer interface

0304-1659294 www.ulmshelp.com

a. idle PG # 97

b. busy

c. Out of paper

d. None of the given

265. DSR stands for_____.

a. Data set ready PG # 111

b. Data service ready

c. Data stock ready

d. None of the given

266. At IRQ 7 Interrupt #_____is used.

a. 0x0A

b. 0x0B

c. 0x0C

d. 0x0FPG # 95

267. The memory addresses of COM ports remain same for all computers

a. True

b. False

268. In keyboard status byte bit no. 2 and 3 are used for ctrl and alt keys respectively. which of the following condition is used to check that Ctrl + Alt keys are pressed. Where: unsigned char far * scr = (unsigned charfar *)(0x00400017);

a. if (((*scr)&12)==12)

b. if (((*scr)&8)==8)

0304-1659294
www.aljunaidtech.com

c. if (((*scr)&4)==4)

d. if (((*scr)&2)==2)

269. In case of synchronous communication a timing signal is required to identify the start and end of a bit.

a. True PG # 105

b. False

270. The baud rate is set in accordance with the divisor value loaded within the UART internal registers base +0 and base +1.

a. True PG # 114

b. False

271. ----------Software based flow control make use of ----------control characters

a. Xon

b. XOFF

c. Both PG # 135

d. None

272. __is used to read time from RTC

a. 1A\02H PG # 137

b. 1A\03H

c. 1A\04H

d. 1A\05H

0304-1659294
www.aimshelp.com

273. Int____service 0 can be used to set the line parameter of the UART or COM port.

a. 14H PG # 119

b. 15H

c. 13H

d. None of the given option

274. When LBA is equal to zero (0), it means_____.

a. First block of the disk PG # 240

b. First block of the logical drive

c. First block of the hidden block

d. None of the given

275. How many bytes can be used to store a file name in NTFS?

a. 128

b. 255

c. 510

d. 1024

276. In boot block BIOS parameter block starts from 03H

a. 05H

b. 08H

c. 0BH PG # 302

277. IN NTFS, FAT and root directory is replaced by

a. FCB

b. MFT PG # 301

c. Hidden blocks

0304-1659294 www.vulmshelp.com

d. Boot sector

278. Block # 2 is the safest block to store the backup of boot block.

a. True

b. False

279. The keyboards interface as discussed earlier uses the IRQ0 and the port 64H as data port.

a. True

b. False

280. FAT12 will have 12-bit wide entries and can have 2^12=4096 entries maximum

a. True

b. False

281.There is/are ____ DMA controller's masking register(s).

- **3**
- **2**
- **4**
- **1 My point of view (Not sure)**

282.An extended partition can have one or more _____ partitions.

- **Logical**
- **Physical**
- **None of Given**
- **Primary**

283.Blocks can be termed as_____.

- **Sector per Track My Point of View (Not SURE)**
- **Cylinder per Track**
- **Track per Sector**
- **None of Given**

284.First _____ bytes of partition table contain code which loads the boot block.

- **64**
- **512**
- **446**

AL-JUNAID TECH INSTITUTE 0304-1659294 www.vulmshelp.com

- 66

285.In NTFS, Backup of boot block is stored at block # _____.

- 8
- 2
- 10
- 6

286.A cluster is a collection of_____Block

- Isolated
- Contiguous
- Random
- Heterogeneous

287.DOS has a Built-in Limit of____

- 256 Block per cluster
- 128 Block per cluster
- 32 Block per cluster
- 64 Block per cluster

288.The user data area in BIOS Parameter Block starts from the cluster number _____ .

- 2
- 3
- 0 My point of View (Not Sure)
- 1

289.To measure the size of each cluster, power of _____ can be used.

- 4
- 16
- 2
- 8

290.The value _____ of partition table signature indicate that the code part contains valid executable code.

- AA44
- AA55
- AA22
- AA33

291.Maximum allowable cluster data size in FAT32 file system is

0304-1659294 www.vulmanhelp.com

_____ sectors per cluster.

- 32
- 16
- 64
- 4

292. Information about _____ partition is kept in the first physical block.

- Extended
- Primary
- Both A and B
- Nether A and B

293. LBA address can be used in place of the _____address.

- Permanent
- CHS
- Primary
- Temporary

294. The maximum length of file name in Windows operating system can be:

- 63 Character
- 255 Character OR 260 Character Both or Correct
- 31 Character
- 512 Character

AL-JUNAID TECH INSTITUTE
0304-1659294
www.vulmshelp.com

CS609-System Programming
FINAL TERM MCQS
Prepared by: JUNAID MALIK

AL-JUNAID TECH INSTITUTE



www.vulmshelp.com



Language Courses Training Available

I'm providing paid courses in different languages within 3 Months,Certificate will be awarded after completion.

- HTML
- CSS
- JAVASCRIPT
- BOOTSTRAPS
- JQUERY
- PHP MYSQL
- NODES.JS
- REACT JS

LMS Handling Services

LMS Activities Paid Task

Assignments 95% Results

Quizes 95% Results

GDB 95% Results

For CS619 Project Feel Free To Contact With Me

Ph# 0304-1659294

Email: junaidfazal08@gmail.com

AL-JUNAID TECH INSTITUTE

1. The_____signals are used by various devices to request a DMA operation.

- **DREQ** **PG #186**
- REQ
- DRE
- None of the given

2.DMA works in__modes.

- 2
- **3** **PG #188**
- 5
- 7

3. DMA Request register can be used to simulate a DMA request through software in case of___.

- I/O to memory transfer
- None of the given
- **Memory to memory transfer** **PG #192**
- Memory to I/O transfer

4. _____are the circular division of the disk.

- **Tracks** **PG #201**
- Sectors
- Blocks
- None of given

5. Blocks are_____.

- **Sectors per track** **PG #202**
- Track per sector
- Cylinder per track
- None of the given

6. Last___bytes of data part in partitioning table are the partition table signature.

- **2** **PG #218**
- 4
- 6
- 8

AL-JUNAID TECH INSTITUTE
0304-1659294
www.vulmshelp.com

7. Information about further partitioned drives is available at_physical block of extended partition.

- **First** **PG #222**
- Second
- Third
- Fourth

8. _____will be used to get drive parameters.

- Int 13H/42H
- **Int 13H/48H** **PG #238**
- Int 13H/66H
- Int 13H/66H

9. Inside a boot block operating system name occupies_bytes.

- 3
- 5
- **8** **Not sure**
- 11

10. A programmer wants to access the DPB(Drive Parameter Block) for C drive using undocumented service 21H/32H, whathe should place in DL register_____?

- **0** **PG #249**
- 1
- 2
- 3

20. Blocks for FAT can be accessed using__.

- BPB
- DPB
- **FCB**
- Both BPB and DPB.

21. In main memory smallest addressable unit is__.

- **Byte**
- Nibble

0304-1659294
www.vulmshelp.com

- Word
- None of the given

22. Using the root directory entry and the___, we can access the contents of File.

- Reserved Blocks
- Number of FAT copies
- **File Allocation Table(FAT) PG # 241**
- None of the given

23. If a file is having more than one cluster then it will be managed by _____.

- FAT
- BPB
- DPB
- **None of the given**

24. Practically in FAT 32, total number of entries are___.

- 2^{26}
- 2^{28}
- 2^{30}
- **2^{32}**

25. _____keeps the back up of its boot block.

- FAT 12
- FAT 16
- **FAT 32**
- None of the given

26. NTFS can store file name in_form.

- **UNICODE**
- Both ASCII and UNICODE
- None of the given
- ASCII

0304-1659294 www.vulmshelp.com

27. FAT based file system can store file name in__form.

- ASCII
- UNICODE
- **Both ASCII and UNICODE**
- None of the given

28. The LSN of first logical sector of NTFS partition is_.

- **0** **PG # 240**
- 1
- 2
- 3

29. To minimize_____defragmenter is used.

- **Seek time** **PG # 316**
- Access time
- Rotational delay
- Both seek and access time

30. In memory map of first 1 MB of RAM, the higher__is called system memory.

- 64 KB
- **384 KB** **PG # 317**
- 640 KB
- None of the given

31. Extended memory can be accessed in___mode.

- Real
- **Protected**
- Both real and protected
- None of the given

32. Extended memory can be accessed only after installing the driver.

- EMM386.EXE
- **HIMEM.SYS** **PG # 318**
- IO.SYS
- None of the given

33. DOS uses_____memory allocation system.

0304-1659294
www.vulmshelp.com

- **Contiguous PG # 325**
- Non Contiguous
- Both Contiguous and Non Contiguous
- None of the given

34. In protected mode segment register will be called___.

- **Selector PG # 326**
- Descriptor
- IDT
- Both Selector and Descriptor

35. There are___Control registers in 80386 and above systems.

- 2
- **4 PG # 331**
- 8
- 16

36. If the CHS (cylinder, head, and sector) address of a disk is known it can be translated in to the_____address and viceversa.

- Logical
- **LBA PG # 213**
- Physical
- None of the Given

37. _____is a data structure maintained by DOS in the boot block for each drive.

- DPB
- **BPB PG # 242**
- FCB
- None of the Given

38. System Stores the DPBs (Drive Parameter Block) for different drives in the form of_____.

- Chain / link list
- Array
- **Table Not sure**
- None of Given

39. The extension of a compiled TSR (Terminate and Study Resident)

0304-1659294 www.vulmshelp.com

program should be _____

- .EXE
- **.COM**
- .CPP
- .C

44. _____is LED control byte.

- 0xF3
- **0xED PG # 181**
- 0xE5
- 0xFF

45. The DMA requests to acquire buses through the____signal.

- **OLD PG # 186**
- ACR
- ACK
- All of the given

46. _____will specify if the next DMA transfer will happen as a single transfer, block transfer or demand transfer

- DMA Request register
- DMA Mask register
- **DMA Mode register**
- DMA Command register

47. _____will specify if the channel is to cascade two DMA controllers.

- DMA Request register
- DMA Mask register
- **DMA Mode register**
- DMA Command register

48. _____are the longitudinal division of the disk.

- Tracks
- **Sectors PG # 201**
- Blocks

0304-1639294
www.vulmshelp.com

- None of given

51. Highest physical capacity of the disk

according to the IDE interface is

- 504 MB
- 80 GB
- **127 GB PG # 212**
- 300 GB

52. In partition table, first_bytes contain code which loads the boot block of active partition.

- 128
- 286
- **446 PG # 218**
- 500

53. On a single disk there can be__different Operating Systems.

- 2
- **4 PG # 219**
- 6

54. _____is physical or absolute address.

- **LBA PG # 240**
- LSN
- CHS
- None of the given

55. The_____block is the first block on disk.

- LSN =0
- **LBA=0 PG # 240**
- Both LBA=0 and LSN=0
- None of the given

56. A directory is collection of_.

- Bios Parameter Blocks
- Drive Parameter Blocks

0304-1659294 www.yuknahelp.com

- **File Control Blocks**
- None of the given

57. First cluster in user data area is numbered____in a FAT based systems.

- 0
- 1
- **2 PG # 258**
- 3

58. Blocks for FAT can be accessed using__.

- BPB
- DPB
- **FCB**
- Both BPB and DPB.

59. In FAT12 to calculate the address or offset from index, we need to multiply it with_____.

- 1/2
- **3/2 PG 267**
- 5/2
- 7/2

60. Using the root directory entry and the___, we can access the contents of File.

- Reserved Blocks
- Number of FAT copies
- **File Allocation Table(FAT)PG #269**
- None of the given

61. In FAT16 if FAT entry is between FFF0H to FFF6H then_.

- Cluster is available
- **It is a Reserved cluster**
- It is next file cluster
- It is a last file cluster

62. _____needed to store a UNICODE character.

0304-1659294
www.vulmshelp.com

- Nibble
- **2 Bytes**
- 4 Bytes
- Byte

63. _____used for FCB in FAT 12 and FAT 16.

- Nibble
- Byte
- **2 Bytes**
- 4 Bytes

64. _____file system is used in CD's.

- None of the given
- **Contiguous**
- Chained
- Indexed

65. FAT based file system can store file name in__form.

- ASCII
- UNICODE
- **Both ASCII and UNICODE**
- None of the given

66. In case of_____contents, the indexes of clusters of file are stored in MFT.

- Small files
- Large files
- **Both small and large files**
- None of the given

67. PC's operated on 20 bit address bus can operate in__mode.

- **Real** **PG # 319**
- Protected
- Both real and protected
- None of the given

68. Int 21H/52H returns the address of DOS internal data structures in ES: BX, the far address of the first MCB in memory lies

0304-1659294
www.vulmshelp.com

_____behind the address returned.

- 2-bytes
- **4-bytes**
- 6-bytes
- 8-bytes

69. In memory management, descriptor entry is of bytes.

- **64**
- 32
- 8
- 16

70. There are___Control registers in 80386 and above systems.

- 2
- **4 PG # 331**
- 8
- 16

71. Control Registers are used for conveying certain control information for_____Addressing and Co-Processors.

- Real Mode
- **Protected Mode PG # 331**
- Virtual Mode
- None of the given

72. The least significant bit of Control register___is PE-bit which can be set to enable Protected Mode Addressing.

- **CR0**
- CR1
- CR2
- CR3

73. CD.. command gives the of directory.

- None of the given
- One previous level
- One next level
- **Current path PG #282**

AL-JUNAID TECH INSTITUTE www.vulmshelp.com

74. Cluster size is variable and can be measured in power of__.

- **2**
- 16
- 4
- 8

75. Who notifies the end of interrupt (EOI) to the programmable interrupt controller?

- Programmer
- **Microprocessor**
- Operating system
- None of the given

76. Choose the correct statement to notify the interrupt controller about end of interrupt (EOI).

- Inportb(0x20,0x20)
- **Outportb(0x20,0x20)**
- Outportb(0x30,0x22)
- None of the given

77. Information about Pending interrupts are held by___.

- Interrupt Service Register
- Interrupt Mask Register
- None of the Given
- **Interrupt Request Register**

78. BPB stands for____.

- **BIOS parameter block**
- BIOS processing block
- Base processing block
- BIOS partition block

79. The input frequency used by the interval timer is the signal generated by the clock generator.

- None of Given
- **PCLK**
- ACL

0304-1659294
www.vulmshelp.com

- ICP

80. BIOS DO NOT support____.

- LPT1
- LPT2
- LPT3
- **LPT4**

81. DTE is_____.

- **Data terminal equipment PG #109**
- Data transmitting equipment
- Dual terminal equipment
- None of the given.

82. Int 14H_____can be used to send a byte.

- Service # 0
- **Service # 1**
- Service # 2
- Service # 3

83. A single DMA can transfer_operands to and from memory in a single a bus cycle.

- **8 bit**
- 16 bit
- 32 bit
- 64 bbit

85. DMA does not work in_mode.

- **Double Mode**
- Block Transfer
- Demand Transfer
- None of the given

86.

An addressable unit on disk can be addressed by three parameters____

- **head #, sector # and track #**
- HD #, AH# and track #
- Hd0 #, sector # and track #

AL-JUNAID TECH INSTITUTE 03041659294 www.vulmshelp.com

- None of the given

87.

Each addressable unit has a unique combination of sec#, head# and track# as its____address.

- Both Physical and Logical
- None of the given
- **Physical**
- Logical

88. In partition table, first___bytes contain code which loads the boot block of active partition.

- 128
- 286
- **446**
- 500

89.

On a single disk there can be____different Operating Systems.

- 2
- **4**
- 6
- 8

90.

The last partition table within the chain contains_entry/entries about the logical drive.

- **Single**
- Double
- Three
- Four

91.

Partition Table can be read using the extended__Services.

- 13H

0304-1659294
www.vulmshelp.com

- **14H**
- 15H
- 17H

92.

File control block (FCB) is_____byte long.

- 8
- 16
- **32**
- 64

93. __is collection of contagious blocks.

- **Cluster**
- Sector
- File
- Directory

94.A FAT 12 table can have maximum___values.

- 1024
- 2048
- **4096**
- None of given

95.

The first cluster number of file can be found in_____.

- BPB
- DPB
- **FCB** **PG # 265**
- None of the given

96.

In FAT12 if FAT entry is 000H then.

- It is a last file cluster
- Cluster is available
- **It is a Reserved cluster**
- It is next file cluster

0304-1659294
www.vulmshelp.com

97.

Total number of clusters of FAT12 are____.

- FF0 H
- FFF H
- FEF H
- **FEE H PG # 266**

98.

When we mark a file as deleted by placing 0xE5 then the chain of clusters in FAT is also replaced by___

- E5
- **0**
- 1
- N

99. keeps the back up of its boot block.

- FAT 12
- FAT 16
- **FAT 32**
- None of the given

100. Up tobytes can be used to store a file name in NTFS.

- 30
- 126
- **254**
- 510

101.

First logical sector of NTFS partition is_.

- DPB
- MFT
- **Boot sector**
- None of the given

102. In case of___the contents of file are stored in MFT.

0304-1659294
www.vulmshelp.com

- Small files
- Large files
- **Both small and large files**
- None of the given

103.

Interrupt Descriptor Table can have up to___entries.

- 64
- 128
- **256**
- 512

104. There can be__different descriptors privilege levels.

- 2
- **4**
- 8
- 10

105. In NTFS, Block # is the safest block to store the back up of boot block.

- 8
- **10** **Not sure**
- 20
- 6

106.

Size of single entry in Partition Table is____

- **512 bytes**
- 128 bytes
- 64 bytes
- 16 bytes

107.

MBR partition table can have_entries at maximum

- 1

0304-1659294
www.vulmshelp.com

- 2
- 3
- **4 Not sure**

108. __is used to identify the cause of interrupt.

- Interrupt Enable register
- **Interrupt ID register**
- Interrupt Status register
- None of the given

109.

To access bettery powered RAM, only_ports are important from programming point of view.

- **70 and 71H PG # 141**
- 71 and 72H
- 70 and 72H
- 72 and 73H

110.

The DMA requests to acquire buses through the___signal.

- **HOLD PG # 186**
- ACR
- ACK
- All of the given

111. ____used to program various common parameters of transfer for all the channels.

- None of the above
- DMA Status Register
- **DMA Command Register PG # 191**
- DMA Request Register

112. Bit #___of mode register in DMA determine the direction of a transfer.

- **2**
- 3

0304-1659294 www.vulmshelp.com

- 4
- 5

113.

Highest capacity of disk can be accessed using BIOS functions is____.

- 128 MB
- 256 MB
- **504 MB PG # 211**
- 127 GB

114.

DOS has built in limit of___blocks per cluster.

- 32
- 64
- **128 PG # 242**
- 256

115.

BIOS Parameter block is situated in_Block.

- **Boot PG # 242**
- Data
- Extended Data
- None of Given

116. The____block is the first block on disk.

- LSN =0
- **LBA=0 PG # 240**
- Both LBA=0 and LSN=0
- None of the given

117. Inside a boot block jump code part occupies_____byte.

- 5
- 8
- 11
- **3 PG # 244**

0304 1659294 www.vulmshelp.com

118.

In boot block BIOS parameter block starts from___.

- 03H
- 05H
- 08H
- **0BH**

119.

Total number of cluster of FAT16 is_

- FFF0 H
- FFFF H
- FFEF H
- **FFEE H Page 272**

120. Total____fragments can be supported for storing long file names.

- 26
- 28
- **32**
- 48

121. MCB is a bytes data structure.

- 8
- **16 PG # 321**
- 32
- 64

122.

Real mode does not support___memory allocation system.

- **Contiguous**
- Non Contiguous
- Both Contiguous and Non Contiguous
- None of the given

123.

0304-1659294
www.vulmshelp.com

For virus to propagate itself, it has to intercept interrupt__.

- 9H
- 11H
- **13H**
- 17H

124.

If virus wants to be in memory independently, it should have its own_.

- MCB
- PSP
- EB
- **Both MCB and PSP PG #334**

125. Bit #__of Eflag is used for alignment check

- 12
- 14
- 15
- **18 PG # 164**

126. First cluster in user data is numbered in a FAT based system.

- 0
- 1
- **2 PG # 258**
- 3

127. ----------BIOS services understand --.

- **LBA PG # 212**
- LSN
- Cluster #
- None

128. The first cluster number of a file can be found in---------

- BPB
- DPB
- **FCB PG # 265**
- None

0304-1659294 www.vulmshelp.com

129. The size of FS Info block is

- 64byte
- 128 byte
- 256 byte
- **512 byte PG # 300**

130. ----------In NTFS first entries are reserved.

- 4
- 6
- **16 PG # 303**
- 32

131. --------- The of boot block constitutes of BPB.

- Code part
- **Data part PG # 242**
- Both
- None

132. ----------Extended BIOS function make use of---------- address

- **LBA PG # 212**
- CHS
- LSN
- None

133. LBA address can be used in place of the CHS address.

- **True PG # 235**
- False

134. In FAT12, the maximum range of clusters is ……...

- 0 ~ FEFH
- 1~ FEFH

0304-1659294
www.vumshelp.com

- **2 ~ FEFH PG # 266**
- 3 ~ FEFH

135. NTFS volume can be accessed directly in DOS.

- True
- **False PG # 310**

136. Each partition information chunk is 16 bytes long and the last two bytes at the end of the partition tabledata part is the partition table signature whose value should be_____indicating that the code part contains valid executable code.

- 00AA
- 0055
- 050A
- **AA55 PG # 219**

137. ervice 21H/52H service returns the address of DOS internal data structures in ES: BX_____behindthe address returned lies the far address of the first MCB in memory.

- 2-bytes
- **4-bytes PG # 322**
- 6-bytes
- 8-bytes

138. 80386 can have_____control registers.

- 2
- 5
- 3
- **4 PG # 331**

139. The partition table uses the extended__service.

- **13H PG # 234**
- 14H

0304 1659294 www.vulmshelp.com

- 15H
- 16H

140. The entry point of execution in EXE File can be

- Start of the first instruction
- Start of the last instruction
- **Anywhere in the Program PG # 335**
- Can be in the middle of the program

141. Using the_entry and the FAT we can access the contents of file.

- Reserved blocks
- **Root Directory PG # 269**
- Number of FAT copies
- None of the given

142. Control information in files is maintained using

- BPB
- DPB
- **FCB PG # 256**
- FPB

143. What will happen if NTFS volume is accessed in DOS?

- Convert it to FAT volume
- Nothing will happen
- **Error of invalid media PG # 310**
- None of the given

144. LSN of FS Info block is available at

- BPB
- **FAT**
- Root Directory
- None of the given

145. DOS device drivers do not understand the_____data structures.

- FAT12

0304 1659294 www.vulmshelp.com

- FAT16
- FAT32
- **NTFS PG # 310**

146. __is a collection of contagious blocks.

- **Cluster PG # 242**
- Sector
- Byte
- None of Given

147. __used to determine the amount of conventional memory interfaced with the processor in kilobytes.

- INT 10 H
- INT 11 H
- **INT 12 H PG # 162**
- INT 13 H

148. Bit number_____of coprocessor control word is the Interrupt Enable Flag.

- **7 PG #168**
- 8
- 9
- 10

149. To distinguish 486 with Pentium CPUID Test is used.

- **True PG # 166**
- False

150. Practically____entries are there in FAT 32.

- 2^26
- 2^28
- 2^30
- **2^32 PG # 265**

151. BPB stands for____.

0304-1659294
www.vulmshelp.com

- **BIOS parameter block PG # 243**
- BIOS processing block
- Base processing block
- BIOS partition block

152. The keyboard input character scan code is received at_____port.

- **60H PG # 179**
- 61H
- 62H
- 63H

153. ___is LED control byte.

- 0xFD
- **0xED PG #181**
- 0xFF
- 0xEE

154. ___means typematic rate will be sent in next byte.

- **0xF3 PG # 180**
- 0xF4
- 0xF5
- 0xF6

155. Keyboard uses port_as status port.

- **64H PG # 177**
- 66H
- 67H
- 69H

156. The keyboard can perform___serial I/O.

- asynchronous
- **synchronous**
- Multiple

AL-JUNAID TECH INSTITUTE 0304-1659294 www.vulmshelp.com

- Single

157. Bit number 2 of port 64H Status register used for output buffer full.

- True
- **False**

158. Bit number__can declares the parity error of port 64H Status register.

- 4
- 5
- 6
- **7**

159. Bit number__of port 64H Status register used for input buffer full.

- **0**
- 1
- 2
- 3

160. Disadvantage of FAT32 is__.

- Large disk size can be managed in FAT32
- Cluster size is reduced
- Internal fragmentation is reduced
- **Very large table PG # 299**

161. Maximum possible entries in FAT12 are_____.

- 1024
- 2048
- **4096 PG # 264**
- 65536

162. What will be the value of the word located at 1Fh in DPB when number of free clusters on drive is notknown?

- 0000H

AL JUNAID TECH INSTITUTE 0304-1659294 www.vulmshelp.com

- 1111H
- **FFFFH PG # 250**
- None of the given.

163. Jump code part contains____bytes in boot block.

- **3 PG # 302**
- 5
- 8
- 11

164. Operating system name contains_bytes in boot block.

- 3
- 5
- **8 PG # 257**
- 11

165. File can be_viewed as organization of data.

- Physically
- **Logically PG # 256**
- Both logically and physically
- None of the give

166. _is used to read a block against its LSN.

- **absread() PG # 247**
- abswrite()
- lsnread()
- None of the given

167. File can be__viewed as collection of clusters or blocks.

- **Physically PG # 256**
- Logically
- Both physically and logically

0304-1659294 www.vulmshelp.com

- None

168. When we talk about FAT based file system, in user data area first cluster number is.

- 0
- 1
- **2 PG # 258**
- None of the given

169. Cluster number can also be referred as block number.

- True
- **False PG # 258**

170. To access the block within cluster using BIOS services the cluster number should be converted into__.

- CHS
- LBA
- **LSN PG # 258**
- None of the given

171. What will be the value of DL register when we are accessing C drive using undocumented service21H/32H?

- 0
- 1
- 2
- **3 PG # 249**

176. The directory structure of DOS is like_.

- Array
- **Tree PG # 256**
- Linked list
- None of the given

177. Control information about files is maintained using__.

- BPB
- DPB
- **FCB PG # 256**

0304-1659294 www.vulmshelp.com

- FPB

178. When LSN is equal to zero (0), it means_.
- First block of the disk
- **First block of the logical drive PG # 240**
- First block of hidden blocks
- None of the given

179. In FAT32, lower___ bits are used.
- 26
- **28 PG # 292**
- 30
- 32

180. _____is relative address with respect to the start of Logical Drive.
- LBA
- **LSN PG #240**
- CHS
- None of the given

181. To access drive parameter block we use undocumented service _
- 09H/32H
- 11H/32H
- 17H/32H
- **21H/32H PG # 249**

182. _____is an absolute address relative to the start of physical drive.
- **LBA PG # 240**
- LSN
- CHS
- None of the above

183. Boot block consists of__bytes.
- 64
- 128
- 256

AL-JUNAID TECH INSTITUTE 0304-1659294 www.vulmshelp.com

512 PG # 242

184. The DMA requests to acquire buses through the _____signal.

- **HOLD PG #186**
- ACR
- ACK
- None of Given

185. The keyboard device writes a code 0xFA on the port 60H to indicate that the_____.

- Input buffer is full
- **Byte has been received properly PG # 179**
- Output buffer is full
- None of the given

186. A single DMA can transfer_operands to and from memory in a single a bus cycle.

- **8-bits PG # 186**
- 16-bits
- 32-bits
- 12-bits

187. In FAT12, to calculate the address or offset from index, we need to multiply it with_____.

- 1/2
- **3/2 PG #267**
- 5/7
- 7/2

188. _____Register can be used to show that the channel is single transfer, block transfer or demandtransfer mode.

- DMA Command register
- DMA Request Register

0304-1659294 www.vulmshelp.com

- DMA Mode Register
- **DMA controller Register PG #187-188**

189. When we mark a file as deleted by placing 0xE5 then the chain of clusters in FAT is also replaced by____
   - E5
   - 1
   - **0 PG # 79**
   - N

190. Cluster size is reduced in_____.
   - FAT12
   - FAT16
   - **FAT32**
   - None of the given

191. In FAT32_____root directory entries are there.
   - 128
   - 256
   - 512
   - **None of the given**

192. If a file is having more than one cluster then it will be managed by
   - FAT
   - BPB
   - DPB
   - **None of the above**

193. Internal fragmentation is reduced in_____.
   - FAT12
   - **FAT16**
   - FAT32
   - None of the given

194. For supporting long file names,___fragments can be supported.
   - 12

0304-1659294
www.vulmshelp.com

- 20
- 26
- **32**

195. To store a cluster in FAT 32 is/are needed.
- Nibble
- Byte
- 2 Bytes
- **4 Bytes**

196. If a file size is 12K and the size of the cluster is 4K then__clusters are used for the file.
- 2
- **3**
- 4
- 5

197. We can access the contents of File by using the root directory entry and_____.
- Reserved Blocks
- Number of FAT copies
- **File Allocation Table (FAT) PG # 269**
- None of the given

198. FAT based file system can store file name inform.
- ASCII
- UNICODE
- **Both ASCII and UNICODE**
- None of the given

199. Drive parameter block is derived from___.
- FCB
- FAT
- **BPB PG # 249**

0304-1639294
www.vumshelp.com

- CPB

200. We can access Blocks for FAT using__.

- BPB
- DPB
- **FCB**
- Both BPB and DPB

201. If we know the cluster number, we can access the blocks within the cluster using BIOS services directly.

- **True PG # 258**
- False

202. _____is an internal data structure of DOS and resides in main memory.

- BPB
- DPB
- CPB
- **None of the given.**

203. The size of DPB data structure is_bytes.

- 16
- 32
- 64
- **128**

204. The size of FCB data structure is_bytes.

- **16**
- 32
- 64
- 128

205. Advantages of FAT32 is/are___.

0304-1659294
www.vulmshelp.com

- Large disk size can be managed in FAT32
- **Cluster size is reduced**
- Internal fragmentation is reduced
- All of the given

206. file system keeps the backup of its boot block.
- FAT12
- FAT16
- **FAT32**
- None of the given

207. To store a UNICODE character___is/are needed.
- Nibble
- Byte
- **2 Bytes**
- 4 Bytes

208. _____is the first block on disk.
- LSN =0
- **LBA=0 PG # 240**
- LBA=1
- Both LBA=0 and LSN=0

209. If FAT entry is between FFF0H to FFF6H in FAT16 then_.
- Cluster is available
- **It is a Reserved cluster PG # 272**
- It is next file cluster
- It is a last file cluster

210. File system used in CD's is__file system
- **Contiguous**
- Chained
- Indexed
- None

AL-JUNAID TECH INSTITUTE
0304-1659294
www.vulmshelp.com

249.A file has 2 clusters and the size of cluster is 4K. What will be the size of file?

- 2K
- **8K**
- 16K
- 32K

250.In NTFS, up to____ characters are used to store files names,

- 30
- 48
- **255 PG # 283**
- 510

251.A cluster is a collection of contiguous__.

- **Blocks PG # 242**
- Sectors
- Bytes
- None of Given

252.In BPB, root directory is saved in__.

a. **Cluster#0**
b. Cluster#1
c. Cluster#2
d. Cluster#3

253.

254. In NTFS, total sizes of MFT entries are_____.

a. **16-bytes PG # 303**
b. 20-bytes
c. 26-bytes
d. 32-bytes

AL-JUNAID TECH INSTITUTE 0304-1659294 www.vulmshelp.com

255. In NTFS,___store the contents of file.

a. Both small & large file Record

b. Small record

c. Large Record

d. None of given

256. In NTFS, contents and indexed of file is stored in_____.

a. Small record

b. Large Record

c. Both small & large file Record

d. None of given

257. Total No. of bytes that can be stored in Keyboard Buffer is_____.

a. 16

b. 32 PG #54

c. 64

d. 128

258. BIOS support__UARTS as COM ports.

a. 6

b. 4 PG #113

c. 3

d. 2

259. DCE stands for_____.

a. Data communication equipment PG # 109

b. Distributed Computing Environment

0304-1659294
www.vulmshelp.com

c. Data Communications Equipment

d. Data Carrier Equipment

260. In counter register bit no. 3 changes its value between 0 and 1 with in__clock cycles

a. 1

b. 2

c. 4

d. 16 PG #69

261. In each byte is needed to be encapsulated in start and end.

a. Synchronous communication

b. Asynchronous communication PG # 106

c. Both

d. None of given

262. The__service # is not used in any interrupt.

a. 01

b. 02

c. 03

d. FF

263. If we want to send printing on the printer then we have to perform following steps.

a. Initialize printer

b. Read Status

c. Check Error

d. All of the given

264. If printer is__then printer sends back the ACK signal to the printer interface

AL-JUNAID TECH INSTITUTE 0304-1659294 www.vulmshelp.com

a. idle PG # 97

b. busy

c. Out of paper

d. None of the given

265. DSR stands for_____.

a. Data set ready PG # 111

b. Data service ready

c. Data stock ready

d. None of the given

266. At IRQ 7 Interrupt #_____is used.

a. 0x0A

b. 0x0B

c. 0x0C

d. 0x0FPG # 95

267. The memory addresses of COM ports remain same for all computers

a. True

b. False

268. In keyboard status byte bit no. 2 and 3 are used for ctrl and alt keys respectively. which of the following condition is used to check that Ctrl + Alt keys are pressed. Where: unsigned char far * scr = (unsigned charfar *)(0x00400017);

a. if (((*scr)&12)==12)

b. if (((*scr)&8)==8)

0304-1659294
www.aljunaidtech.com

c. if (((*scr)&4)==4)

d. if (((*scr)&2)==2)

269. In case of synchronous communication a timing signal is required to identify the start and end of a bit.

a. True PG # 105

b. False

270. The baud rate is set in accordance with the divisor value loaded within the UART internal registers base +0 and base +1.

a. True PG # 114

b. False

271. ----------Software based flow control make use of ----------control characters

a. Xon

b. XOFF

c. Both PG # 135

d. None

272. __is used to read time from RTC

a. 1A\02H PG # 137

b. 1A\03H

c. 1A\04H

d. 1A\05H

0304-1659294
www.aimshelp.com

273. Int____service 0 can be used to set the line parameter of the UART or COM port.

a. 14H PG # 119

b. 15H

c. 13H

d. None of the given option

274. When LBA is equal to zero (0), it means_____.

a. First block of the disk PG # 240

b. First block of the logical drive

c. First block of the hidden block

d. None of the given

275. How many bytes can be used to store a file name in NTFS?

a. 128

b. 255

c. 510

d. 1024

276. In boot block BIOS parameter block starts from 03H

a. 05H

b. 08H

c. 0BH PG # 302

277. IN NTFS, FAT and root directory is replaced by

a. FCB

b. MFT PG # 301

c. Hidden blocks

0304-1659294 www.vulmshelp.com

d. Boot sector

278. Block # 2 is the safest block to store the backup of boot block.

a. True

b. False

279. The keyboards interface as discussed earlier uses the IRQ0 and the port 64H as data port.

a. True

b. False

280. FAT12 will have 12-bit wide entries and can have 2^12=4096 entries maximum

a. True

b. False

AL-JUNAID TECH INSTITUTE
0304-1659294
www.vulmshelp.com

CS609 Final Term Solved MCQs with References Without Repetitions 14/02/2013

In BPB, root directory is saved in _____. (BIOS parameter block)
Cluster#0
Cluster#1 (Ref)
Cluster#2
Cluster#3

In NTFS, total sizes of MFT entries are _____.
16-bytes (page303= An MFT entry is 1024 bytes in size. The first 16 entries of the MFT are reserved)
20-bytes
26-bytes
32-bytes

In NTFS, _____ store the contents of file.
Both small & large file Record (Reference)
Small record
Large Record
None of given

Question # 3
Cluster size is reduced in _____.
Select correct option:
FAT12 (Ref1, Ref2)
FAT16
FAT32
None of the given

Question # 4
To store a cluster in FAT 32 _____ is/are needed.
Select correct option:
Nibble
Byte
2 Bytes
4 Bytes (Ref1, Ref2)

- A single digit of a binary number is a bit.
- A byte is 8 bits
- A nibble is 4 bits (half a byte)
- A word is 2 bytes

awais4you@gmail.com

Question # 4
When Driven LSN is equal to zero (0), it means _____.
LSN is relative address with respect to the start of Logical (Lec#30 p#240)

Question # 5
When LSN is equal to zero (0), it means _____.
Select correct option:
First block of the disk
First block of the logical drive (p240)
First block of hidden blocks
None of the given

The LBA=0 block is the first block on disk. Whereas each logical partition has LSN=0 block which is the first block in logical drive. (p240)

Question # 6
File can be _____ viewed as collection of clusters or blocks.
Select correct option:
Physically (p256)
Logically
Both logically and physically
None of the give

Question # 7
The size of FCB data structure is _____ bytes.
Select correct option:
16
32 (close to 37)
64
128

- An *FCB* is a 37-byte *data structure*

Question # 10
File can be _____ viewed as organization of data.
Select correct option:
Physically
Logically (p256)
Both logically and physically
None of the give

Q#4 File is …… viewed as an organization of data physically it can be collection of clusters or blocks
Ans: Logically Lec#33 p#256

Question # 1
_____ file system keeps the backup of its boot block.

Waisayou@gmail.com

Select correct option:
FAT12
FAT16
FAT32 (Ref)
None of the given

Question No. 3
Keyboard Status Byte is located at the address
0040:0000H
0040:0013H
0040:0015H
0040:0017H (p29)

Question No. 5
Maximum number of interrupts in a standard PC is
64
128
256 (p10)
512

Using the____ entry and the FAT we can access the contents of file.
Reserved blocks
Root Directory (p269)
Number of FAT copies
None of the given

Control information in files is maintained using
BPB
DPB
FCB (p256)
FPB

What will happen if NTFS volume is accessed in DOS?
Convert it to FAT volume
Nothing will happen
Error of invalid media (p310)
None of the given

LSN of FS Info block is available at
BPB
FAT (p299)
Root Directory
None of the given

DOS device drivers do not understand the_____ data structures.
FAT12

aiais4you@gmail.com

FAT16
FAT32
NTFS (p310)

Question # 4
To store a UNICODE character ____ is/are needed.
Select correct option:
Nibble
Byte
2 Bytes (Ref)
4 Bytes

Operating system name contains ____ bytes in boot block.
3
5
8 (p253)
11

Question # 6
In FAT32, lower _____ bits are used.
Select correct option:
26
28 (p292)
30
32

Question # 8
What will be the value of DL register when we are accessing C drive using undocumented service 21H/32H?
Select correct option:
0
1
2
3 (p249)

Question # 1
To access drive parameter block we use undocumented service _____
Select correct option:
09H/32H
11H/32H
17H/32H
21H/32H (p249)

Question # 10
_____ is relative address with respect to the start of Logical Drive.
Select correct option:

anais4you@gmail.com

LBA
LSN (p240)
CHS
None of the given

- LBA is physical or absolute address.

Question # 3
Maximum possible entries in FAT12 are _____.
Select correct option:
1024
2048
4096 (p264)
65536

- FAT12 will have 12-bit wide entries and can have 2^12=4096 entries maximum.

Question # 6
Jump code part contains ____ bytes in boot block.
Select correct option:
3 (p302, p244) While "short jump" only requires 2 bytes.
5
8
11

Question # 8
Drive parameter block is derived from _____.
Select correct option:
FCB
FAT
BPB (p249)
CPB

Question # 10
_____ is the first block on disk.
Select correct option:
LSN =0
LBA=0 (p240)
LBA=1
Both LBA=0 and LSN=0

Question # 3
If FAT entry is between FFF0H to FFF6H in FAT16 then _____.
Select correct option:
Cluster is available
It is a Reserved cluster (p496 PC Intern Encyclopedia of System Programming)

awais4you@gmail.com

It is next file cluster
It is a last file cluster

Question # 4
What will be the value of the word located at 1Fh in DPB when number of free clusters on drive is not known?
Select correct option:
0000H
1111H
FFFFH (p250)
None of the given.

Question # 7
In dos we have limit of _____ .
Select correct option:
128 blocks per cluster (p242)
256 blocks per cluster
32 blocks per cluster
64 blocks per cluster

- DOS has a built in limit of 128 blocks per cluster.
- But **practically** limit of 64 blocks per cluster has been established.

Question # 2
The practical limit of blocks per cluster is _____.
Select correct option:
32 blocks per cluster
64 blocks per cluster (p242)
128 blocks per cluster
256 blocks per cluster

Question # 8
We can access the contents of File by using the root directory entry and _____.
Select correct option:
Reserved Blocks
Number of FAT copies
File Allocation Table (FAT) (p269)
None of the given

Question # 10
When we mark a file as deleted by placing 0xE5 then the chain of clusters in FAT is also replaced by _____.
Select correct option:
E5
1

vulais4you@gmail.com

0 (p279)
N

Highest capacity physical capacity of the disk according to the IDE interface is _____.
127 GB (p212)
100 GB
80 GB
300 GB
If CPUID instruction is not present then the processor can be a _____
486 processor (p166)
386 processor
286 processor
All of the given choices

Extended memory is available if the processor is of the type _____
AT (p171)
XT
All of the given choices
None of them

The built in mechanism within the UART for error detection is_____
Hamming code
Parity (p107)
CRC16 (cyclic redundancy check 16 bit)
CRC32 (cyclic redundancy check 32 bit)

In Protected Mode, the segment registers are used as _____
Descriptor
Selector (p326)
All of the given choices
None of the given choices

_____ is an absolute address relative to the start of physical drive.
LBA (p240)
LSN
CHS
None of the above

NTFS volume can be accessed directly in DOS.
True
False (p310)

_____ is used to read a block against its LSN.
Select correct option:
absread() (p247)
abswrite()

awais4you@gmail.com

lsnread()
None of the given

Cluster number can also be referred as block number.
Select correct option:
True
False (Cluster is not same as block number) (P258)

_____ is a collection of contagious blocks.
Cluster (p242)
Sector
Byte
None of Given

The directory structure of DOS is like _____.
Select correct option:
Array
Tree (p256)
Linked list
None of the given

1) DOS command _____ which gives the status of the memory and also points out which memory area occupied by which process.
mem/d (P#13)
mem/e
mem/m
None of the given

2) Each entry in the IVT is _____ in size.
4-bytes (P # 12)
6-bytes
8-bytes
2-bytes

3) The tail value of the keyboard buffer should be examined to get to the _____ of the buffer.
Start (P # 56)
End
Middle
None of given)

4) Interrupt Procedures are Reentrant procedures.
True (P # 38)
False

5) _____ is Disk interrupt.
10H

avais4you@gmail.com

11H
13H (P # 42)
14H

6) Boot block consists of _____ bytes.
64
128
256
512 (P # 242)

7) In parallel communication, the maximum numbers of bits we can send between two computers are _____.
2-bits
4-bits
6-bits
8-bits

8) 14h include _____ which is used to send a byte.
Service #0
Service #1 (P # 121)
Service #2
Service #3

9) The status register _____ is the main control register.
B (P # 146)
A
C
D

10) The DMA requests to acquire buses through the _____ signal.
HOLD (P # 186)
ACR
ACK
None of Given

11) The keyboard device writes a code 0xFA on the port 60H to indicate that the _____.
Input buffer is full
Byte has been received properly (P # 179)
Output buffer is full
None of the given

12) A single DMA can transfer _____ operands to and from memory in a single a bus cycle.
8-bits (P # 186)
16-bits
32-bits

awais4you@gmail.com

12-bits

14) In FAT12, to calculate the address or offset from index, we need to multiply it with ___.
1/2
3/2 (p# 267)
5/7
7/2

15) _____ Register can be used to show that the channel is single transfer, block transfer or demand transfer mode.
DMA Command register
DMA Request Register
DMA Mode Register
DMA controller Register (p187,188 Handouts && p234 PC Intern Encyclopedia of System Programming)

16) LBA address can be used in place of the CHS address.
True (p# 235)
False

17) In FAT12, the maximum range of clusters is ……..
0 ~ FEFH
1~ FEFH
2 ~ FEFH (p# 266, 272)
3 ~ FEFH

18) Each partition information chunk is 16 bytes long and the last two bytes at the end of the partition table data part is the partition table signature whose value should be_____ indicating that the code part contains valid executable code.
00AA
0055
050A
AA55 (p# 219)

19) _____ is used to identify the cause of interrupt.
Modem control register
Modem status register
Interrupt ID register (P # 116)
None

20) In NTFS, up to _____ characters are used to store files names,
30
48
255 (p283) Doubtful
510

owais4you@gmail.com

21) A cluster is a collection of contiguous _____.
Blocks (P # 242)
Sectors
Bytes
None of Given

26) Service 21H/52H service returns the address of DOS internal data structures in ES: BX _____behind the address returned lies the far address of the first MCB in memory.
2-bytes
4-bytes (P # 322)
6-bytes
8-bytes

27) 80386 can have _____ control registers.
2
5
3
4 (P # 331)

-80386 and above have 4 Control Registers CR0 ~ CR3.

29) The partition table uses the extended _____ service.
13H (P # 234)
14H
15H
16H

30) Partition Table can be read using the extended _____ Services.
13H (P # 234)
14H
15H
None of given

31) The entry point of execution in EXE File can be
Start of the first instruction
Start of the last instruction
Anywhere in the Program (P # 335)
Can be in the middle of the program

Question # 3
To access the block within cluster using BIOS services the cluster number should be converted into _____.
Select correct option:
CHS
LBA
LSN (p258)

anais4you@gmail.com

None of the given

If the value of attribute byte is **4** means it is a folder. **(p279)**

- The attribute byte of this entry is 0x20 which indicates that it's a directory.

In IRQ2 and IRQ3 which one has the highest priority?
Can't be determined
Both have same priority
IRQ3
IRQ2 (p47 =>As IRQ0 has the highest priority and IRQ7 has the lowest.)

Write stands for
THR = transmitter holding register
RBR = receiver buffer register
DLL = (band rate divisor) low
DLM = (band rate divisor) high

Write stands for.
DSR = Data set ready
DTR = Date terminal ready
RTS = ready to send
CTS = clear to send

Question No: 1
Following is not a method of I/O
Programmed I/O
Input driven I/O
Hardware Based I/O (p4)
None of given

It is possible to perform I/O operations from three different methods:
• Programmed I/O
• Interrupt driven I/O
• DMA driven I/O (Direct Memory Access)

Question No: 2
The Function of I/O controller is to provide _____.
I/O control signals
Buffering
Error Correction and Detection
All of given (p5)

Question No: 3
Which of the following are types of ISR _____.
BIOS (Basic I/O service) ISR
DOS ISR

awais4you@gmail.com

ISR provided by third party device drivers
All of the given (p13)

Question No: 4
Interrupt service number is usually placed in _____ register.
CH
CL
AH (p26)
AL

Question No: 5
NMI Stand for
Non Maskable Interrupt (p46)
Non Multitude Interrupt
Non Maskable Instruction
None of Given

Question No: 6
A single interrupt controller can arbitrate among ____ different devices.
4
6
8 (p47)
10

Question No: 8
The microprocessor package has many signals for data. Below are some incorrect priority order (Higher to Lower).
Reset,Hold,NMI,INTR (p46)
NMI, INTR,Hold,Reset
INTR,NMI,Reset,Hold
None of the Given

Question No: 9
The interval timer can operate in ____modes.
Three
Four
Five
Six (p72)

Question No: 13
The following command "outportb (0x61,inportb(0x61) | 3);" will _____ .
Turn on the speaker (p74)
Turn off the speaker
Toggle the speaker
None of the above

awais4you@gmail.com

Question No: 11
The following command “outportb (0x61,inportb(0x61) & 0xFC);” will
Turn on the speaker
Turn off the speaker (p75)
Toggle the speaker
None of the given

Question No: 12
The PPI acts as an interface between the CPU and a parallel _____ .
I/O device (p83)
CPU
BUS
None of Given

Question No: 13
BIOS DO NOT support _____.
LPT1
LPT2
LPT3
LPT4 (p91 =>BIOS support up to three parallel ports)

Question No: 14
_____ bit is cleared to indicate the low nibble is being sent.
D1
D2
D3
D4 (p101)

Question No: 15
The bit _____ of Line control register in UART, if cleared will indicate that DLL is the data register.
1
3
5
7 (p114)

Question No: 17
_____ used to determine the amount of conventional memory interfaced with the processor in kilobytes.
INT 10 H
INT 11 H
INT 12 H (p162)
INT 13 H

Question No: 18
Bit number _____ of coprocessor control word is the Interrupt Enable Flag.

awais4you@gmail.com

7 (p168)
8
9
10

Question No: 19
To distinguish 486 with Pentium CPUID Test is used.
True (p166)
False

Question # 3
_____ file system is used in NTFS based systems.
Select correct option:
Contiguous
Chained
Indexed (Ref)
None of the given

Question No: 20
Communication between keyboard and keyboard controller is _____.
Asynchronous serial
Synchronous serial (p177)
Parallel communication
None of the given

In NTFS, boot sector is stored at
First and 6th sector
First and Last sector
Only at Last sector
Only at First sector (p492 PC Intern Encyclopedia of System Programming)
- Boot sectors always located in sector 0.

Practically _____ entries are there in FAT 32.
Select correct option:
2^26
2^28
2^30
2^32 (p265)

Question No: 2
Standard PC operates in two modes in terms of memory which are
Real mode and Extended Mode
Base mode and Memory Mode
None of the given
Real mode and protected mode (p319)

awais1you@gmail.com

Question No: 3
BPB stands for _____.
BIOS parameter block (p241)
BIOS processing block
Base processing block
BIOS partition block

Question No: 5
IVT is a table containing _____ byte entries each of which is a far address of an interrupt service routine.
2
4 (p20)
8
16

Question No: 6
Each paragraph in keep function is ____ bytes in size.
4
8
16 (p24)
32

Question No: 8
A software interrupt does not require EOI (End of interrupt).
True (p49)
False

Question No: 9
To store each character in keyboard buffer ____ bytes are required.
2 (p54) Has 32 bytes of memory 2 bytes for each character
4
6
8

Question No: 10
Interrupt ____ is empty; we can use its vector as a flag.
9H
13H
15H
65H (p65)

Question No: 11
Command register is an _____ bit register
4
8 (p71)
16

awais4you@gmail.com

32

FAT based file system can store file name in ____ form
ASCII
UNICODE
Both ASCII and UNICODE (Ref)
None

- Short filenames use ASCII character set. Long filenames are stored in Unicode.

How many bytes can be used to store a file name in NTFS?
128
255 (Ref) All NTFS file names are stored in a format called *Unicode*.
510
1024

Question # 4
When we talk about FAT based file system, in user data area first cluster number is _____.
Select correct option:
0
1
2 (p258, p241)
None of the given

____ is the first logical sector of NTFS partition.
DPB
MFT
Boot sector (Ref)
None

In boot block BIOS parameter block starts from
03H
05H
08H
0BH (page#302, Ref)

Advantages of FAT32 is/are_____
Large disk size can be managed in FAT32
Cluster size is reduced
Internal fragmentation is reduced
All of the given

Question # 9
Disadvantage of FAT32 is _____.
Select correct option:
Large disk size can be managed in FAT32

nwais4you@gmail.com

Cluster size is reduced
Internal fragmentation is reduced
Very large table (p299)

IN NTFS, FAT and root directory is replaced by
FCB
MFT (p301)
Hidden blocks
Boot sector

File control block (FCB) is _____ byte long.
32 (Ref) A full FCB is 36 bytes long, close to 32
64
16
128

Control information about files is maintained using _____.
Select correct option:
BPB
DPB
FCB (p256)
FPB

CS609 Quiz # 2 fall 2012 Solved with Reference by +~Sparkle Fairy~+
Lecture Included 21 to 40

Q#1 …… Is used to program various common parameters of transfer for all the channels
Ans: DMA Command Register Lec#23 p#191

Q#5 a cluster is same as block
Ans: False Lec#33 p#258

Q#6 In NTFS FAT and root directory has been replaced by ……..
Ans: MFT Lec#39 p#301

Q#7 if NTFS volume is accessed in DOS then what will happen…….?
Ans: it will fire error of invalid media Lec#40 p#310

Q#8 …….. Structure is tree like structure
Ans: DOS Directory Lec# 33 p# 256

Q#9 the accumulative time that is required to access the selected block is called ……..
Ans: Access time Lec#25 p#205

Q#10 DMA may not support memory to memory transfer.

vu4you@gmail.com

Ans: True Lec#25 p#199

Question # 3
…… is used to read a block given its LSN.
absread ( ) function (Lec# 31 p#248)

Doubtful MCQS

28) When the content of the file is _____, it means it can be terminated or no operation left.
A
B
C
D

13) For supporting long file names, _____ fragments can be supported.
24
26
32
40

25) In NTFS, contents and indexed of file is stored in _____.
Small record
Large Record
Both small & large file Record
None of given

_____ is an internal data structure of DOS and resides in main memory.
Select correct option:
BPB
DPB
CPB
None of the given.

If three Programmable interrupt controllers are cascaded then how many interrupt driven hardware IO devices can be attached _____
12
18
23
24

On the execution of IRET instruction, number of bytes popped from stack is _____
4 bytes
6 bytes
8 bytes
10 bytes

awais4you@gmail.com

Question # 4
If a file size is 12K and the size of the cluster is 4K then _____ clusters are used for the file.
Select correct option:
2
3
4
5

Question # 5
If we know the cluster number, we can access the blocks within the cluster using BIOS services directly.
Select correct option:
True
False

If a file is having more than one cluster then it will be managed by
FAT
BPB
DPB
None

A file has 2 clusters and the size of cluster is 4K. What will be the size of file?
2K
8K
16K
32K

In FAT32____ root directory entries are there
128
256
512
None

Block # 2 is the safest block to store the backup of boot block.
True
False

In NTFS, Backup of boot block is stored at block # _____.
Select correct option:
2
6
8
10

We can access blocks for FAT using_____

awais4you@gmail.com

BPB
DPB
FCB
Both BPB and DPB

File system used in CD's is_____ file system
Contiguous
Chained
Indexed
None

The size of DPB data structure is_____ bytes
16
32
64
128

Question No. 4
If we use keep (0, 1000) in a TSR program, the memory allocated to it is
64000 bytes
32000 bytes
16000 bytes
80000 bytes

awais4you@gmail.com

For More Visit

www.VUAnswer.com

DMA Controller

DMA is a device which can acquire complete control of the buses and hence can be used to transfer data directly from port to memory or vice versa. Transferring data like this can prove faster because a transfer will consume 2 bus cycles if it is performed using the processor. So in this approach the processor is bypasses and its cycles are stolen and are used by the DMA controller.

Direct Memory Access (DMA)

The latch B of the DMA interface is used to hold the higher 4 or 8 bits of the 20 or 24 bit absolute address respectively. The lower 16bits are loaded in the base address register and the number of bytes to be loaded are placed in the count register. The DMA requests to acquire buses through the HOLD signal, it receives a HLDA (Hold Acknowledge) signal if no higher priority signal is available. On acknowledgment the DMA acquires control of the buses and can issue signals for read and write operations to memory and I/O ports simultaneously. The DREQ signals are used by various devices to request a DMA operation. And if the DMA controller is successful in acquiring the bus it sends back the DACK signal to signify that the request is being serviced. For the request to be serviced properly the DMA channel must the programmed accurately before the request.

A single DMA can transfer 8bit operands to and from memory in a single a bus cycle. If 16bit values are to be transmitted then two DMA controllers are required and should be Cascaded

DMA Programming Model

• DMA has 4 – Channels
• Each Channel can be programmed to transfer a block of maximum size of 64k.
• For each Channel there is a
• **Base Register**
• **Count Register**
• **Higher Address Nibble/Byte is placed in Latch B.**
• The Mode register is conveyed which Channel is to be programmed and for what purpose i.e. Read Cycle, Write Cycle, Memory to memory transfer.
• A request to DMA is made to start it's transfer.

Internal Registers

• No of 16 & 8 bit Internal registers
• Total of 27 internal registers in DMA

Register	Number	Width
Starting Address	4	16
Counter	4	16
Current Address	4	16
Current Counter	4	16
Temporary Address	1	16
Temporary Counter	1	16
Status	1	8
Command	1	8
Intermediate Memory	1	8
Mode	4	8
Mask	1	8
Request	1	8

For More Visit

www.VUAnswer.com

DMA Modes

- Block Transfer
- Single Transfer
- Demand Transfer

In block transfer mode the DMA is programmed to transfer a block and does not pause or halt until the whole block is transferred irrespective of the requests received meanwhile.
In Single transfer mode the DMA transfers a single byte on each request and updates the counter registers on each transfer and the registers need not be programmed again. On the next request the DMA will again transfer a single byte beginning from the location it last ended.
Demand transfer is same as block transfer, only difference is that the DREQ signal remains active throughout the transfer and as soon as the signal deactivates the transfer stops and on reactivation of the DREQ signal the transfer may start from the point it left.

Programming the DMA controller

The following table shows the different DMA controller registers which are used to determine the status of the controller or define the parameters:

DMA register in the PC/XT (or AT) that directs the DMA controller

Register	Port*	Port**	Read	Write
Status	08h	0D0h	x	
Command	08h	0D0h		x
Request	09h	0D2h		x
Masking	10Ah	0D4h		x
Mode	0Bh	0D6h		x
Byte\Word-FlipFlop	0Ch	0D8h		x
Intermediate memory	0Dh	0DAh	x	
Reset	0Dh	0DAh		x
Masking reset	0Eh	0DCh		x
Masking	0Fh	0DEh		x

* slave in an AT / only one DMA in a PC/XT
** master in an AT / not present in a PC/XT

Before you access one of these registers, decide if you're addressing the master or the slave. If you have a PC/XT that has only one DMA controller, it's not possible to access a second DMA controller (master in the AT).

The command register is located at the same port address as the status register. It goes through several settings on the DMA controller. Some of these settings, especially the 5 bit, are interesting in certain situations for DMA programming using software. The problem with this register, however, is that it cannot be selected and used due to the status of the other bits.

Terminal count if reached signifies that the whole of the block as requested through some DMA channel has been transferred.

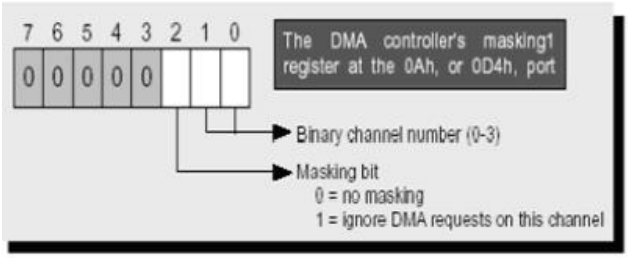
This is the command register. It is used to program various common parameters of transfer for all the channels.

DMA Request Register

The request register is used to initiate a DMA transfer under software control. This is done by simulating the activation or clearing of one of the DREQx lines The request register is also used to initiate a memory to memory transfer, since a peripheral device is not involved and therefore cannot send a signal over a DREQx line.

This register can be used to simulate a DMA request through software (in case of memory to memory transfer). The lower 2 bits contains the channel number to be requested and the bit # 2 is set to indicate a request.

DMA Mask–1 Register



Another way to mask or make a channel receptive to DMA requests is provided by mask register 2. In contrast to the mask register 1, all four channels are affected. Use this register only to change the status for all four channels simultaneously.

As its name suggests, the mode register determines a channel's operating mode. You can specify if the next DMA transfer will happen as a single transfer, a block transfer, or a demand transfer. It also specifies if the channel is to cascade two DMA controllers. In most cases you won't have to change this later setting since this happens when the computer is booted.

This register can be used to mask/unmask requests from a device for a certain DMA channel. The lower 2 bits contains the channel number and the bit #2 is set if the channel is to be masked.

DMA Mask–2 Register

This register can also be used to mask the DMA channels. It contains a single bit for each channel. The corresponding bit is set to mask the requests for that channel.

DMA Mode Register

Bit 5 of the mode register, determine the "direction" of a transfer. This "direction" isn't to or from a peripheral, rather it's forward or backward direction in memory. So you can decrement instead of increment the memory address during a DMA transfer. In his case, a data block is read backwards to forwards by the peripheral. Also the ending address of the buffer is loaded into the proper register before starting the transfer.

This register can be used to set the mode on each. The slide shows the detail of the values and bits which should be placed in the register in order to program a required mode.

Setup & Query of DMA

To set up one of these registers to determine the start address or the length of a DMA transfer, you must output to port 0Ch or 0D8h. An internal FlipFlop, lowered to zero, shows the state of a 16-bit transfer. After the FlipFlop is lowered to zero, it sends the low-order byte of the address to the port, for example port 0C4h for channel 1 of the AT master DMA controller (AT channel 5). This output trips the internal FlipFlop. The port now knows that the most signficant byte of the address is coming. This procedure is necessary because access to the different 16-bit registers has to fit into the 8-bit wide DMA hardware. Therefore, a 16-bit value has to be divided into a low byte and a high byte. And since the low and high bytes are

A channel is programmed for a start address and the count of bytes to be transferred before the transfer can take place. Both these values are placed in various registers according to the channel number as shown by the slide above. Once the transfer starts these values start changing. The start address is updated in to the current address and the count is also updates as bytes are transferred. During the transfer the status of the transfer can be analyzed by getting the values of these registers listed In the slide above for the channel(s) involved in the transfer.

File System

File System

•Disk Architecture
•Disk Partitioning
•File systems

For More Visit

www.VUAnswer.com

Disk Architecture

•Disk is a circular which is hollow from the center
•This shape is inherently useful for random access.

Tracks are the circular division of the disk and the sectors are the longitudinal division of the disk

Addressable unit Parameters

• Heads
•Sectors
•Tracks

An addressable unit on disk can be addressed by three parameters i.e. head #, sector # and track #. The disk rotates and changing sectors and a head can move to and fro changing tracks. Each addressable unit has a unique combination of sec#, head# and track# as its physical address.

Blocks

• Blocks are the sectors per track
•Smallest addressable unit in memory
•Address of block is specified as a unique combination of three parameters (i.e. track, head, sec)

Density of Magnetic media

• Density of magnetic media is the determinant of

the amount of data that can reside stably on the disk for example floppy disk come with different densities.

•Double Density
•High Density

Effect of surface area on disk size

• Increasing the surface area clearly increases the amount of data that can reside on the disk as more magnetic media no resides on disk but it might have some drawbacks like increased seek time in case only one disk platter is being used

Effect of surface area on disk size

• Increasing the surface area clearly increases the amount of data that can reside on the disk as more magnetic media no resides on disk but it might have some drawbacks like increased seek time in case only one disk platter is being used

Cylinders

• In case of hard disk where there are number of platters the term track is replaced by cylinder
•Cylinder is a collection of corresponding tracks if track on platter changes so will the tracks on rest of the platters as all the heads move simultaneously

Rotational Delay

• While accessing a selected block the time required by the disk to rotate to the specified sector is called rotational delay

Seek Time

• While accessing a selected block Time required by the head to reach the particular track/cylinder is called seek time

Access Time

• The accumulative time that is required to access the selected block is called access time
•Access time includes the delay required by disk rotation as well as head movement.

Head is like Electric Coil

• Disk follow the basic principle of magnetism of dynamo.
•When ever a magnetized portion of disk runs along the coil like head electricity is produced in the head which is interpreted as a logic 1
•And whenever a demagnetized portion on the

For More Visit

www.VUAnswer.com

disk runs through the head no electricity is produced in head which is interpreted as logic 0

Head position and precautions

• The head is touching the surface of floppy disk which rotates at a low speed of 300 RPM
•The head is not touching the surface of hard disk which run at high speeds up to 9600 RPM but is at a few microns distance from the surface
•All the magnetic disk are made out of magnetic media and hence data may be lost by exposing them to sunlight, heat, radiation, magnetic or electric fields.
•Dust is harmful and even fatal in case of head disk by the virtue of its speed and its distance of head from the surface

Hard Disk

Reading/Writing a physical Block

• biosdisk(int cmd, int drive, int head, int track, int sector, int nsects, void * buffer);
Cmd 0 = disk reset
1 = disk status (status of last disk operation)
2 = disk read
3 = disk write
4 = disk verify
5 = disk format

The function biosdisk() can be used to read or write a physical block. The slide shows its parameter. It takes the command (cmd), drive number, head number, track number, number of sectors to be read or written and the memory from where the data is to read from or written to. Command signifies the operation that is to be performed.

Reading Writing a physical Block

• Drive 0x80 = first fixed disk (physcial drive)
0x81 = second fixed disk
0x82 = third fixed disks
…. ….
0x00 = first removable disk
0x01 = second removable disk
…. ….
Drive number is described in the slide below it starts from 0 for first removable disk and starts from 0x80 for first fixed disk.

Limitation of biosdisk

• Biosdisk() calls int 13H/0/1/2/3/4/5
•Details of 13H services used by Biosdisk()
•On Entry
AH = service #

For More Visit

www.VUAnswer.com

AL=No. of sectors
BX = offset address of data buffer
CH = track #
CL = sector #
DH = head/side #
DL = Drive #
ES = Segment Address of buffer.

However there are some limitation of this biosdisk() while using large disks. This function uses the int 13H services listed in the slide above.

Limitation of biosdisk()

• Large sized disk are available now with thousands of tracks
•But this BIOS routine only is capable of accessing a max. of 1024 tracks.
•Hence if a large disk is being used not whole of the disk can be accessed using this routine.

The parameter sizes provided by these services may not be sufficient to hold the track number of block to be accessed.

Extended BIOS functions

• Extended BIOS functions of int 13h can be used for operations on higher tracks
•As discussed later usual BIOS functions can access a maximum of 504MB of disk approx.

Highest biosdisk() capacity

• Hence the highest capacity of disk can be accessed using bios functions is
•63x16x1024x512= 504 MB approx.

But IDE disk interface can support disks with memory space larger than 504MB

Highest IDE capacity

• Hence highest physical capacity of the disk according to the IDE interface is
255x16x65536x512 = 127GB
•Extended BIOS functions allow to access disk with sizes greater than 504 MB through LBA translation.

Extended services require that the address of the block is specified as a LBA address.

LBA Translation Method

• Each unique combination of three parameters is assigned a unique index as shown below
•Firstly all the sectors of a cylinder are indexed for head=0, then when whole track has been indexed the sector in the track of same cylinder with head =1 are indexed and so on up till the end of all heads
When done with one cylinder the same is repeated for the next cylinder till the end of cylinders

For More Visit

www.VUAnswer.com

LBA translation is done by numbering the blocks with a single index. The indexes are assigned to blocks as shown in the slide below. In terms of the disk geometry firstly all the sectors of a track will be indexed sequentially, then the track exhausts the next track is chosen on the other side of the disk and so on all the tracks in a cylinder are indexed. When all the blocks within a cylinder has been indexed the same is done with the next cylinder.

if the CHS (cylinder, head , sector) address of a disk is known it can be translated in to the LBA address and vice versa. For this purpose the total number of cylinders, heads and sectors must also be known.

Mathematical Notation for LBA translation

- LBA address = (C * H' +H)* S' + S – 1

Where

C = Selected cylinder number
H' = No. of heads
H = Selected head number
S'=Maximum Sector number
S= Selected Sector number

Also conversely LBA to CHS translation can also be done using the formulae discussed in the following slide but for this the total number of cylinders, heads and sectors within the disk geometry should be known

LBA to CHS translation

- Conversely LBA address can be translated into CHS address

```
cylinder = LBA / (heads_per_cylinder *
                                sectors_per_track)

temp = LBA % (heads_per_cylinder *
                                sectors_per_track)

head = temp / sectors_per_track
sector = temp % sectors_per_track + 1
```

Disk Address Packet is a data structure used by extended int 13H services to address a block and other information for accessing the block. Its structure is defined in the slide below.

Disk Address Packet

Offset	Size	Description
0	Byte	Size, Should not be less than 16
1	Byte	Reserved
2	Byte	No. of blocks to transfer, Max value no greater than 7FH
3	Byte	Reserved
4	Double Word	Far address of buffer
8	Quad word	LBA address

For More Visit

www.VUAnswer.com

Hard Disk, Partition Table

Extended Read

• Service used for extended read is int 13h/42h
On Entry
AH=42H
DL=drive #
DS:SI= far address of Disk address packet
On Exit
If CF=0
AH=0= Success
If CF=1
AH= Error code

Interrupt 13H/42H can be used to read a LBA addressed block whose LBA address is placed in the Disk Address packet as described in the slide above.

Extended Write

• Service used for extended write is int 13h/43h
On Entry
AH=43H
AL=0,1 write with verify off
2 write with verify on
DL=drive #
DS:SI= far address of Disk address packet
On Exit
If CF=0
AH=0= Success
If CF=1
AH= Error code
Similarly int 13H / 43H can be used to write onto to LBA addressed block as described in the slide above.

The above slides list a program that that performs a block read operation using the interrupt 13H/42H. A structure of type DAP is create an appropriate values are placed into it which includes its LBA address. The offset address of dap is placed in SI register and the DS already contains its segment address as it has been declared a global variable. The drive number is also specified and the interrupt is invoked. The interrupt service reads the contents of the block and places it in a buffer whose address was specified in dap.

Disk Partitioning

• Partition Table contains information pertaining to disk partitions.
• Partition Table is the first physical sector
Head = 0
Track/Cylinder = 0
Sec = 1 or LBA = 0
• Partition Table at CHS = 001 is also called MBR (Master Boot Record).

For More Visit

www.VUAnswer.com

Structure of Partitioning Table

• Total size of Partition Table is 512 bytes.

• First 446 bytes contains code which loads the boot block of active partition and is executed at Boot Time.
• Rest of the 66 bytes is the Data part.
• Last two bytes of the Data part is the Partition table signature.

File System for Each O.S.

• On a single disk there can be 4 different file systems and hence 4 different O.S.
• Each O.S. will have its individual partition on disk.
• Data related to each partition is stored in a 16-bytes chunk within the Data Part of Partition Table.

Structure of Data Part of P.T.

The data part can contain information about four different partitions for different Operating systems. Each partition information chunk is 16 bytes long and the last two bytes at the end of the partition table data part is the partition table signature whose value should be AA55 indicating that the code part contains valid executable code.

Primary Partition

• Partition defined in the MBR (Master Boot Record) are primary partition.
• Each Primary Partition contains information about its respective O.S.
• However if only one O.S. is to be installed then extended partitions.

Extended Partitions

However if a single operating system is to be kept for instance, then the disk can be divided into primary and extended partitions. Information about primary and extended partition is kept in the first physical block. The extended partition may again be divided into a number of partitions, information about further partitions will be kept in extended partition table which will be the first physical block within extended partition (i.e. it will not the first block of primary partition.). Moreover there can be extended partitions within extended partitions and such that in then end there are number of logical partitions this can go on till the last drive number in DOS.

Partition Table II

Here it can be seen that the first partition table maintains information about the primary and extended partitions. The second partition table similarly stores information about a logical and a extended partition within the previous extended partition. Similarly for each such extended partition there will be a partition table that stores information about the logical partition and may also contain information about any further extended partition. In this way the partition tables form a chain as depicted in the slide below. The last partition table within the chain contains just a single entry signifying the logical drive.

For More Visit

www.VUAnswer.com

Reading Extended Partition

Above is a listing of a simple program that reads the partition table using the extended 13H services. It then displays the contents of the data part of the partition table read. For this purpose it uses various data structures designed in reflection of the partition table and 16 bytes data entries within. The program uses recursion and calls the getpart() function recursively whenever it finds an extended partition to read the data within the extended partition table.

Get Drive Parameters

On Entry:
AH – 48
DL – Drive number
DS:SI – Address of result buffer
On Exit:
Carry Clear
AH – 0
DS:SI – result buffer
Carry Set
AH – Error Code

The partition table data entry also stores the CHS address of the starting block. But this address is left insignificant if a LBA enable disk is in question. However LBA address can be used in place of the CHS address, and in case CHS address is required it can be calculated if the total number of tracks, sectors and heads are known. To get the total number of tracks, sectors and head the above described service can be used.

File System Data Structures (LSN, BPB)

Type	Description
Word	Buffer Size, must be 26 or greater. *The caller sets this value to the maximum buffer size*. If the length of this buffer is less than 30, this functions does not return the pointer to the Enhanced Disk Drive structure (EDD). If the Buffer Size is 30 or greater on entry, it is set to exactly 30 on exit. If the Buffer Size is between 26 and 29, it is set to exactly 26 on exit. If the Buffer Size is less than 26 on entry an error is returned.
Word	Information Flags In the following table, a 1 bit indicates that the feature is available, a 0 bit indicates the feature is not available and will operate in a manner consistent with the conventional Int 13h interface.

Bit	Description
0	DMA boundary errors are handled transparently
1	The geometry supplied in bytes 8-12 is valid
2	Device is removable
3	Device supports write with verify
4	Device has change line support (bit 2 must be set)
5	Device is lockable (bit 2 must be set).
6	Device geometry is set to maximum, no media is present (bit 2 must be set). This bit is turned off when media is present in a removable media device.
7-15	Reserved, must be 0

For More Visit

www.VUAnswer.com

4	Double Word	Number of *physical* cylinders. This is 1 greater than the maximum cylinder number. Use Int 13h Fn 08h to find the *logical* number of cylinders.
8	Double Word	Number of *physical* heads. This is 1 greater than the maximum head number. Use Int 13h Fn 08h to find the *logical* number of heads.
12	Double Word	Number of *physical* sectors per track. This number is the same as the maximum sector number because sector addresses are 1 based. Use Int 13h Fn 08h to find the *logical* number of sectors per track.
16	Quad Word	Number of *physical* sectors. This is 1 greater than the maximum sector number.
24	Word	Number of bytes in a sector.
26	**Double Word**	**Pointer to Enhanced Disk Drive (EDD) configuration parameters. This field is only present if Int 13h, Fn 41h, CX register bit 2 is enabled. This field points to a temporary buffer which the BIOS may re-use on subsequent Int 13h calls. A value of FFFFh:FFFFh in this field means that the pointer is invalid.**

LSN (Logical Sector Number)

Boot Block has LSN = 0
• If the blocks are indexed from the boot block such that the boot block has index = 0,Then this index is called LSN.
• LSN is relative index from the start of logical drive, not the physical drive.

For fixed disk
Hidden Blocks =
No. of Sec/Track

LSN is also indexed like LBA the only difference is that LBA is the address relative to the start of physical drive (i.e. absolute), whereas LSN address is the address from the start of logical partition i.e relative.

As in the above example it can be noticed that the LBA = 0 is not the same as LSN=0. The LBA=0 block is the first block on disk. Whereas each logical partition has LSN=0 block which is the first block in logical drive and is not necessarily the first block on physical drive. Also notice the hidden blocks between the first physical block on each partition and its first LSN block. These hidden blocks are not used by the operating system for storing any kind of data.

Conclusion

• LBA is physical or absolute address.
• LSN is relative address with respect to the start of Logical Drive.

File System Data Structures

• BIOS Parameter Block (BPB)
• Drive Parameter Block (DPB)
• File Control Block (FCB)
• FAT 12, FAT 16, FAT 32
• Master File Table (MFT)

To understand the file systems of DOS and Windows the above given data structure should be understood which are used by the operating system for file management.

Anatomy of a FAT based file system

Starting block(s) is /are
the boot block(s), immediately after which the FAT (File allocation table) starts. A typical volume will contain two copies of FAT. After FAT the root directory is situated which contain information about the files and folders in the root directory. Whole of this area constitutes the systems area rest of the area is used to store user data and folders

For More Visit

www.VUAnswer.com

Clusters

• A cluster is a collection of contiguous blocks.
• User Data is divided into clusters
• Number of blocks within a cluster is in power of 2.
• Cluster size can vary depending upon the size of the disk.
• DOS has a built in limit of 128 blocks per cluster.
• But practically limit of 64 blocks per cluster has been established.
• We will learn more about the size of clusters, later.

BPB (BIOS Parameter Block)

• Situated within the Boot Block.
• Contains vital information about the file system.

BIOS parameter block is a data structure maintained by DOS in the boot block for each drive. The boot block is typically a 512 byte block which as seen the previous slides is the first logical block i.e. LSN = 0. It contains some code and data. The data part constitutes the BPB.

BPB (BIOS Parameter Block)

Byte Offset	Field Length	Meaning
0x0B	WORD	Bytes per Sector. The size of a hardware sector. Usually 512.
0x0D	BYTE	Sectors Per Cluster. The number of sectors in a cluster. The default cluster size for a volume depends on the disk size and the file system.
0x0E	WORD	Reserved Sectors. The number of sectors from the Partition Boot Sector to the start of the first file allocation table, including the Partition Boot Sector. The minimum value is 1.
0x10	BYTE	Number of file allocation tables (FATs). The number of copies of the file allocation table on the volume. Typically, the value of this field is 2.
0x11	WORD	Root Entries. The total number of file name entries that can be stored in the root folder of the volume.
0x13	WORD	Small Sectors. The number of sectors on the volume if the number fits in 16 bits (65535). For volumes larger than 65536 sectors, this field has a value of 0 and the Large Sectors field is used instead.
0x15	BYTE	Media Type. Provides information about the media being used. A value of 0xF8 indicates a hard disk.
0x16	WORD	Sectors per file allocation table (FAT). Number of sectors occupied by each of the file allocation tables on the volume.
0x18	WORD	Sectors per Track.
0x1A	WORD	Number of Heads.
0x1C	DWORD	Hidden Sectors.
0x20	DWORD	Large Sectors. If the Small Sectors field is zero, this field contains the total number of sectors in the volume. If Small Sectors is nonzero, this field contains zero..

0x24	BYTE	Physical Disk Number. This is related to the BIOS physical disk number. Floppy drives are numbered starting with 0x00 for the A disk. Physical hard disks are numbered starting with 0x80. The value is typically 0x80 for hard disks, regardless of how many physical disk drives exist, because the value is only relevant if the device is the startup disk.
0x25	BYTE	Current Head. Not used by the FAT file system. (Reserved)
0x26	BYTE	Signature. Must be either 0x27, 0x28 or 0x29 in order to be recognized by Windows.
0x27	4 bytes	Volume Serial Number. A unique number that is created when you format the volume.
0x2B	11 bytes	Volume Label.
0x36	8 bytes	System ID. Either FAT12 or FAT16, depending on the format of the disk.

File System Data Structures II (Boot block)

The LSN of the boot block is 0. The information contained within the BPB in boot block can be used to calculate the LSN of the block from where the user data starts. It can be simply calculated by adding the number of reserved sector, sectors occupied by FAT copies * number of FAT copies and the the number of blocks reserved for root dir.

Inside a Boot Block

• Contains Code and Data
jmp codepart
OSName
BIOS
Parameter Block
codepart:
• Boot Block executes at Booting time.

For More Visit

www.VUAnswer.com

Besides the LBA address a LSN address can also be used to address a block. If the LSN address is known the absread() function can be used to read a block and abswrite() can be used to write on it as described in the slide below where nsect is the number of sector to be read/written.

Reading/ Writing a Block

• absread( )
is used to read a block given its LSN
• abswrite( )
is used to write a block given its LSN
absread(int drive, int nsects, long lsec, void *buffer);
abswrite(int drive, int nsects, long lsec, void *buffer);

File System Data Structures III (DPB)

Besides the BPB another data structure can be used equivalently called the DPB (Drive parameter block). The operating system translates the information in BPB on disk into the DPB which is maintained main memory. This data structure can be accessed using the undocumented service 21H/32H. Its detail is shown in the slide below.

The DPB contains the information shown in the table below. This information can be derived from the BPB but is placed in memory in the form of DPB.

DPB (Drive Parameter Block)

Offset	Size	Description
00h	BYTE	Drive number (00h = A:, 01h = B:, etc)
01h	BYTE	Unit number within device driver
02h	WORD	Bytes per sector
04h	BYTE	Highest sector number within a cluster
05h	BYTE	Shift count to convert clusters into sectors
06h	WORD	Number of reserved sectors at beginning of drive
08h	BYTE	Number of FAT's
09h	WORD	Number of root directory entries
0Bh	WORD	Number of first sector containing user data
0Dh	WORD	Highest cluster number (number of data cluster +1)

DPB (Drive Parameter Block)

Offset	Size	Description
0Fh	WORD	number of sectors per FAT
11h	WORD	Sector number of first directory sector
13h	DWORD	Address of device driver header
17h	BYTE	Media ID byte
18h	BYTE	00h if disk accessed, FFh if not
19h	DWORD	Pointer to next DPB
1Dh	WORD	Cluster at which to start search for free space when writing, usually the last cluster allocated
1Fh	WORD	Number of free clusters on drive, FFFFh if not known

Root Directory, FAT12 File System

The DOS directory structure is a Tree like structure. The top most level of the tree being the root directory. The root directory contains files and folders. Each folder can contains more files and folders and so on it continues recursively.

File

- Is logically viewed as an organization of Data.
- Physically it can be a collection of clusters or blocks.
- O.S. needs to maintain information about the cluster numbers of which a file may be comprised of.

Control information about files are maintained in a data structure called the File control block (FCB). The FCB for each file is created and stored in the disk.

The root directory consists of FCBs for all the files and folders stored on the root directory. To obtain these FCBs, the portion on disk reserved for root directory can be read.

In the above two slides first the contents of DPB are read to find the start of the root directory. Using this block number the contents of root directory are read, as it can be seen they contain number of FCBs each containing information about a file within the directory.
The user data area is divided into clusters. The first cluster in user data area is numbered 2 in a FAT based systems. A cluster is not the same as block and also there are no system calls available which use the cluster number. All the system calls use the LSN address. If

the cluster number is known it should be converted into LSN to access the blocks within the cluster. Moreover all the information about file management uses the cluster number rather than the LSN for simplicity and for the purpose of managing large disk space. So here we devise a formula to convert the cluster number into LSN.

FAT12 File System II, FAT16 File System

The root directory contains a collection of FCBs. The FCB for the file in question is searched from where the first cluster of the file can be get.

After calculating the sector number for the cluster the contents of the file can be accessed by reading all the blocks within the cluster. In this way only the starting cluster will be read. If the file contains a number of cluster the subsequent clusters numbers within the file chain can be accessed from the FAT.

Larger File Contents

- Larger files would be comprised of numerous clusters.
- The first Cluster # can be read from FCB for rest of the Cluster, a chain is maintained within the FAT.

FAT12

- FAT is a simple table which contains cluster number of each file.
- FAT12 will have 12-bit wide entries and can have 2^12 entries maximum.
- Although some of these entries may be reserved.

Above slides show how a cluster chain for a file is maintained in the FAT. The first cluster number is in the FCB. Subsequent clusters will be accessed from the FAT using the previous cluster number as index to look up into the FAT for the next cluster number. A FAT theoretically will contain 2_n entries where n is 12 for FAT 12 and 16 for FAT16. But all the entries are not used some of the entries are reserved following slide shows its detail.

Unused FAT Entries

- Reserved Entries = FF0H ~ FF6H
- EOF value = FF7H ~ FFFH
- First Two Clusters = 0,1
- Free Cluster = 0
- Max. range of Cluster # = 2 ~ FEFH
- Total # of Clusters of FAT12 = FEEH

There can various volume with various sizes with FAT12 or FAT16. The number of entries for FAT 12 or FAT16 are limited then the question arises how can a certain volume with moderate space and another volume with large space can be managed by the same FAT system. The answer is that the number of entries might be same but the size of cluster may be different. The cluster size can vary from 512 bytes to 32K in powers of 2 depending upon the volume size.

FAT12 File System (Selecting a 12-bit entry within FAT12 System)

For More Visit

www.VUAnswer.com

There is no primitive data type of 12 bits. But the entries in 12 bit FAT are 12 bits wide. Three consecutive bytes in FAT 12 will contain two entries.

File Organization

Deleted Files

- 0xE5 at the start of file entry is used to mark the file as deleted.
- The contents of file still remain on disk.
- The contents can be recovered by placing a valid file name, character in place of E5 and then recovering the chain of file in FAT.
- If somehow the clusters used by deleted file has been overwritten by some other file, it cannot be recovered.

FAT32 File System

Let's now perform few more experiments to see how long file names are managed. Windows can have long file names up to 255 characters. For This purpose a file is created with a long file name

Long FileName

Volume in drive H is NEW VOLUME
Volume Serial Number is 8033-3F79

I n the above slide it can be noticed that the long file name is also stored with the FCBs. Also the fragments of Unicode strings in the long file name forms a chain. The first byte in the chain will be 0x01, the first byte of the next fragment will be 0x02 and so on till the last fragment. If the last fragment is the nth fragment starting from 0 such that n is between 0 and 25 the first byte of the last fragment will be given as ASCII of 'A' plus n.

Now lets move our discussion to FAT32. In theory the major difference between FAT 16 and FAT 32 is of course the FAT size. FAT32 evidently will contain more entries and can hence manage a very large disk whereas FAT16 can manage a space of 2 GB maximum practically.
Following slide shows the structure of the BPB for FAT32. Clearly there would be some additional information required in FAT32 so the structure for BPB used in FAT32 is different.

Fat32 BPB Structure

BPB_RsvdSecCnt	14	2	Number of reserved sectors in the Reserved region of the volume starting at the first sector of the volume. This field must not be 0. For FAT12 and FAT16 volumes, this value should never be anything other than 1. For FAT32 volumes, this value is typically 32. There is a lot of FAT code in the world "hard wired" to 1 reserved sector for FAT12 and FAT16 volumes and that doesn't bother to check this field to make sure it is 1. Microsoft operating systems will properly support any non-zero value in this field.

For More Visit

www.VUAnswer.com

BPB_NumFATs	16	1	The count of FAT data structures on the volume. This field should always contain the value 2 for any FAT volume of any type. Although any value greater than or equal to 1 is perfectly valid, many software programs and a few operating systems' FAT file system drivers may not function properly if the value is something other than 2. All Microsoft file system drivers will support a value other than 2, but it is still highly recommended that no value other than 2 be used in this field. The reason the standard value for this field is 2 is to provide redundancy for the FAT data structure so that if a sector goes bad in one of the FATs, that data is not lost because it is duplicated in the other FAT. On non-disk-based media, such as FLASH memory cards, where such redundancy is a useless feature, a value of 1 may be used to save the space that a second copy of the FAT uses, but some FAT file system drivers might not recognize such a volume properly.
BPB_RootEntCnt	17	2	For FAT12 and FAT16 volumes, this field contains the count of 32-byte directory entries in the root directory. For FAT32 volumes, this field must be set to 0. For FAT12 and FAT16 volumes, this value should always specify a count that when multiplied by 32 results in an even multiple of BPB_BytsPerSec. For maximum compatibility, FAT16 volumes should use the value 512.
BPB_TotSec16	19	2	This field is the old 16-bit total count of sectors on the volume. This count includes the count of all sectors in all four regions of the volume. This field can be 0; if it is 0, then BPB_TotSec32 must be non-zero. For FAT32 volumes, this field must be 0. For FAT12 and FAT16 volumes, this field contains the sector count, and BPB_TotSec32 is 0 if the total sector count "fits" (is less than 0x10000).
BPB_Media	21	1	0xF8 is the standard value for "fixed" (non-removable) media. For removable media, 0xF0 is frequently used. The legal values for this field are 0xF0, 0xF8, 0xF9, 0xFA, 0xFB, 0xFC, 0xFD, 0xFE, and 0xFF. The only other important point is that whatever value is put in here must also be put in the low byte of the FAT[0] entry. This dates back to the old MS-DOS 1.x media determination noted earlier and is no longer usually used for anything.
BPB_FATSz16	22	2	This field is the FAT12/FAT16 16-bit count of sectors occupied by ONE FAT. On FAT32 volumes this field must be 0, and BPB_FATSz32 contains the FAT size count.
BPB_SecPerTrk	24	2	Sectors per track for interrupt 0x13. This field is only relevant for media that have a geometry (volume is broken down into tracks by multiple heads and cylinders) and are visible on interrupt 0x13. This field contains the "sectors per track" geometry value.
BPB_NumHeads	26	2	Number of heads for interrupt 0x13. This field is relevant as discussed earlier for BPB_SecPerTrk. This field contains the one based "count of heads". For example, on a 1.44 MB 3.5-inch floppy drive this value is 2.
BPB_HiddSec	28	4	Count of hidden sectors preceding the partition that contains this FAT volume. This field is generally only relevant for media visible on interrupt 0x13. This field should always be zero on media that are not partitioned. Exactly what value is appropriate is operating system specific.
BPB_TotSec32	32	4	This field is the new 32-bit total count of sectors on the volume. This count includes the count of all sectors in all four regions of the volume. This field can be 0; if it is 0, then BPB_TotSec16 must be non-zero. For FAT32 volumes, this field must be non-zero. For FAT12/FAT16 volumes, this field contains the sector count if BPB_TotSec16 is 0 (count is greater than or equal to 0x10000).

Fat12 and Fat16 Structure Starting at Offset 36

Name	Offset (byte)	Size (bytes)	Description
BS_DrvNum	36	1	Int 0x13 drive number (e.g. 0x80). This field supports MS-DOS bootstrap and is set to the INT 0x13 drive number of the media (0x00 for floppy disks, 0x80 for hard disks). **NOTE:** This field is actually operating system specific.
BS_Reserved1	37	1	Reserved (used by Windows NT). Code that formats FAT volumes should always set this byte to 0.
BS_BootSig	38	1	Extended boot signature (0x29). This is a signature byte that indicates that the following three fields in the boot sector are present.
BS_VolID	39	4	Volume serial number. This field, together with BS_VolLab, supports volume tracking on removable media. These values allow FAT file system drivers to detect that the wrong disk is inserted in a removable drive. This ID is usually generated by simply combining the current date and time into a 32-bit value.
BS_VolLab	43	11	Volume label. This field matches the 11-byte volume label recorded in the root directory. **NOTE:** FAT file system drivers should make sure that they update this field when the volume label file in the root directory has its name changed or created. The setting for this field when there is no volume label is the string "`NO NAME    `".
BS_FilSysType	54	8	One of the strings "`FAT12   `", "`FAT16   `", or "`FAT     `". **NOTE:** Many people think that the string in this field has something to do with the determination of what type of FAT—FAT12, FAT16, or FAT32—that the volume has. This is not true.

There can be different volumes with different volume sizes. The device driver for file handling would require knowing the FAT size. The following slide illustrates an algorithm that can be used to determine the FAT size in use after reading the BPB.

FAT32 Structure Starting at Offset 36

Name	Offset (byte)	Size (bytes)	Description
BPB_FATSz32	36	4	This field is only defined for FAT32 media and does not exist on FAT12 and FAT16 media. This field is the FAT32 32-bit count of sectors occupied by ONE FAT. BPB_FATSz16 must be 0.
BPB_ExtFlags	40	2	This field is only defined for FAT32 media and does not exist on FAT12 and FAT16 media. Bits 0-3 -- Zero-based number of active FAT. Only valid if mirroring is disabled. Bits 4-6 -- Reserved. Bit 7 -- 0 means the FAT is mirrored at runtime into all FATs. -- 1 means only one FAT is active; it is the one referenced in bits 0-3. Bits 8-15 -- Reserved.
BPB_FSVer	42	2	This field is only defined for FAT32 media and does not exist on FAT12 and FAT16 media. High byte is major revision number. Low byte is minor revision number. This is the version number of the FAT32 volume. This supports the ability to extend the FAT32 media type in the future without worrying about old FAT32 drivers mounting the volume. This document defines the version to 0:0. If this field is non-zero, back-level Windows versions will not mount the volume. **NOTE:** Disk utilities should respect this field and not operate on volumes with a higher major or minor version number than that for which they were designed. FAT32 file system drivers must check this field and not mount the volume if it does not contain a version number that was defined at the time the driver was written.
BPB_RootClus	44	4	This field is only defined for FAT32 media and does not exist on FAT12 and FAT16 media. This is set to the cluster number of the first cluster of the root directory, usually 2 but not required to be 2. **NOTE:** Disk utilities that change the location of the root directory should make every effort to place the first cluster of the root directory in the first non-bad cluster on the drive (i.e., in cluster 2, unless it's marked bad). This is specified so that disk repair utilities can easily find the root directory if this field accidentally gets zeroed.
BPB_FSInfo	48	2	This field is only defined for FAT32 media and does not exist on FAT12 and FAT16 media. Sector number of FSINFO structure in the reserved area of the FAT32 volume. Usually 1. **NOTE:** There will be a copy of the FSINFO structure in BackupBoot, but only the copy pointed to by this field will be kept up to date (i.e., both the primary and backup boot record will point to the same FSINFO sector).
BPB_BkBootSec	50	2	This field is only defined for FAT32 media and does not exist on FAT12 and FAT16 media. If non-zero, indicates the sector number in the reserved area of the volume of a copy of the boot record. Usually 6. No value other than 6 is recommended.
BPB_Reserved	52	12	This field is only defined for FAT32 media and does not exist on FAT12 and FAT16 media. Reserved for future expansion. Code that formats FAT32 volumes should always set all of the bytes of this field to 0.
BS_DrvNum	64	1	This field has the same definition as it does for FAT12 and FAT16 media. The only difference for FAT32 media is that the field is at a different offset in the boot sector.
BS_Reserved1	65	1	This field has the same definition as it does for FAT12 and FAT16 media. The only difference for FAT32 media is that the field is at a different offset in the boot sector.
BS_BootSig	66	1	This field has the same definition as it does for FAT12 and FAT16 media. The only difference for FAT32 media is that the field is at a different offset in the boot sector.
BS_VolID	67	4	This field has the same definition as it does for FAT12 and FAT16 media. The only difference for FAT32 media is that the field is at a different offset in the boot sector.
BS_VolLab	71	11	This field has the same definition as it does for FAT12 and FAT16 media. The only difference for FAT32 media is that the field is at a different offset in the boot sector.
BS_FilSysType	82	8	Always set to the string "FAT32 ". Please see the note for this field in the FAT12/FAT16 section earlier. This field has nothing to do with FAT type determination.

FAT32 File System II

Fat32 Entry

• Each entry is of 32-bits size but only lower 28-bits are used.
• Higher 4-bits are not tempered.
• While reading higher 4-bits are ignored.
• While writing higher 4-bits are not changed.

FAT 32 Byte Directory Entry Structure

Name	Offset (byte)	Size (bytes)	Description
DIR_Name	0	11	Short name.
DIR_Attr	11	1	File attributes: ATTR_READ_ONLY 0x01 ATTR_HIDDEN 0x02 ATTR_SYSTEM 0x04 ATTR_VOLUME_ID 0x08 ATTR_DIRECTORY 0x10 ATTR_ARCHIVE 0x20 ATTR_LONG_NAME ATTR_READ_ONLY | ATTR_HIDDEN | ATTR_SYSTEM | ATTR_VOLUME_ID The upper two bits of the attribute byte are reserved and should always be set to 0 when a file is created and never modified or looked at after that.
DIR_NTRes	12	1	Reserved for use by Windows NT. Set value to 0 when a file is created and never modify or look at it after that.
DIR_CrtTimeTenth	13	1	Millisecond stamp at file creation time. This field actually contains a count of tenths of a second. The granularity of the seconds part of DIR_CrtTime is 2 seconds so this field is a count of tenths of a second and its valid value range is 0-199 inclusive.
DIR_CrtTime	14	2	Time file was created.
DIR_CrtDate	16	2	Date file was created.
DIR_LstAccDate	18	2	Last access date. Note that there is no last access time, only a date. This is the date of last read or write. In the case of a write, this should be set to the same date as DIR_WrtDate.
DIR_FstClusHI	20	2	High word of this entry's first cluster number (always 0 for a FAT12 or FAT16 volume).
DIR_WrtTime	22	2	Time of last write. Note that file creation is considered a write.
DIR_WrtDate	24	2	Date of last write. Note that file creation is considered a write.

Anatomy of FAT32 based system differs from FAT16 based systems significantly as explained by the slide below.

Fat32 File System

Reserved Blocks

• No fixed space reserved for root

For More Visit

www.VUAnswer.com

directory.
• FCB of root directory are saved in a cluster and the cluster # for root directory is saved in BPB as discussed earlier.

In reflection of the anatomy of FAT32 based system the method used to translate the cluster # into LSN also varies. The following formula is used for this purpose.

Starting Sector # for a Cluster

Starting Sector = Reserved Sect. + FatSize * FatCopies + (cluster # - 2) * size of cluster

In the FAT32 there is another special reserved block called FSInfo sector. The block contains some information required by the operating system while cluster allocation/deallocation to files. This information is also critical for FAT16 based systems. But in FAT12 and 16 this information is calculated when ever required. This calculation at the time of allocation is not feasible in FAT32 as the size of FAT32 is very large and such calculations will consume a lots of time, so to save time this information is stored in the FSInfo block and is updated at the time of allocation/deallocation.

FAT32 FSInfo Sector Structure and Backup Boot Sector

Name	Offset (byte)	Size (bytes)	Description
FSI_LeadSig	0	4	Value 0x41615252. This lead signature is used to validate that this is in fact an FSInfo sector.
FSI_Reserved1	4	480	This field is currently reserved for future expansion. FAT32 format code should always initialize all bytes of this field to 0. Bytes in this field must currently never be used.
FSI_StrucSig	484	4	Value 0x61417272. Another signature that is more localized in the sector to the location of the fields that are used.
FSI_Free_Count	488	4	Contains the last known free cluster count on the volume. If the value is 0xFFFFFFFF, then the free count is unknown and must be computed. Any other value can be used, but is not necessarily correct. It should be range checked at least to make sure it is <= volume cluster count.
FSI_Nxt_Free	492	4	This is a hint for the FAT driver. It indicates the cluster number at which the driver should start looking for free clusters. Because a FAT32 FAT is large, it can be rather time consuming if there are a lot of allocated clusters at the start of the FAT and the driver starts looking for a free cluster starting at cluster 2. Typically this value is set to the last cluster number that the driver allocated. If the value is 0xFFFFFFFF, then there is no hint and the driver should start looking at cluster 2. Any other value can be used, but should be checked first to make sure it is a valid cluster number for the volume.
FSI_Reserved2	496	12	This field is currently reserved for future expansion. FAT32 format code should always initialize all bytes of this field to 0. Bytes in this field must currently never be used.

New Technology File System (NTFS)

The following slide shows the anatomy of an NTFS based system. The FAT and root

directory has been replaced by the MFT. It will generally have two copies the other copy will be a mirror image of the original. Rests of the blocks are reserved for user data. In the middle of the volume is a copy of the first 16 MTF record which are very important to the system.

NTFS General Boot Sector Structure

Byte Offset	Field Length	Field Name
0x00	3 bytes	Jump Instruction
0x03	LONGLONG	OEM ID
0x0B	25 bytes	BPB
0x24	48 bytes	Extended BPB
0x54	426 bytes	Bootstrap Code
0x01FE	WORD	End of Sector Marker

For More Visit

www.VUAnswer.com

The first 16 entries of the MFT are reserved. Rests of the entries are used for user files. There is an entry for each file in the MFT. There can be difference in the way a file is managed depending upon the size of the file.

MFT Entry Details

Attribute Type	Description
Standard Information	Includes information such as timestamp and link count.
Attribute List	Lists the location of all attribute records that do not fit in the MFT record.
File Name	A repeatable attribute for both long and short file names. The long name of the file can be up to 255 Unicode characters. The short name is the 8.3, case-insensitive name for the file. Additional names, or hard links, required by POSIX can be included as additional file name attributes.
Security Descriptor	Describes who owns the file and who can access it.
Data	Contains file data. NTFS allows multiple data attributes per file. Each file typically has one unnamed data attribute. A file can also have one or more named data attributes, each using a particular syntax.
Object ID	A volume-unique file identifier. Used by the distributed link tracking service. Not all files have object identifiers.
Logged Tool Stream	Similar to a data stream, but operations are logged to the NTFS log file just like NTFS metadata changes. This is used by EFS.
Reparse Point	Used for volume mount points. They are also used by Installable File System (IFS) filter drivers to mark certain files as special to that driver.
Index Root	Used to implement folders and other indexes.
Index Allocation	Used to implement folders and other indexes.
Bitmap	Used to implement folders and other indexes.
Volume Information	Used only in the $Volume system file. Contains the volume version.
Volume Name	Used only in the $Volume system file. Contains the volume label.

MFT System Entries

System File	File Name	MFT Record	Purpose of the File
Master file table	$Mft	0	Contains one base file record for each file and folder on an NTFS volume. If the allocation information for a file or folder is too large to fit within a single record, other file records are allocated as well.
Master file table 2	$MftMirr	1	A duplicate image of the first four records of the MFT. This file guarantees access to the MFT in case of a single-sector failure.
Log file	$LogFile	2	Contains a list of transaction steps used for NTFS recoverability. Log file size depends on the volume size and can be as large as 4 MB. It is used by Windows NT/2000 to restore consistency to NTFS after a system failure.
Volume	$Volume	3	Contains information about the volume, such as the volume label and the volume version.
Attribute definitions	$AttrDef	4	A table of attribute names, numbers, and descriptions.
Root file name index	$	5	The root folder.
Cluster bitmap	$Bitmap	6	A representation of the volume showing which clusters are in use.
Boot sector	$Boot	7	Includes the BPB used to mount the volume and additional bootstrap loader code used if the volume is bootable.
Bad cluster file	$BadClus	8	Contains bad clusters for the volume.
Security file	$Secure	9	Contains unique security descriptors for all files within a volume.
Upcase table	$Upcase	10	Converts lowercase characters to matching Unicode uppercase characters.
NTFS extension file	$Extend	11	Used for various optional extensions such as quotas, reparse point data, and object identifiers.
		12–15	Reserved for future use.

For NTFS simply the following formula will be used to translate the sector number into cluster number.

Determining the Sector # from Cluster

Sector # = Cluster # * Sector Per Cluster

The following slides explain how the NTFS volume can be accessed in DOS. Normally it can not be accessed if the system has booted in DOS as the DOS device drivers do not understand NTFS.7

Accessing NTFS volume in DOS

• NTFS volume can not be accessed in DOS using DOS based function like absread( ) etc.

• DOS device drivers does not understand

the NTFS data structures like MFT etc.
• If NTFS volume is accessed in DOS, it will fire the error of Invalid Media.

How to Access NTFS volume using BIOS Functions

• If the system has booted in DOS then a NTFS volume can be accessed by an Indirect Method, using BIOS functions..
• This technique makes use of physical addresses.
• Sector can be accessed by converting their LSN into LBA address and then using the LBA address in extended BIOS functions to access the disk sectors.

The above program uses the DPB to reach the clusters of a file. The getDPB() function gets the far address of the DPB. Using this address the drive parameters are used to determine the location of FAT and root directory. The file is firstly searched in the root directory through sequential search. If the file name and extension is found the first cluster number is used to look up into the FAT for subsequent clusters. The particular block containing the next cluster within the FAT is loaded and the entry is read, similarly the whole chain is traversed till the end of file is encountered.

Disk Utilities

Format
• Low Level Format
-- sets the block size.
-- sets the Initial values in the block.
-- indexes the block for optimal usage.
-- can be accomplished using BIOS routines for small disks or extended BIOS services for larger disks.
• Quick Format
-- initializes the data structures for file management.
-- initializes and sets the size of FAT, root directory etc, according to the drive size.
-- initializes the data in boot block and places appropriate boot strap code for the boot block.

Disk Partitioning Software

•Write the code part of partition table to appropriately load the Boot Block of active partition in primary partition table.
• Places data in the partition table regarding primary and extended partitions.

• As per specification of the user assigns a appropriate size to primary and extended partition by modifying their data part.

Scan Disk

Surface Scan for Bad Sectors

• It attempts to write a block.
• After write it reads back the block contents.
• Performs the CRC test on data read back.
• If there is an error then the data on that block is not stable the cluster of that block should be marked bad.
• The cluster is marked bad by placing the appropriate code for bad cluster so that they may not be allocated to any file.

Lost Chains

• The disk scanning software may also look for lost chains.
• Lost chains are chains in FAT which apparently don't belong to any file.
• They may occur due to some error in the system like power failure during deletion process.

Looking for Lost Chains

• For each file entry in the directory structure its chain in FAT is traversed.
• All the cluster in the file are marked.
• When done with all the files and folders, if some non-zero and non-reserved clusters are left then they belong to some lost chains.
• The lost chains are firstly discretely identified and then each chain can either be restored to newly named files or can be deleted.

Cross References

• If a cluster lie in more than one file chain, then its said to be Cross Referenced.
• Cross references can pose great problems.
• Cross references can be detected easily by traversing through the chain of all files and marking the cluster # during traversal.
• If a cluster is referenced more than once then it indicates a cross reference.
• To solve the problem only one reference should be maintained.

For More Visit

www.VUAnswer.com

Defragmenter

• Disk fragmentation is unwanted.
• Fragmentation means that clusters of a same file are not contiguously placed, rather they are far apart, increasing seek time hence access time.
• So its desirable that files clusters may be placed contiguously, this can be done by compaction or defragme ntation.
• Defragmentation Software reserves space for each file in contiguous block by moving the data in clusters and readjusting.
• As a result of defragmentation the FAT entries will change and data will move from one cluster to other localized cluster to reduce seek time.
• Defragmentation has high computation cost and thus cannot be performe d frequently.

File Restoration

• FAT structure provides the possibility of recovering a file after deletion, if its clusters were contiguous and have not been over-written.
• DOS perform file deletion by placing 0xE5 at the first byte of it FCB entry and placing 0's (meaning available) in the entries for the file clusters in the FAT.
• Two task should be performed successfully to undelete a file
-- Replacing the 0xE5 entry in FCB by a valid file name character.
-- placing the appropriate values in FAT for representation of file cluster chain.
• If any one of the above cannot be done then the file cannot be fully recovered.

Memory Management

Memory Management

• Understanding of the data structures and techniques used for memory management.
• Study of the overall memory areas used by operating system and applications.

The following slide shows the memory map of the first 1MB of RAM. The first 640KB is called conventional RAM and the higher 384KB is called system memory. Some of the memory areas are reserved for special purposes as described by the slide rest is user area where user application can reside.

In higher processors, the main memory may be greater than 1MB. In this slide it shows that the memory portion higher than 1MB is called extended memory and some unused portion in system memory is called the expanded memory.

Expanded Memory

• also called EMS
• can be accessed using a driver called EMM386.EXE
• this driver allows the use of unused memory within system memory.

Extended Memory

• also called XMS
• can be accessed by installing the driver HIMEM.SYS
• this driver enable the extended memory by shifting from Real to Protected Mode.

Dual Modes in Higher PCs

Higher PCs can operate in two modes
• REAL MODE

For More Visit

www.VUAnswer.com

• PROTECTED MODE

Real Mode

• PCs initially boots up in Real Mode. It may be shifted to protected mode during the booting process using drivers like HIMEM.SYS
• Only first 1 MB of RAM can be accessed in Real Mode.
• The Real Mode address is a 20-bit address, stored and represented in the form of Segment : Offset
• OS like DOS has a memory management system in reflection of the Real Mode.

Protected Mode

• PC has to be shifted to Protected Mode if originally boots in Real Mode.
• In Protected Mode whole of the RAM is accessible that includes the Conventional, Expanded and Extended Memories.
• OS like Windows has a memory management system for Protected Mode.
• A privilege level can be assigned to a memory area restricting its access.

Memory Management in DOS

• DOS uses the conventional memory first 640 KB for its memory management.
• Additional 64 KB can be utilized by installing EMM386.EXE and additional 64 KB in the start of extended memory by installing HIMEM.SYS
• Smallest allocatable unit in DOS is a Paragraph, not a Byte.

Paragraph

• Whenever memory is to be allocated DOS allocates memory in form of Paragraph.
• A Paragraph can be understood from the following example
consider two Physical Addresses
1234 H : 0000 H
1235 H : 0000 H
• Note there is a difference of 1 between the Segment address.
• Now lets calculate the Physical address
12340 H
12350 H
Difference = 10 H
• A difference of 1 H in Segment address cause a difference of 10 H in Physical address.
• DOS loader assign a segment address whenever memory area is allocated, hence a change of 1 in Segment address will impart a difference of 16 D | 10 H in physical address.

Data Structures for Memory Management

• DOS makes use of various Data Structures for Memory Management:
• MCB (Memory Control Block)
• EB (Environment Block)
• PSP (Program Segment Prefix)

For More Visit

www.VUAnswer.com

MCB or Arena Header

• MCB is used to control an allocated block in memory.
• Every allocated block will have a MCB before the start of block.
• MCB is a 16-bytes large structure.

Size	Offset	
Byte	0	Contains 'M' if the MCB controls allocated memory and 'Z' if it controls free space.
Word	1	Contains the Segment address of the PSP and the program controlled by MCB.
Word	3	Contains number of Paragraphs controlled by the MCB.
Byte [11]	5	Reserved or contains the program name in case of higher versions of DOS.

Environment Block

• Contains Environment information like Environment variables and file paths for that program

PSP

• is situated before the start of a process.
• contains control information like DTA (Disk Transfer Area) and command line parameters.

The following slide shows that two MCBs are allocated for each program typically. The first MCB controls the Environment Block the next MCB controls the PSP and the program. If this is the last program in memory then the MCB after the program has 'Z' in its first byte indicating that it is the last MCB in the chain.

All the MCB forms a chain. If the address of first MCB is known the segment address of next MCB can be determined by adding the number of paragraph controlled by MCB + ! into the segment address of the MCB. Same is true for all MCBs and hence the whole chain can be traversed.

How to Access the Start of Chain

• An documented service can be used to obtain the address of the first MCB.
• Service 21H/52H is used for this purpose.
• This service returns
The address of DOS internal data structures in ES : BX
• 4-bytes behind the address returned lies the far address of the first MCB in memory.
• Using this address and hence traversing through the chain of MCBs using the information within MCBs.

The above slide shows how service 21H/52H is used to get the address of first MCB in memory.

In the following slide the dump of the first MCB is taken. 'M' in the first byte at the location read indicates the placement of MCB at this location. The address of next MCB can be calculated by adding the number of paragraphs controlled by MCB + 1 into the segment address. Using this method all the MCBs in memory are traversed till the last MCB with first byte 'Z' is encountered.

Non-Contiguous memory allocation

Non-Contiguous Allocation

• Earlier Operating System like DOS has contiguous memory management system i.e. a program cannot be loaded in memory if a contiguous block of memory is not available to accommodate it.
• 80286 and higher processors support non-contiguous allocation.

• 80286 support Segmentation in Protected Mode, i.e. a process is subdivided into segment of variable size and each segment or few segments of the process can be placed anywhere in memory
• 80386 and higher processors also support Paging, i.e. a Process may be divided into fixed size Pages and then only few pages may be loaded any where in memory for Process Execution.
• The key to such non-contiguous allocation systems is the addressing technique.

Address Translation

• In Protected Mode the direct method of seg * 10H + offset for Logical to Physical address translation is discarded and an indirect method is adopted.

Selectors

• In Protected Mode the Segment Registers are used as Selector.
• As the name suggest they are used to select a descriptor entry from some Descriptor Table.

Descriptor

• A Descriptor describes a Memory Segment by storing attributes related to a Memory Segment.
• Significant attributes of a Memory Segment can be its base (starting) address, it length or limit and its access rights.

Descriptor Table

• GDT: Global Descriptor Table
• LDT: Local Descriptor Table
• IDT: Interrupt Descriptor Table
• GDT and LDT can have up to 8192 entries, each of 8-bytes width.
• IDT can have up to 256 entries

Address translation in Protected mode

Selector

• A Selector is called a Selector because it acts as an index into the Descriptor Table to select a GDT or LDT entry.

Address Translation in Protected Mode

• All the tables are maintained in Main Memory.
• Segment Registers are used as Selectors.
• The Descriptor Entry selected from the Descriptor Table is placed in a hidden cache to optimize address translation.

Address Translation in Protected Mode

•Whenever a Selector is assigned a new value, the hardware looks up into the Descriptor Table and loads the Base Address, Limit and Access Rights into the hidden cache.
•Whenever an instruction is issued the address referred is translating into Physical address using the effective Offset within the instruction and the Base Address in the corresponding Segment Cache, e.g.
movAX, [1234H]
effective offset = 1234H
base = base within the cache of DS
abs. address = base +1234H
Or in instruction
movDL, [EBP]

For More Visit

www.VUAnswer.com

effective offset address = EBP
base address = base address in cache of SS register
abs. address = base address + EBP
• Hence the absolute address cannot be calculated directly from the Segment address value.

Control Register

• 80386 and above have 4 Control Registers CR0 ~ CR3.
• These Control Registers are used for conveying certain control information for Protected Mode Addressing and Co-Processors.
• Here we will illustrate only the least significant bit of CR0.
• The least significant bit of CR0 is PE-bit which can be set to enable Protected Mode Addressing and can be cleared to enter Real Mode.

Moving to Protected Mode

• Protected Mode can be entered by setting the PE bit of CR0, but before this some other initialization must be done. The following steps accomplish the switching from Real to Protected Mode correctly.
1. Initialize the Interrupt Descriptor Table, so it contains valid Interrupt gates for at least the first 32 Interrupt type numbers. The IDT may contain up to 256, 8-byte interrupt gates defining all 256 interrupt types.
2. Initialize the GDT, so it contains a NULL Descriptor, at Descriptor 0 and valid Descriptor for at least one Data and one Stack.
3. Switch to Protected by setting the PE-bit in CR0.
4. Perform a IntraSegment (near) JMP to flush the Internal Pre-fetch Queue.
5. Load all the Data Selectors (Segment Registers) with their initial Selectors Values.
6. The 80386 is now in Protected Mode.

Viruses

• Viruses are special program having ability to embed themselves in a system resources and there on propagate themselves.

State of Viruses

• Dormant State: A Virus in dormant state has embedded itself within and is observing system activities.
• Activation State: A Virus when activated would typically perform some unwanted tasks causing data loss. This state may triggered as result of some event.
• Infection State: A Virus is triggered into this state typically as a result of some disk operation. In this state, the Virus will infect some media or file in order to propagate itself.

Viruses

Types of Viruses

• Partition Table Virus
• Boot Sector Virus
• File Viruses

How Partition Table Virus Works

• The Partition Table Code is executed at boot time to choose the Active Partition.
• Partition Table Viruses embed themselves in the Partition Table of the disk.
• If the Virus Code is large and cannot be accommodated in the Code Part of 512-bytes Partition Table block then it may also use other Physically Addressed Blocks to reside itself.
• Hence at Boot time when Partition Table is to be executed to select the Active Partition, the virus executes. The Virus when executed loads itself in the Memory, where it can not be reached by the OS and then executes the original Partition Table Code (stored in some other blocks after infection) so that the system may be booted

properly.
•When the system boots the Virus will be resident in memory and will typically intercept 13H (the disk interrupt).
•Whenever a disk operation occurs int 13H occurs. The Virus on occurrence of 13H checks if removable media has been accessed through int 13H. If so then it will copy its code properly to the disk first Physical Block (and other blocks depending upon size of Virus Code). The removable disk is now infected.
• If the disk is now removed and is then used in some other system, the Hard Drive of this system will not be infected unless the system is booted from this disk. Because only on booting from this removable disk its first physical block will get the chance to be executed.

How Partition Table Virus Loads itself

• The transient part of Command.Com loads itself such that its last byte is loaded in the last byte of Conventional Memory. If somehow there is some Memory beyond Command.Com's transient part it will not be accessible by DOS.
• At 40:13H a word contains the amount of KBs in Conventional Memory which is typically 640.
• If the value at 40:13H is somehow reduced to 638 the transient part of Command.Com will load itself such that its last byte is loaded at the last byte of 638KB mark in Conventional RAM.
• In this way last 2KB will be left unused by DOS. This amount of memory is used by the Virus according to its own size.

How Boot Sector Virus Works

• Boot Sector also works in almost the same pattern, the only difference is that it will embed itself within the Boot Block Code.

File Viruses

• Various Viruses embeds themselves in different executable files.
• Theoretically any file that can contain any executable code, a Virus can be embedded into it. i.e. .COM, .EXE are executable files so Viruses can be embedded into them, Plain Text Files, Plain Bitmap Files are pure data and cannot be executed so Viruses cannot be actively embedded into them, and even if they are somehow embedded they will never get a chance to execute itself.

COM File

• COM File is a mirror image of the program code. Its image on disk is as it is loaded into the memory.
• COM Files are single segment files in which both Code and Data resides.
• COM File will typically have a Three Bytes Near Jump Instruction as the first instruction in program which will transfer the execution to the Code Part of the Program.

How COM File Virus Infects Files

• A COM File Virus if resident may infect COM Files on execution.
• Typically COM File Virus will Interrupt 21H Service 4B. This Service is used to load a Program.
• Whenever a Program is to be Loaded int 21H Service # 4BH is used to Load a Program. The Virus if resident will check the parameters of this Service to get the file path. If the File is .COM File then the Virus appends itself to the

For More Visit

www.VUAnswer.com

file and tempers with the first 3-bytes of .COM File so that the execution branches to the Virus Code when the program is executed.

How COM Virus Loads Itself

• When a file is Loaded in Memory it will occupy a number of Paragraphs controlled by some MCB.
• If the file is infected the Virus is also loaded within the Memory Area allocated to the Program.
• In this case the Virus does not exist as an Independent Program as it does not have its own PSP. If the Program is terminated the Virus Code will also be unloaded with the program. The Virus will try to attain an Independent Status for which it needs to relocate itself and create its own PSP and MCB in Memory.
• When the program runs the Virus Code executes first. The Virus creates an MCB, defines a new PSP initializes the PSP and relocates itself, updates the last MCB, so that it can exist as an Individual Program, and then transfers the execution back to the Original Program Code.
• Now if the Original Program Terminates the Virus will still remain resident.

EXE File Viruses

• The EXE File Viruses also works the same way in relocating themselves.
• The main difference in COM File and DOS EXE File is that the COM File starts its execution from the first instruction, whereas the entry point of execution in EXE File can be anywhere in the Program.
• The entry point in case of EXE File is tempered by the Virus which is stored in a 27-byte header in EXE File.

For More Visit

www.VUAnswer.com

Detection

• Viruses can be detected by searching for their Signature in Memory or Executable Files.
• Signature is a binary subset of Virus Code. It is a part of Virus Code that is unique for that particular Virus only and hence can be used to identify the Virus
• Signature for a Virus is selected by choosing a unique part of its Code. To find a Virus this Code should be searched in memory and in files. If a match is found then the system is infected.

Removal

Partition Table & Boot Sector Viruses

• Partition Table and Boot Sector Viruses can be removed by re-writing the Partition Table or Boot Sector Code.
• If the Virus is resident it may exhibit stealth i.e. prevent other programs from writing on Partition Table or Boot Sector by intercepting int 13H
• In case it's a stealth Virus the system should be booted from a clean disk will not give the Virus any chance to execute or load itself.

File Viruses

• If the Virus size is known Viruses can be removed easily from file.
• Firstly, the original value of first 3-bytes in case of COM File or the entry point in case of EXE should be restored.
• The appended portion of Virus can be removed by coping the contents of original file into a temporary file.
•The Virus Code is not copied.
• The original file is then deleted and the temporary file is renamed as the original file.

Virtual University	CS609-System Programming Solved MCQ(S) From FinalTerm Papers BY Arslan	Jan 14, 2018
V-U	For Updated Files Visit Our Site : Www.VirtualUstaad.blogspot.com	Updated.



Final-Term Papers Solved MCQS with Reference

1. DOS command _____ gives the status of the memory and also points out which memory area occupied by which process.

- **mem/d** **PG # 13**
- mem/e
- mem/m
- None of the Given

2.To store each character in keyboard buffer ____ bytes are required.

- **2** **PG # 54**
- 4
- 6
- 8

بری صحبت سے تنہائی بہتر ہے اور تنہائی سے نیک صحبت بہتر ہے

3.The interval timer is used to divide an input frequency.

- **True** **PG #68**
- False

4.The printer interface uses the _____.

- IRQ 6
- **IRQ 7** **PG # 95**
- IRQ 1
- None of given options

5.You can send _____ bits in parallel communication between two PC's at a time.

- 2
- 4
- 6
- **8**

6.In UART the line control register signify the _____.

- Word size and length of stop bits
- parity check and parity type
- Control bit
- **All of given** **PG # 114**

اللہ کا خوف سب سے بڑی دانائی ہے

7.The bit _____ of Line control register in UART, if set indicates that the DLL and DLM will act as the divisor register.

- 1
- 3
- 5
- **7** **PG # 114**

8.Int 14H _____ can be used to set the line parameter of the UART or COM port.

- **Service # 0** **PG #119**
- Service # 1
- Service # 2
- Service # 3

9._____ is used to read time from RTC.

- **1A/02H** **PG # 137**
- 1A/03H
- 1A/04H
- 1A/05H

10.The _____ signals are used by various devices to request a DMA operation.

- **DREQ** **PG #186**
- REQ
- DRE
- None of the given

ایماندار کو غصہ دیر سے آتا ہے اور جلدی دور ہو جاتا ہے

11.DMA works in _____ modes.

- 2
- **3** **PG #188**
- 5
- 7

DMA Modes Block Transfer, Single Transfer, Demand Transfer

12.DMA Request register can be used to simulate a DMA request through software in case of _____.

- I/O to memory transfer
- None of the given
- **Memory to memory transfer** **PG #192**
- Memory to I/O transfer

13._____ are the circular division of the disk.

- **Tracks** **PG #201**
- Sectors
- Blocks
- None of given

Tracks are the circular division of the disk and the sectors are the longitudinal division of the disk

14.Blocks are _____.

- **Sectors per track** **PG #202**
- Track per sector
- Cylinder per track
- None of the given

زندگی میں کامیابی کا یہی راز ہے کہ پریشانیوں سے پریشان مت بنو

15.Last ____ bytes of data part in partitioning table are the partition table signature.

- **2** **PG #218**
- 4
- 6
- 8

16.Information about further partitioned drives is available at _____ physical block of extended partition.

- **First** **PG #222**
- Second
- Third
- Fourth

17._____ will be used to get drive parameters.

- Int 13H/42H
- **Int 13H/48H** **PG #238**
- Int 13H/66H
- Int 13H/66H

18.Inside a boot block operating system name occupies _____ bytes.

- 3
- 5
- 8 **Not sure**
- 11

دنیا کی سب سے بڑی فتح نفس پر قابو رکھنا ہے

19. A programmer wants to access the DPB(Drive Parameter Block) for C drive using undocumented service 21H/32H, what he should place in DL register _____ ?

- **0** **PG #249**
- 1
- 2
- 3

DL – 0 for current Drive

20. Blocks for FAT can be accessed using _____.

- BPB
- DPB
- **FCB**
- Both BPB and DPB.

21. In main memory smallest addressable unit is ____.

- **Byte**
- Nibble
- Word
- None of the given

22. Using the root directory entry and the _____, we can access the contents of File.

- Reserved Blocks
- Number of FAT copies
- **File Allocation Table(FAT)** **PG # 241**
- None of the given

23.If a file is having more than one cluster then it will be managed by _____.

- FAT
- BPB
- DPB
- **None of the given**

24.Practically in FAT 32, total number of entries are _____.

- 2^{26}
- 2^{28}
- 2^{30}
- **2^{32}**

25._____ keeps the back up of its boot block.

- FAT 12
- FAT 16
- **FAT 32**
- None of the given

26.NTFS can store file name in _____ form.

- **UNICODE** **Click Here For More Detail**
- Both ASCII and UNICODE
- None of the given
- ASCII

27.FAT based file system can store file name in _____ form.

- ASCII
- UNICODE
- **Both ASCII and UNICODE**
- None of the given

28.The LSN of first logical sector of NTFS partition is _____.

- **0** **PG # 240**
- 1
- 2
- 3

29.To minimize _____ defragmenter is used.

- **Seek time** **PG # 316**
- Access time
- Rotational delay
- Both seek and access time

30.In memory map of first 1 MB of RAM, the higher _____ is called system memory.

- 64 KB
- **384 KB** **PG # 317**
- 640 KB
- None of the given

Memory map of the first 1MB of RAM. The first 640KB is called conventional RAM and the higher 384KB is called system memory.

جھوٹ انسان اور ایمان دونوں کا دشمن ہے

31.Extended memory can be accessed in _____ mode.

- Real
- **Protected**
- Both real and protected
- None of the given

32.Extended memory can be accessed only after installing the driver _____ .

- EMM386.EXE
- **HIMEM.SYS** **PG # 318**
- IO.SYS
- None of the given

33.DOS uses _____ memory allocation system.

- **Contiguous** **PG # 325**
- Non Contiguous
- Both Contiguous and Non Contiguous
- None of the given

34.In protected mode segment register will be called _____.

- **Selector** **PG # 326**
- Descriptor
- IDT
- Both Selector and Descriptor

35.There are _____ Control registers in 80386 and above systems.

- 2
- **4** **PG # 331**
- 8
- 16

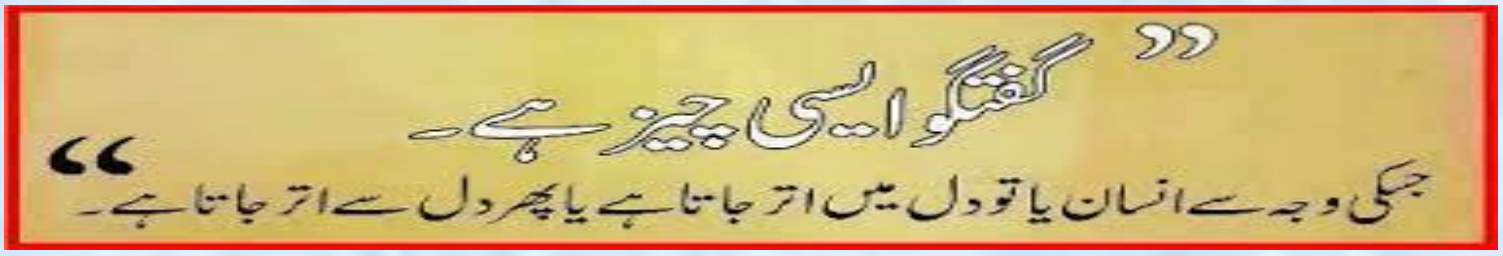
36.If the CHS (cylinder, head, and sector) address of a disk is known it can be translated in to the _____ address and vice versa.

- Logical
- **LBA** **PG # 213**
- Physical
- None of the Given

37._____ is a data structure maintained by DOS in the boot block for each drive.

- DPB
- **BPB** **PG # 242**
- FCB
- None of the Given

BIOS parameter block



38.System Stores the DPBs (Drive Parameter Block) for different drives in the form of _____.

- Chain / link list
- Array
- **Table** **Not sure**
- None of Given

39.The extension of a compiled TSR (Terminate and Study Resident) program should be _____

- .EXE
- **.COM**
- .CPP
- .C

40. A single interrupt controller can arbitrate among ____ different devices.

- 4
- 6
- **8** **PG # 47**
- 10

41.The microprocessor package has many signals for data. Below are some in Correct priority order (Higher to Lower).

- Reset,Hold,NMI,INTR
- **NMI, INTR,Hold,Reset** **PG # 46**
- INTR,NMI,Reset,Hold
- None of the Given

42.A standard PC can have _____PPI.

- 1
- **4** **PG # 84**
- 8
- 16

A standard PC can have 4 PPI named LPT1, LPT2, LPT3 and LPT4

43.BIOS DO NOT support _____.

- LPT1
- LPT2
- LPT3
- **LPT4**

44._____ is used to set date.

- 1A/02H
- 1A/03H
- 1A/04H
- **1A/05H** **PG # 139**

45.If real time clock interrupts microprocessor then interrupt ____ will occur.

- 13h
- 17h
- **70h**
- 71h

46._____ is LED control byte.

- 0xF3
- **0xED** **PG # 181**
- 0xE5
- 0xFF

47.The DMA requests to acquire buses through the _____ signal.

- **HOLD** **PG # 186**
- ACR
- ACK
- All of the given

48._____ will specify if the next DMA transfer will happen as a single transfer, block transfer or demand transfer

- DMA Request register
- DMA Mask register
- **DMA Mode register**
- DMA Command register

49._____ will specify if the channel is to cascade two DMA controllers.

- DMA Request register
- DMA Mask register
- **DMA Mode register**
- DMA Command register

50._____ are the longitudinal division of the disk.

- Tracks
- **Sectors** **PG # 201**
- Blocks
- None of given

Tracks are the circular division of the disk and the sectors are the longitudinal division of the disk

51.Highest physical capacity of the disk according to the IDE interface is_____.

- 504 MB
- 80 GB
- **127 GB** **PG # 212**
- 300 GB

52.In partition table, first _____ bytes contain code which loads the boot block of active partition.

- 128
- 286
- **446** **PG # 218**
- 500

53.On a single disk there can be _____ different Operating Systems.

- 2
- **4** **PG # 219**
- 6
- 8

عقل مند کہتا ہے میں کچھ نہیں جانتا جبکہ بے وقوف کہتا ہے کہ میں سب کچھ جانتا ہوں

54._____ is physical or absolute address.

- **LBA** **PG # 240**
- LSN
- CHS
- None of the given

55.The _____ block is the first block on disk.

- LSN =0
- **LBA=0** **PG # 240**
- Both LBA=0 and LSN=0
- None of the given

56.A directory is collection of _____.

- Bios Parameter Blocks
- Drive Parameter Blocks
- **File Control Blocks**
- None of the given

57.First cluster in user data area is numbered _____ in a FAT based systems.

- 0
- 1
- **2** **PG # 258**
- 3

58.Blocks for FAT can be accessed using _____.

- BPB
- DPB
- **FCB**
- Both BPB and DPB.

59.In FAT12 to calculate the address or offset from index, we need to multiply it with ___.

- 1/2
- **3/2** **PG 267**
- 5/2
- 7/2

60.Using the root directory entry and the _____, we can access the contents of File.

- Reserved Blocks
- Number of FAT copies
- **File Allocation Table(FAT)** **PG #269**
- None of the given

61.In FAT16 if FAT entry is between FFF0H to FFF6H then _____.

- Cluster is available
- **It is a Reserved cluster**
- It is next file cluster
- It is a last file cluster

62._____ needed to store a UNICODE character.

- Nibble
- **2 Bytes**
- 4 Bytes
- Byte

63._____ used for FCB in FAT 12 and FAT 16.

- Nibble
- Byte
- **2 Bytes**
- 4 Bytes

64._____ file system is used in CD's.

- None of the given
- **Contiguous**
- Chained
- Indexed

65.FAT based file system can store file name in _____ form.

- ASCII
- UNICODE
- **Both ASCII and UNICODE**
- None of the given

66.In case of _____ contents, the indexes of clusters of file are stored in MFT.

- Small files
- Large files
- **Both small and large files**
- None of the given

67.In memory map of first 1 MB of RAM, the higher _____ is called system memory.

- 64 KB
- **384 KB**
- 640 KB
- None of the given

Memory map of the first 1MB of RAM. The first 640KB is called conventional RAM and the higher 384KB is called system memory

68.PC's operated on 20 bit address bus can operate in _____ mode.

- **Real** **PG # 319**
- Protected
- Both real and protected
- None of the given

خود کو تمھیں سے بڑھ کر کوئی اچھا مشورہ نہیں دے سکتا

69. Int 21H/52H returns the address of DOS internal data structures in ES: BX, the far address of the first MCB in memory lies _____ behind the address returned.

- 2-bytes
- **4-bytes**
- 6-bytes
- 8-bytes

70. In memory management, descriptor entry is of _____ bytes.

- **64**
- 32
- 8
- 16

71. There are _____ Control registers in 80386 and above systems.

- 2
- **4** **PG # 331**
- 8
- 16

72. Control Registers are used for conveying certain control information for _____ Addressing and Co-Processors.

- Real Mode
- **Protected Mode** **PG # 331**
- Virtual Mode
- None of the given

73.The least significant bit of Control register _____ is PE-bit which can be set to enable Protected Mode Addressing.

- **CR0**
- CR1
- CR2
- CR3

74.CD.. command gives the _____ of directory.

- None of the given
- One previous level
- One next level
- **Current path** **PG #282**

75.Cluster size is variable and can be measured in power of _____.

- **2**
- 16
- 4
- 8

کسی انسان کی خوبی کو پہچانو اور اِسے بیان کرو، لیکن اگر کسی کی
خامی مل جائے تو یہاں تمہاری خوبی کا امتحان ہے۔
فرمان حضرت علیؓ

76.Who notifies the end of interrupt (EOI) to the programmable interrupt controller?

- Programmer
- **Microprocessor**
- Operating system
- None of the given

77.Choose the correct statement to notify the interrupt controller about end of interrupt (EOI).

- Inportb(0x20,0x20)
- **Outportb(0x20,0x20)**
- Outportb(0x30,0x22)
- None of the given

78.Information about Pending interrupts are held by _____.

- Interrupt Service Register
- Interrupt Mask Register
- None of the Given
- **Interrupt Request Register**

79. BPB stands for _____.

- **BIOS parameter block**
- BIOS processing block
- Base processing block
- BIOS partition block

80.The input frequency used by the interval timer is the _____signal generated by the clock generator.

- None of Given
- **PCLK**
- ACL
- ICP

81.BIOS DO NOT support _____.

- LPT1
- LPT2
- LPT3
- **LPT4**

82.DTE is _____.

- **Data terminal equipment** **PG #109**
- Data transmitting equipment
- Dual terminal equipment
- None of the given.

83.Int 14H _____ can be used to send a byte.

- Service # 0
- **Service # 1**
- Service # 2
- Service # 3

جو شخص ناکامیوں سے ڈر کر بھاگتا ہے کامیابی اُس سے ڈر کر بھاگتی ہے

84.A single DMA can transfer _____ operands to and from memory in a single a bus cycle.

- **8 bit**
- 16 bit
- 32 bit
- 64 bbit

85.DMA does not work in _____ mode.

- **Double Mode**
- Block Transfer
- Demand Transfer
- None of the given

86.An addressable unit on disk can be addressed by three parameters _____.

- **head #, sector # and track #**
- HD #, AH# and track #
- Hd0 #, sector # and track #
- None of the given

87.Each addressable unit has a unique combination of sec#, head# and track# as its _____ address.

- Both Physical and Logical
- None of the given
- **Physical**
- Logical

جو لوگوں کے سامنے فخر کرتا ہے وہ لوگوں کی نظروں سے گر جاتا ہے

88.In partition table, first _____ bytes contain code which loads the boot block of active partition.

- 128
- 286
- **446**
- 500

89.On a single disk there can be _____ different Operating Systems.

- 2
- **4**
- 6
- 8

90.The last partition table within the chain contains _____ entry/entries about the logical drive.

- **Single**
- Double
- Three
- Four

91.Partition Table can be read using the extended _____ Services.

- 13H
- **14H**
- 15H
- 17H

92.File control block (FCB) is _____ byte long.

- 8
- 16
- **32**
- 64

93._____ is collection of contagious blocks.

- **Cluster**
- Sector
- File
- Directory

94.A FAT 12 table can have maximum _____ values.

- 1024
- 2048
- **4096**
- None of given

2^12 entries maximum = 4096

95.The first cluster number of file can be found in _____.

- BPB
- DPB
- **FCB** **PG # 265**
- None of the given

96.In FAT12 if FAT entry is 000H then _____.

- It is a last file cluster
- Cluster is available
- **It is a Reserved cluster**
- It is next file cluster

97.Total number of clusters of FAT12 are _____.

- FF0 H
- FFF H
- FEF H
- **FEE H** **PG # 266**

98.When we mark a file as deleted by placing 0xE5 then the chain of clusters in FAT is also replaced by _____.

- E5
- **0**
- 1
- N

99.To store a cluster in FAT 32 _____ is/are needed.

- Byte
- 2 Bytes
- **4 Bytes**
- Nibble

100._____ keeps the back up of its boot block.

- FAT 12
- FAT 16
- **FAT 32**
- None of the given

101.Up to _____ bytes can be used to store a file name in NTFS.

- 30
- 126
- **254**
- 510

102.First logical sector of NTFS partition is _____.

- DPB
- MFT
- **Boot sector**
- None of the given

103.The LSN of first logical sector of NTFS partition is _____.

- **0**
- 1
- 2
- 3

104.In case of _____ the contents of file are stored in MFT.

- Small files
- Large files
- **Both small and large files**
- None of the given

105.Interrupt Descriptor Table can have up to _____ entries.

- 64
- 128
- **256**
- 512

106.There can be _____ different descriptors privilege levels.

- 2
- **4**
- 8
- 10

107.The least significant bit of Control register _____ is PE-bit which can be set to enable Protected Mode Addressing.

- **CR0** **PG # 331**
- CR1
- CR2
- CR3

108.Entry point of execution in EXE File can be _____.

- From first instruction
- From last instruction
- **Anywhere in the program** **PG # 335**
- From anywhere in the middle

109.In NTFS, Block # ____ is the safest block to store the back up of boot block.

- 8
- **10** **Not sure**
- 20
- 6

110.Size of single entry in Partition Table is _____.

- **512 bytes**
- 128 bytes
- 64 bytes
- 16 bytes

111.MBR partition table can have _____ entries at maximum.

- 1
- 2
- 3
- **4** **Not sure**

112._____ is a data structure maintained by DOS in the boot block for each drive.

- DPB
- **BPB** **PG # 242**
- FCB
- None of the Given

113.Hidden blocks between the first physical block on each partition are used by _____ for storing of data

- User
- Operating System
- BIOS
- **None of the Given**

114.Which of the following data structure is NOT used in Operating System's File management?

- **Memory Control Block** **PG # 06**
- File Allocation Table
- Drive Parameter Block
- File Control Block

115.Interrupt procedures are _____ procedures.

- **Reentrant** **PG # 38**
- Non-Reentrant
- Recursive
- Virtual

116.The extension of a compiled TSR (Terminate and Study Resident) program should be _____

- .EXE
- **COM**
- .CPP
- .C

to make a .C program a TSR program, you are required to generate a .COM file by using the following command in DOS prompt.

bcc –EEXAMPLE.COM –IC:\BORLANDC\INCLUDE –LC:\BORLANDC\LIB EXAMPLE.C

Here, we are generating EXAMPLE.COM for of EXAMPLE.C program.

117.Choose the correct statement to notify the interrupt controller about end of interrupt (EOI).

- Inportb(0x20,0x20)
- **Outportb(0x20,0x20)**
- Outportb(0x30,0x22)
- None of the given

118.Information about Pending interrupts are held by _____.

- Interrupt Service Register
- Interrupt Mask Register
- **Interrupt Request Register**
- None of the Given

عقل مند اپنے عیب خود دیکھتا ہے اور بیوقوفوں کے عیب دنیا دیکھتی ہے

119. _____ Procedure can have parameters.

- Hardware Interrupt
- Software Interrupt
- **Both Hardware and software Interrupt** **PG # 10**
- None of the Given

120.The Address of partition block on hard disk is _____.

- head # =0, track # = 0 and sector # = 0
- head # =0, track # = 0 and sector # = 1
- head # =0, track # = 1 and sector # = 1
- **head # =1, track # = 0 and sector # = 1** **PG # 42**

121.NMI Stand for _____

- **Non Maskable Interrupt** **PG #46**
- Non Multitude Interrupt
- Non Maskable Instruction
- None of Given

122.The interval timer is used to divide an input frequency.

- **True** **PG # 68**
- False

بدصورت چہرہ بدصورت دماغ سے بہتر ہے

123.The PPI acts as an interface between the CPU and a parallel _____ .

- **I/O device** **PG # 86**
- CPU
- BUS
- None of Given

124.BIOS DO NOT support _____.

- LPT1
- LPT2
- LPT3
- **LPT4**

125._____ is used to identify the cause of interrupt.

- Interrupt Enable register
- **Interrupt ID register**
- Interrupt Status register
- None of the given

126.The bit _____ of Line control register in UART, if cleared will indicate that DLL is the data register.

- 1
- 3
- 5
- **7**

127.Int 14H _____ can be used to send a byte.

- Service # 0
- **Service # 1** **PG # 121**
- Service # 2
- Service # 3

Service of 14h include service #1 which is used to send a byte and service #2 which is used to receive a byte

128.To access bettery powered RAM, only _____ ports are important from programming point of view.

- **70 and 71H** **PG # 141**
- 71 and 72H
- 70 and 72H
- 72 and 73H

129.The DMA requests to acquire buses through the _____ signal.

- **HOLD** **PG # 186**
- ACR
- ACK
- All of the given

130._____ used to program various common parameters of transfer for all the channels.

- None of the above
- DMA Status Register
- **DMA Command Register** **PG # 191**
- DMA Request Register

131.Bit # _____ of mode register in DMA determine the direction of a transfer.

- **2**
- 3
- 4
- 5

132._____ will specify if the next DMA transfer will happen as a single transfer, block transfer or demand transfer.

- DMA Request register
- DMA Mask register
- **DMA Mode register**
- DMA Command register

133.Each addressable unit has a unique combination of sec#, head# and track# as its _____ address.

- Both Physical and Logical
- None of the given
- **Physical** **PG # 202**
- Logical

134.Highest capacity of disk can be accessed using BIOS functions is _____.

- 128 MB
- 256 MB
- **504 MB** **PG # 211**
- 127 GB

عقل مند آدمی اس وقت تک نہیں بولتا جب تک خاموشی نہیں ہو جاتی

135.DOS has built in limit of _____ blocks per cluster.

- 32
- 64
- **128** **PG # 242**
- 256

136.BIOS Parameter block is situated in _____ Block.

- **Boot** **PG # 242**
- Data
- Extended Data
- None of Given

137.The _____ block is the first block on disk.

- LSN =0
- **LBA=0** **PG # 240**
- Both LBA=0 and LSN=0
- None of the given

138.Inside a boot block jump code part occupies _____ byte.

- 5
- 8
- 11
- **3** **PG # 244**

بہترین تجربہ وہ ہے جس سے نصیحت حاصل ہو

139.In boot block BIOS parameter block starts from _____.

- 03H
- 05H
- 08H
- **0BH**

140.In main memory smallest addressable unit is ____.

- **Byte**
- Nibble
- Word
- None of the given

141.Total number of clusters of FAT12 are _____.

- FF0 H
- FFF H
- FEF H
- **FEE H** **PG # 266**

142.In FAT12 to calculate the address or offset from index, we need to multiply it with ___.

- 1/2
- **3/2** **PG# 267**
- 7/2
- 5/2

143.Total number of cluster of FAT16 is _____.

- FFF0 H
- FFFF H
- FFEF H
- **FFEE H** **Page 272**

144.Total ____ fragments can be supported for storing long file names.

- 26
- 28
- **32**
- 48

145._____ used for FCB in FAT 12 and FAT 16.

- Nibble
- Byte
- **2 Bytes**
- 4 Bytes

146.To store a cluster in FAT 32 _____ is/are needed.

- Byte
- 2 Bytes
- **4 Bytes**
- Nibble

147.Practically in FAT 32, total number of entries are _____.

- 2^{26}
- 2^{28}
- 2^{30}
- **2^{32}**

148.First logical sector of NTFS partition is _____.

- DPB
- MFT
- **Boot sector**
- None of the given

149.Extended memory can be accessed in _____ mode.

- Real
- **Protected**
- Both real and protected
- None of the given

150.MCB is a _____ bytes data structure.

- 8
- **16** **PG # 321**
- 32
- 64

151.Real mode does not support _____ memory allocation system.

- **Contiguous**
- Non Contiguous
- Both Contiguous and Non Contiguous
- None of the given

152.There can be _____ different descriptors privilege levels.

- 2
- **4**
- 8
- 10

153.In memory management, descriptor entry is of _____ bytes.

- 8
- 16
- 32
- **64**

154.For virus to propagate itself, it has to intercept interrupt _____.

- 9H
- 11H
- **13H**
- 17H

خوبصورتی علم و ادب سے ہوتی ہے لباس و حسن سے نہیں

155.If virus wants to be in memory independently, it should have its own _____.

- MCB
- PSP
- EB
- **Both MCB and PSP** **PG #334**

156.Size of single entry in Partition Table is _____.

- **512 bytes** **PG#218**
- 128 bytes
- 64 bytes
- 16 bytes

157._____ is a data structure maintained by DOS in the boot block for each drive.

- DPB
- **BPB** **PG#242**
- FCB
- None of the Given

158.Cluster size is variable and can be measured in power of _____.

- **2** **PG#242**
- 16
- 4
- 8

159.Bit # _____ of Eflag is used for alignment check

- 12
- 14
- 15
- **18** **PG # 164**

160.Each addressable unit has a unique combination of sec#, head #, track # as its _____ address.

- **Physical** **PG #202**
- Logical
- Both
- None

161.First cluster in user data is numbered in a FAT based system.

- 0
- 1
- **2** **PG # 258**
- 3

162.BIOS services understand --------.

- **LBA** **PG # 212**
- LSN
- Cluster #
- None

163.The first cluster number of a file can be found in---------

- BPB
- DPB
- **FCB** **PG # 265**
- None

164.The size of FS Info block is

- 64byte
- 128 byte
- 256 byte
- **512 byte** **PG # 300**

165.In NTFS first -------- entries are reserved.

- 4
- 6
- **16** **PG # 303**
- 32

166.In memory map of first 1 MB of ram ,the first -------- is called conventional RAM.

- 64kb
- 384kb
- **640kb** **PG # 317**
- None

167.In memory map of first 1 MB of ram ,the higher -------- is called system memory.

- 64kb
- **384kb** **PG # 317**
- 640kb
- None

168.The -------- of boot block constitutes of BPB.

- Code part
- **Data part** **PG # 242**
- Both
- None

169.Extended BIOS function make use of --------- address

- **LBA** **PG # 212**
- CHS
- LSN
- None

170.LBA address can be used in place of the CHS address.

- **True** **PG # 235**
- False

171.In FAT12, the maximum range of clusters is ……...

- 0 ~ FEFH
- 1~ FEFH
- **2 ~ FEFH** **PG # 266**
- 3 ~ FEFH

172.NTFS volume can be accessed directly in DOS.

- True
- **False** **PG # 310**

173.Each partition information chunk is 16 bytes long and the last two bytes at the end of the partition table data part is the partition table signature whose value should be _____ indicating that the code part contains valid executable code.

- 00AA
- 0055
- 050A
- **AA55** **PG # 219**

174.Service 21H/52H service returns the address of DOS internal data structures in ES: BX _____behind the address returned lies the far address of the first MCB in memory.

- 2-bytes
- **4-bytes** **PG # 322**
- 6-bytes
- 8-bytes

175.80386 can have _____ control registers.

- 2
- 5
- 3
- **4** **PG # 331**

176.The partition table uses the extended _____ service.

- **13H** **PG # 234**
- 14H
- 15H
- 16H

177.The entry point of execution in EXE File can be

- Start of the first instruction
- Start of the last instruction
- **Anywhere in the Program** **PG # 335**
- Can be in the middle of the program

انسان کے لئے بری صحبت سے بڑھ کر بری کوئی چیز نہیں

178.Using the____ entry and the FAT we can access the contents of file.

- Reserved blocks
- **Root Directory** **PG # 269**
- Number of FAT copies
- None of the given

179.Control information in files is maintained using

- BPB
- DPB
- **FCB** **PG # 256**
- FPB

180.What will happen if NTFS volume is accessed in DOS?

- Convert it to FAT volume
- Nothing will happen
- **Error of invalid media** **PG # 310**
- None of the given

181.LSN of FS Info block is available at

- BPB
- **FAT**
- Root Directory
- None of the given

خاموشی غصے کا بہترین علاج ہے

182.DOS device drivers do not understand the_____ data structures.

- FAT12
- FAT16
- FAT32
- **NTFS** **PG # 310**

183._____ is a collection of contagious blocks.

- **Cluster** **PG # 242**
- Sector
- Byte
- None of Given

184._____ used to determine the amount of conventional memory interfaced with the processor in kilobytes.

- INT 10 H
- INT 11 H
- **INT 12 H** **PG # 162**
- INT 13 H

185.Bit number _____ of coprocessor control word is the Interrupt Enable Flag.

- **7** **PG #168**
- 8
- 9
- 10

186.To distinguish 486 with Pentium CPUID Test is used.

- **True** **PG # 166**
- False

187.Practically _____ entries are there in FAT 32.

- 2^26
- 2^28
- 2^30
- **2^32** **PG # 265**

188.BPB stands for _____.

- **BIOS parameter block** **PG # 243**
- BIOS processing block
- Base processing block
- BIOS partition block

189.The keyboard input character scan code is received at ___ port.

- **60H** **PG # 179**
- 61H
- 62H
- 63H

190._____ is LED control byte.

- 0xFD
- **0xED** **PG #181**
- 0xFF
- 0xEE

جھوٹ رزق کو کھا جاتا ہے

191._____means typematic rate will be sent in next byte.

- **0xF3** **PG # 180**
- 0xF4
- 0xF5
- 0xF6

192.Keyboard uses port ____ as status port.

- **64H** **PG # 177**
- 66H
- 67H
- 69H

193.The keyboard can perform _____ serial I/O.

- asynchronous
- **synchronous**
- Multiple
- Single

194.Bit number 2 of port 64H Status register used for output buffer full.

- True
- **False**

195.Bit number _____ can declares the parity error of port 64H Status register.

- 4
- 5
- 6
- **7**

196.Bit number _____ of port 64H Status register used for input buffer full.

- **0**
- 1
- 2
- 3

197.Disadvantage of FAT32 is _____.

- Large disk size can be managed in FAT32
- Cluster size is reduced
- Internal fragmentation is reduced
- **Very large table** **PG # 299**

198. Maximum possible entries in FAT12 are _____.

- 1024
- 2048
- **4096** **PG # 264**
- 65536

199. What will be the value of the word located at 1Fh in DPB when number of free clusters on drive is not known?

- 0000H
- 1111H
- **FFFFH** **PG # 250**
- None of the given.

افضل انسان وہ ہے جو اپنی اصلاح کی کوشش کرتا ہے

200.Jump code part contains ____ bytes in boot block.

- **3** **PG # 302**
- 5
- 8
- 11

201. Operating system name contains ____ bytes in boot block.

- 3
- 5
- **8** **PG # 257**
- 11

202.File can be _____ viewed as organization of data.

- Physically
- **Logically** **PG # 256**
- Both logically and physically
- None of the give

203. _____ is used to read a block against its LSN.

- **absread()** **PG # 247**
- abswrite()
- lsnread()
- None of the given

204. File can be_____ viewed as collection of clusters or blocks.

- **Physically** **PG # 256**
- Logically
- Both physically and logically
- None

205. When we talk about FAT based file system, in user data area first cluster number is _____.

- 0
- 1
- **2** **PG # 258**
- None of the given

206. Cluster number can also be referred as block number.

- True
- **False** **PG # 258**

207.To access the block within cluster using BIOS services the cluster number should be converted into _____.

- CHS
- LBA
- **LSN** **PG # 258**
- None of the given

208.What will be the value of DL register when we are accessing C drive using undocumented service 21H/32H?

- 0
- 1
- 2
- **3** **PG # 249**

209.The directory structure of DOS is like _____.

- Array
- **Tree** **PG # 256**
- Linked list
- None of the given

210.Control information about files is maintained using _____.

- BPB
- DPB
- **FCB** **PG # 256**
- FPB

211.When LSN is equal to zero (0), it means _____.

- First block of the disk
- **First block of the logical drive** **PG # 240**
- First block of hidden blocks
- None of the given

212.In FAT32, lower _____ bits are used.

- 26
- **28** **PG # 292**
- 30
- 32

213._____ is relative address with respect to the start of Logical Drive.

- LBA
- **LSN** **PG #240**
- CHS
- None of the given

214.The practical limit of blocks per cluster is _____.

- 32 blocks per cluster
- **64 blocks per cluster** **PG #242**
- 128 blocks per cluster
- 256 blocks per cluster

215.In dos we have limit of _____.

- **128 blocks per cluster** **PG #242**
- 256 blocks per cluster
- 32 blocks per cluster
- 64 blocks per cluster

216.Highest capacity physical capacity of the disk according to the IDE interface is _____.

- **127 GB** **PG # 212**
- 100 GB
- 80 GB
- 300 GB

217.Partition Table can be read using the extended _____ Services.

- **13 H** **PG # 234**
- 14 H
- 15 H
- None of given

218.In Protected Mode, the segment registers are used as _____

- Descriptor
- **Selector** **PG # 326**
- All of the given choices
- None of the given choices

219.To access drive parameter block we use undocumented service _____

- 09H/32H
- 11H/32H
- 17H/32H
- **21H/32H** **PG # 249**

220. _____ is an absolute address relative to the start of physical drive.

- **LBA** **PG # 240**
- LSN
- CHS
- None of the above

221.Boot block consists of _____ bytes.

- 64
- 128
- 256
- **512** **PG # 242**

222.The DMA requests to acquire buses through the _____ signal.

- **HOLD** **PG #186**
- ACR
- ACK
- None of Given

اس سے پہلے کہ تمہیں شہوت فتنے میں ڈالے نکاح کرلو

223.The keyboard device writes a code 0xFA on the port 60H to indicate that the _____.

- Input buffer is full
- **Byte has been received properly** **PG # 179**
- Output buffer is full
- None of the given

224.A single DMA can transfer _____ operands to and from memory in a single a bus cycle.

- **8-bits** **PG # 186**
- 16-bits
- 32-bits
- 12-bits

225.In FAT12, to calculate the address or offset from index, we need to multiply it with ___.

- 1/2
- **3/2** **PG #267**
- 5/7
- 7/2

226._____ Register can be used to show that the channel is single transfer, block transfer or demand transfer mode.

- DMA Command register
- DMA Request Register
- DMA Mode Register
- **DMA controller Register** **PG #187-188**

227.When we mark a file as deleted by placing 0xE5 then the chain of clusters in FAT is also replaced by _____.

- E5
- 1
- **0** **PG # 79**
- N

228.Cluster size is reduced in _____.

- FAT12
- FAT16
- **FAT32** Click here for detail
- None of the given

229.In FAT32 _____ root directory entries are there.

- 128
- 256
- 512
- **None of the given** Click here for detail

230.If a file is having more than one cluster then it will be managed by _____.

- FAT
- BPB
- DPB
- **None of the above**

بری صحبت سے تنہائی بہتر ہے اور تنہائی سے نیک صحبت بہتر ہے

231.Internal fragmentation is reduced in _____.

- FAT12
- **FAT16** **Click here for detail**
- FAT32
- None of the given

232.For supporting long file names, _____ fragments can be supported.

- 12
- 20
- 26
- **32**

233.To store a cluster in FAT 32 _____ is/are needed.

- Nibble
- Byte
- 2 Bytes
- **4 Bytes** **Click here for detail**

234.If a file size is 12K and the size of the cluster is 4K then _____ clusters are used for the file.

- 2
- **3**
- 4
- 5

ایماندار کو غصہ دیر سے آتا ہے اور جلدی دور ہو جاتا ہے

235.We can access the contents of File by using the root directory entry and _____.

- Reserved Blocks
- Number of FAT copies
- **File Allocation Table (FAT)** **PG # 269**
- None of the given

236.FAT based file system can store file name in _____ form.

- ASCII
- UNICODE
- **Both ASCII and UNICODE**
- None of the given

237.Drive parameter block is derived from _____.

- FCB
- FAT
- **BPB** **PG # 249**
- CPB

238.We can access Blocks for FAT using _____.

- BPB
- DPB
- **FCB**
- Both BPB and DPB

جھوٹ انسان اور ایمان دونوں کا دشمن ہے

239.If we know the cluster number, we can access the blocks within the cluster using BIOS services directly.

- **True** **PG # 258**
- False

240._____ is an internal data structure of DOS and resides in main memory.

- BPB
- DPB
- CPB
- **None of the given**. **Click here for detail**

241.The size of DPB data structure is _____ bytes.

- 16
- 32
- 64
- **128** **click here for detail**

242.The size of FCB data structure is _____ bytes.

- **16** **Click here for detail**
- 32
- 64
- 128

243.Advantages of FAT32 is/are _____.

- Large disk size can be managed in FAT32
- **Cluster size is reduced** Click here for detail
- Internal fragmentation is reduced
- All of the given

ایماندار کو غصہ دیر سے آتا ہے اور جلدی دور ہو جاتا ہے

244._____ file system keeps the backup of its boot block.

- FAT12
- FAT16
- **FAT32** **Click here for detail**
- None of the given

245.To store a UNICODE character ____ is/are needed.

- Nibble
- Byte
- **2 Bytes** **Click here for detail**
- 4 Bytes

246._____ is the first block on disk.

- LSN =0
- **LBA=0** **PG # 240**
- LBA=1
- Both LBA=0 and LSN=0

247.If FAT entry is between FFF0H to FFF6H in FAT16 then _____.

- Cluster is available
- **It is a Reserved cluster** **PG # 272**
- It is next file cluster
- It is a last file cluster

خود کو تمھیں سے بڑھ کر کوئی اچھا مشورہ نہیں دے سکتا

248.File system used in CD's is_____ file system

- **Contiguous** **Click here for detail**
- Chained
- Indexed
- None

249.A file has 2 clusters and the size of cluster is 4K. What will be the size of file?

- 2K
- **8K**
- 16K
- 32K

250.In NTFS, Backup of boot block is stored at block # _____.

- 2
- 6
- **8**
- 10

251.The interval timer can operate in _____ modes.

- Five
- Seven
- Four
- **Six** **PG # 72**

252.File control block (FCB) is _____ byte long.

- **32** **Click here for detail**
- 64
- 16
- 128

253.DOS command _____ which gives the status of the memory and also points out which memory area occupied by which process.

- **mem/d** **PG # 13**
- mem/e
- mem/m
- None of the given

254.Each entry in the IVT is _____ in size.

- **4-bytes** **PG # 12**
- 6-bytes
- 8-bytes
- 2-bytes

255.The tail value of the keyboard buffer should be examined to get to the _____ of the buffer.

- **Start** **PG # 56**
- End
- Middle
- None of given

256.Usually interrupt procedures are reentrant procedures especially those interrupt procedure compiled using C language compiler are reentrant.

- **True** **PG # 38**
- False

جو شخص ناکامیوں سے ڈر کر بھاگتا ہے کامیابی اُس سے ڈر کر بھاگتی ہے

257._____ is Disk interrupt.

- 10H
- 11H
- **13H PG # 42**
- 14H

258.In parallel communication, the maximum numbers of bits we can send between two computers are _____.

- 2-bits
- 4-bits
- 6-bits
- **8-bits**

259.14h include _____ which is used to send a byte.

- Service #0
- **Service #1 PG # 121**
- Service #2
- Service #3

260.The status register _____ is the main control register.

- **B PG # 146**
- A
- C
- D

261._____ is used to identify the cause of interrupt.

- **Interrupt ID Register PG # 116**
- PC Register
- AC Register
- None of All These

262.In NTFS, up to _____ characters are used to store files names,

- 30
- 48
- **255** **PG # 283**
- 510

263.A cluster is a collection of contiguous _____.

- **Blocks** **PG # 242**
- Sectors
- Bytes
- None of Given

264.In BPB, root directory is saved in _____.

- **Cluster#0**
- Cluster#1
- Cluster#2
- Cluster#3

265.In NTFS, total sizes of MFT entries are _____.

- **16-bytes** **PG # 303**
- 20-bytes
- 26-bytes
- 32-bytes

266.In NTFS, _____ store the contents of file.

- **Both small & large file Record**
- Small record
- Large Record
- None of given

267.In NTFS, contents and indexed of file is stored in _____.

- Small record
- Large Record
- Both small & large file Record
- **None of given**

268.Total No. of bytes that can be stored in Keyboard Buffer is____.

- 16
- **32** **PG #54**
- 64
- 128

269.BIOS support _____UARTS as COM ports.

- 6
- **4** **PG #113**
- 3
- 2

270.DCE stands for _____.

- **Data communication equipment** **PG # 109**
- Distributed Computing Environment
- Data Communications Equipment
- Data Carrier Equipment

271.In counter register bit no. 3 changes its value between 0 and 1 with in ____clock cycles

- 1
- 2
- 4
- **16** **PG #69**

272.In _____each byte is needed to be encapsulated in start and end.

- Synchronous communication
- **Asynchronous communication** **PG # 106**
- Both
- None of given

273.The _____ service # is not used in any interrupt.

- 01
- 02
- 03
- **FF**

274.If we want to send printing on the printer then we have to perform following steps.

- Initialize printer
- Read Status
- Check Error
- **All of the given**

275.If printer is _____ then printer sends back the ACK signal to the printer interface

- **idle** **PG # 97**
- busy
- Out of paper
- None of the given

276.DSR stands for _____.

- **Data set ready** **PG # 111**
- Data service ready
- Data stock ready
- None of the given

277.At IRQ 7 Interrupt # ___ is used.

- 0x0A
- 0x0B
- 0x0C
- **0x0F PG # 95**

278.The memory addresses of COM ports remain same for all computers

- True
- **False**

279.In keyboard status byte bit no. 2 and 3 are used for ctrl and alt keys respectively. which of the following condition is used to check that Ctrl + Alt keys are pressed. Where: unsigned char far * scr = (unsigned char far *)(0x00400017);

- **if (((*scr)&12)==12)**
- if (((*scr)&8)==8)
- if (((*scr)&4)==4)
- if (((*scr)&2)==2)

280.In case of synchronous communication a timing signal is required to identify the start and end of a bit.

- **True PG # 105**
- False

281.The baud rate is set in accordance with the divisor value loaded within the UART internal registers base +0 and base +1.

- **True PG # 114**
- False

عقل مند آدمی اس وقت تک نہیں بولتا جب تک خاموشی نہیں ہو جاتی

282.Software based flow control make use of -------- control characters

- Xon
- XOFF
- **Both** **PG # 135**
- None

283._____ is used to read time from RTC

- **1A\02H** **PG # 137**
- 1A\03H
- 1A\04H
- 1A\05H

284.Int_____ service 0 can be used to set the line parameter of the UART or COM port.

- **14H** **PG # 119**
- 15H
- 13H
- None of the given option

285.When LBA is equal to zero (0), it means _____.

- **First block of the disk** **PG # 240**
- First block of the logical drive
- First block of the hidden block
- None of the given

286.In IRQ2 and IRQ3 which one has the highest priority?

- Can't be determined
- Both have same priority
- IRQ3
- **IRQ2** **PG # 47**

287.Following is not a method of I/O

- Programmed I/O
- Interrupt driven I/O
- **Hardware Based I/O** **PG # 4**
- None of given

288.It is possible to perform I/O operations from three different methods.

- **True** **PG # 7**
- False

289.The Function of I/O controller is to provide _____.

- I/O control signals
- Buffering
- Error Correction and Detection
- **All of given** **PG # 5**

290.Which of the following are types of ISR _____.

- BIOS (Basic I/O service) ISR
- DOS ISR
- ISR provided by third party device drivers
- **All of the given** **PG # 13**

291.Interrupt service number is usually placed in _____ register.

- CH
- CL
- **AH** **PG # 26**
- AL

خوبصورتی علم و ادب سے ہوتی ہے لباس و حسن سے نہیں

292.NMI Stand for

- **Non Maskable Interrupt** **PG # 46**
- Non Multitude Interrupt
- Non Maskable Instruction
- None of Given

293.A single interrupt controller can arbitrate among ____ different devices.

- 4
- 6
- **8** **PG # 47**
- 10

294. The microprocessor package has many signals for data. Below are some incorrect priority order (Higher to Lower).

- **Reset,Hold,NMI,INTR** **PG # 46**
- NMI, INTR,Hold,Reset
- INTR,NMI,Reset,Hold
- None of the Given

295.The following command "outportb (0x61,inportb(0x61) & 0xFC);" will

- Turn on the speaker
- **Turn off the speaker** **PG # 75**
- Toggle the speaker
- None of the given

296.The following command "outportb (0x61,inportb(0x61) | 3);" will _____ .

- **Turn on the speaker** **PG # 74**
- Turn off the speaker
- Toggle the speaker
- None of the above

297.The PPI acts as an interface between the CPU and a parallel _____ .

- **I/O device** **PG # 83**
- CPU
- BUS
- None of Given

298.BIOS DO NOT support _____.

- LPT1
- LPT2
- LPT3
- **LPT4** **PG # 91**

299._____ bit is cleared to indicate the low nibble is being sent.

- D1
- D2
- D3
- **D4** **PG # 104**

300.The bit _____ of Line control register in UART, if cleared will indicate that DLL is the data register.

- 1
- 3
- 5
- **7** **PG # 114**

301._____ file system is used in NTFS based systems.

- Contiguous Chained
- **Indexed**
- None of the given

302.Communication between keyboard and keyboard controller is _____.

- Asynchronous serial
- **Synchronous serial** **PG # 77**
- Parallel communication
- None of the given

303.In NTFS, boot sector is stored at

- First and 6th sector
- First and Last sector
- Only at Last sector
- **Only at First sector**

304.Standard PC operates in two modes in terms of memory which are

- Real mode and Extended Mode
- Base mode and Memory Mode
- None of the given
- **Real mode and protected mode** **PG # 6**

305.IVT is a table containing _____ byte entries each of which is a far address of an interrupt service routine.

- 2
- **4** **PG # 20**
- 8
- 16

306.Each paragraph in keep function is ____ bytes in size.

- 4
- 8
- **16** **PG # 24**
- 32

307.A software interrupt does not require EOI (End of interrupt.

- **True** **PG # 49**
- False

308.To store each character in keyboard buffer ____ bytes are required.

- **2** **PG # 54**
- 4
- 6
- 8

309.Interrupt _____ is empty; we can use its vector as a flag.

- 9H
- 13H
- 15H
- **65H** **PG # 65**

310.Command register is an _____ bit register

- 4
- **8** **PG # 71**
- 16
- 32

311.How many bytes can be used to store a file name in NTFS?

- 128
- **255**
- 510
- 1024

ہر چیز کی ایک پہچان ہوتی ہے اور عقلمند کی پہچان غوروفکر کرنا ہے اور غوروفکر کی پہچان خاموشی ہے

312.____ is the first logical sector of NTFS partition.

- DPB
- MFT
- **Boot sector**
- None

313.In boot block BIOS parameter block starts from 03H

- 05H
- 08H
- **0BH** **PG # 302**

314.IN NTFS, FAT and root directory is replaced by

- FCB
- **MFT** **PG # 301**
- Hidden blocks
- Boot sector

315.Block # 2 is the safest block to store the backup of boot block.

- True
- **False**

316.The keyboards interface as discussed earlier uses the IRQ0 and the port 64H as data port.

- True
- **False**

317.FAT12 will have 12-bit wide entries and can have 2^12=4096 entries maximum

- **True**
- False

318.In order to produce the sound from PC internal Speaker we have to load the___bit divisor value at the ___port.

- 8, 0x21
- **16, 0x42**
- 32, 0x22
- 64, 0x32

319.DMA stands for_____

- **Direct Memory Access** **PG # 4**
- Distinct Memory Access
- Direct Module Access
- Direct Memory Allocation

320.UART stands for_____

- **Universal Asynchronous Receiver Transmitter** **PG # 107**
- Universal Adjustment and Realigning Tool
- Unconventional Assisted Recovery Team
- None of these

321.Interrupt Vector Table (IVT) in short is a _____ bytes sized table.

- **1024** **PG # 10**
- 2048
- 3072
- 4096

322.Hardware Interrupts are _____.

- Preemptive
- **Non-Preemptive** **PG # 48**
- Both Preemptive and Non-Preemptive
- None of Given

323.Timer interrupt is a _____.

- **Hardware Interrupt** **PG # 28**
- Software Interrupt
- Both of these
- None of These

324.The keyboard makes use of interrupt number _____ for its input operations.

- **9** **PG # 34**
- 10
- 11
- 12

325.Register can be used to divide frequency is _____

- **Counter Register** **PG # 69**
- Accumulator Register
- None of these

326.Which port is known as Data Port_____

- **60H** **PG # 177**
- 61H
- 64H
- 69H

327.LPTs can be swapped.

- **True** **PG # 92**
- False

328.PPI is used to perform parallel communication

- **True** **PG # 81**
- False

329._____is used to control the printer via the BIOS

- Int 16H
- **Int 17H** **PG # 84**
- Int 18H
- Int 19H

330.There are two main types of interrupts namely _____.

- PC based and Window based
- Hardware based and Kernal based
- **Hardware interrupts and Software interrupts** **PG # 10**
- None of the given

331.To set the interrupt vector means is to change the double word sized interrupt vector within the IVT.

- **True** **PG # 22**
- False

332.The service number is usually placed in the _____ register.

- AL
- CL
- **AH** **PG # 26**
- AX

333.The keyboard makes use of interrupt number _____ for its input operations.

- **9** **PG # 34**
- 10
- 11
- 12

ایماندار کو غصہ دیر سے آتا ہے اور جلدی دور ہو جاتا ہے

334.The service _____ is called the keyboard hook service.

- 15H/2FH
- **15H/4FH** **PG # 44**
- 15H/FFH

335.The BIOS interrupt _____ can be used to configure RTC.

- **1AH** **PG # 136**
- 2AH
- 3AH
- 4AH

336.PPI stands for

- Parallel Programmable interface
- **Peripheral Programmable interface** **PG # 76**
- Port Programmable interface
- None of the given

337.Int _____ is used to control the printer via the BIOS.

- **17H** **PG # 84**
- 18H
- 20H
- 21H

338.Counter register can be used to divide clock signal.

- **True** **PG # 69**
- False

دنیا میں سب سے مشکل کام اپنی اصلاح اور سب سے آسان کام دوسروں پر نکتہ چینی کرنا ہے

339.The bit number _____ of the coprocessor control word is the interrupt enable flag.

- **7** **PG # 168**
- 8
- 9
- 6

340.There are _____ kinds of serial communication.

- **2** **PG # 105**
- 3
- 4
- 5

341._____ store the base address for LPT1.

- 40:00H
- 40:02H
- **40:08H** **PG # 92**
- 40:1AH

342.The amount of memory above conventional memory (extended memory can be determined using the service _____.

- **15H/88H** **PG # 162**
- 16H/88H
- 17H/88H
- 21H/88H

343.The output on the monitor is controlled by a controller called _____within the PC.

- **Video controller** **PG # 30**
- Bus controller
- Ram controller
- None of the given

344.Interrupt 9 usually reads the _____ from keyboard.

- ASCII code
- **Scan code** **PG # 34**
- Both ASCII and Scan code
- None of the above

345.NMI Stand for

- **Non Maskable Interrupt** **PG # 46**
- Non Multitude Interrupt
- Non Maskable Instruction
- None of Given

346.A single interrupt controller can arbitrate among ____ different devices.

- 4
- 6
- **8** **PG # 47**
- 10

347.The microprocessor package has many signals for data. Below are some in Correct priority order (Higher to Lower).

- **Reset,Hold,NMI,INTR** **PG # 46**
- NMI, INTR,Hold,Reset
- INTR,NMI,Reset,Hold
- None of the Given

348.The _____function initialize the COM port whose number is passed as parameter using BIOS services.

- Initializecom()
- **Initialize()** **PG # 125**
- Recievechar()
- None of these option

349.There are two type of communication synchronous and Anti Synchronous

- True
- **False** **PG # 105**

350.REGS is a Union

- **True**
- False

351.Keyboard Status Byte is located at the address

- 0040:0000H
- 0040:0013H
- 0040:0015H
- **0040:0017H** **PG # 29**

352.If we use keep (0, 1000) in a TSR program, the memory allocated to it is

- 64000 bytes
- 32000 bytes
- **16000 bytes**
- 80000 bytes

353.Maximum number of interrupts in a standard PC is

- 64
- 128
- **256** **PG #1 0**
- 512

354.The -------- function receive a byte and COM port number is passed as parameter using BIOS service

- Receivebyte ();
- Receive ();
- **Receivechar ();** **PG # 125**
- None of the given option

355._____ whenever receive indicates the start of communication …….. whenever receive indicates the end of communication

- **XON\XOFF** **PG #135**
- XOFF\XON
- XON\YOFF
- YON\XOFF

356._____ is used to set time from RTC

- 1A\02H
- **1A\03H** **PG #138**
- 1A\04H
- 1A\05H

357.Set the Interrupt vector means to change the double word sized interrupt vector within IVT.

- **True** **PG #22**
- False

358.If keyboard buffer is empty the head and tail points at the same location.

- **True** **PG #55**
- False

359.Standard PC can have _____ PPI.

- 1
- **4** **PG #84**
- 8
- 16

360.By cascading two DMAs ____ bits can be transferred.

- 4
- 8
- **16** **PG #186**
- 32

361.PPI interconnection _____ bits is cleared to indicate low nibble is being sent.

- D1
- D2
- D3
- **D4** **PG # 101**

362.Display device (Monitor) performs _____ **I**/O.

- **memory mapped** **PG #30**
- Isolated
- Both of above
- None of these

363.Timer interrupt occurs _____ times every second by means of hardware.

- **18.2** **PG # 28**
- 16.2
- 15.2
- 14.2

364.An I/O device cannot be directly connected to the busses so controller is placed between CPU and I/O.

- **True** **PG # 83**
- False

365.Tail of keyboard should get to get the start of buffer.

- **True** **PG # 55**
- False

366.____ No. of bytes are used to store the character in the keyboard buffer.

- 1
- **2** **PG # 54**
- 4
- 8

367.We have set the bit No. 7 of IMR(Interrupt Mask Register) to unmask the Interrupt so that interrupt _____ can occur at ____ line.

- **0xf ,IRQ 7**
- 0xa, IRQ 6
- 0x8, IRQ 5
- 0x6, IRQ 2

368.If we want to produce the grave voice from speaker phone then we have to load the ____ divisor values at Port ____.

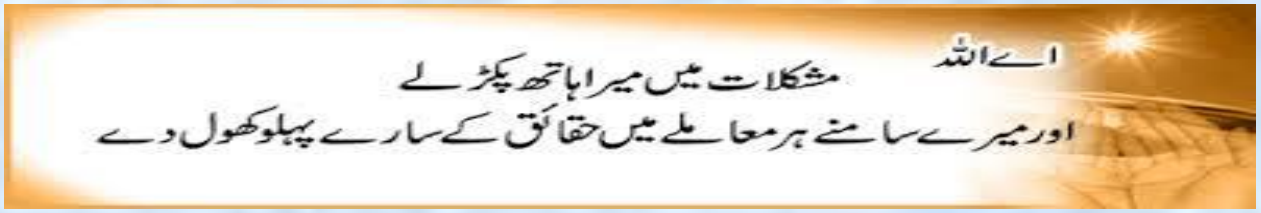
- high, 0x42
- low, 0x22
- high, 0x22
- **low, 0x42**

369.On the execution of IRET instruction, number of bytes popped from stack is

- 4 bytes
- 6 bytes
- **8 bytes** **Click here for detail**
- 10 bytes

370.If CPUID instruction is not present then the processor can be a

- **486 processor** **PG # 166**
- 386 processor
- 286 processor
- All of the above



371.Extended memory is available if the processor is of the type _____

- **AT** **PG # 171**
- XT
- All of the given choices
- None of them

372.The built in mechanism within the UART for error detection is_____

- hamming code
- **parity** **PG # 107**
- CRC16 (cyclic redundancy check 16 bit)
- CRC32 (cyclic redundancy check 32 bit)

373.If three Programmable interrupt controllers are cascaded then how many interrupt driven hardware IO devices can be attached _____

- 12
- 18
- 23
- **24** **PG #48**

374.Int 14H_____ can be used to send a byte

- Service#0
- **Service#1** **PG #121**
- Service#2
- None of the given option.

375.Int 14H _____ can be used to set the line parameter of the UART or COM port.

- **Service # 0** **PG # 119**
- Service # 1
- Service # 2
- None of the given options

376.Int 14H _____ can be used to receive a byte.

- Service # 0
- Service # 1
- **Service # 2** **PG # 121**
- None of the given options

377.The _____ function simply enables the self test facility within the modem control register

- STOn()
- SelfTest()
- **SelfTestOn()** **PG # 127**
- None of these

378._____ is a device incorporated into the PC to update time even if the computer is off.

- Clock counter
- ROM
- Clock
- **Real time clock** **PG # 136**

379.Interrupt _____ is used to get or set the time.

- 0AH
- **1AH** **PG #136**
- 2AH
- 3AH

380._____ is used to set time.

- 1A/02H
- **1A/03H** **PG # 138**
- 1A/04H
- 1A/05H

381.---------- is used to read date from RTC

- 1A\02H
- 1A\03H
- **1A\04H** **PG # 138**
- 1A\05H

382.____whenever received indicates the start of communication and _____ whenever received indicates a temporary pause in the communication.

- **XON & XOFF** **PG # 135**
- XOFF & XON
- XON & YOFF
- YON & XOFF

383.The_____ function uses the COM port number to receive a byte from the COM port using BIOS services.

- recievebyte()
- initialize ()
- receive()
- **recievechar()** **PG # 125**

384.In self test mode the output of the UART is routed to its input.

- **True** **PG # 117**
- False

385.Only _____ ports are important from programming point of view.

- **70 and 71H** **PG # 141**
- 71 and 72H
- 70 and 72H
- 72 and 73H

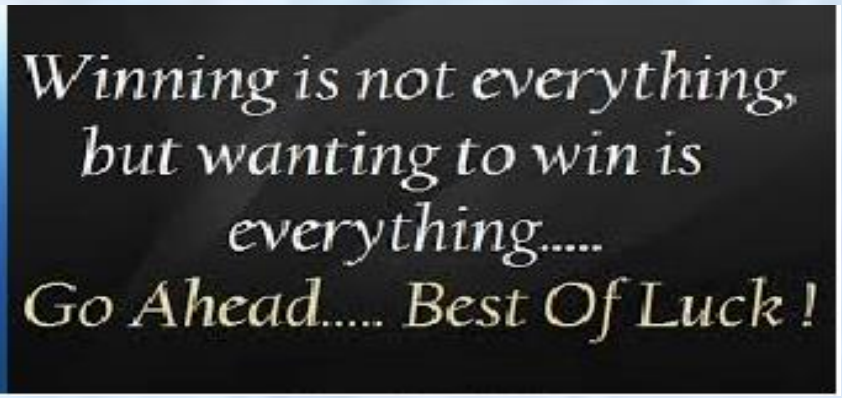
386.The BIOS interrupt 0x1AH can be used to configure real time clock

- **True** **PG # 136**
- False

Note: Give me a feedback and your Suggestion also If you find any mistake in mcqz plz inform me Viva Contact us Page on our Site. And tell me your answer with references.

For More Solved Papers By Arslan Visit Our Website:
Www.VirtualUstaad.blogspot.com







1. DOS command _____ gives the status of the memory and also points out which memory area occupied by which process.

- **mem/d** **PG # 13**
- mem/e
- mem/m
- None of the Given

2. To store each character in keyboard buffer ____ bytes are required.

- **2** **PG # 54**
- 4
- 6
- 8

بری صحبت سے تنہائی بہتر ہے اور تنہائی سے نیک صحبت بہتر ہے

3. The interval timer is used to divide an input frequency.

- **True** **PG #68**
- False

4. The printer interface uses the _____.

- IRQ 6
- **IRQ 7** **PG # 95**
- IRQ 1
- None of given options

5. You can send _____ bits in parallel communication between two PC's at a time.

- 2
- 4
- 6
- **8**

6. In UART the line control register signify the _____.

- Word size and length of stop bits
- parity check and parity type
- Control bit
- **All of given** **PG # 114**

اللہ کا خوف سب سے بڑی دانائی ہے

7. The bit _____ of Line control register in UART, if set indicates that the DLL and DLM will act as the divisor register.

- 1
- 3
- 5
- **7** **PG # 114**

8. Int 14H _____ can be used to set the line parameter of the UART or COM port.

- **Service # 0** **PG #119**
- Service # 1
- Service # 2
- Service # 3

9. _____ is used to read time from RTC.

- **1A/02H** **PG # 137**
- 1A/03H
- 1A/04H
- 1A/05H

10. The _____ signals are used by various devices to request a DMA operation.

- **DREQ** **PG #186**
- REQ
- DRE
- None of the given

ایماندار کو غصہ دیر سے آتا ہے اور جلدی دور ہو جاتا ہے

11. DMA works in _____ modes.

- 2
- **3** **PG #188**
- 5
- 7

DMA Modes Block Transfer, Single Transfer, Demand Transfer

12. DMA Request register can be used to simulate a DMA request through software in case of _____.

- I/O to memory transfer
- None of the given
- **Memory to memory transfer** **PG #192**
- Memory to I/O transfer

13. _____ are the circular division of the disk.

- **Tracks** **PG #201**
- Sectors
- Blocks
- None of given

Tracks are the circular division of the disk and the sectors are the longitudinal division of the disk

14. Blocks are _____.

- **Sectors per track** **PG #202**
- Track per sector
- Cylinder per track
- None of the given

زندگی میں کامیابی کا یہی راز ہے کہ پریشانیوں سے پریشان مت بنو

15. Last ____ bytes of data part in partitioning table are the partition table signature.

- **2** **PG #218**
- 4
- 6
- 8

16. Information about further partitioned drives is available at _____ physical block of extended partition.

- **First** **PG #222**
- Second
- Third
- Fourth

17. _____ will be used to get drive parameters.

- Int 13H/42H
- **Int 13H/48H** **PG #238**
- Int 13H/66H
- Int 13H/66H

18. Inside a boot block operating system name occupies _____ bytes.

- 3
- 5
- 8 **Not sure**
- 11

دنیا کی سب سے بڑی فتح نفس پر قابو رکھنا ہے

19. A programmer wants to access the DPB(Drive Parameter Block) for C drive using undocumented service 21H/32H, what he should place in DL register _____ ?

- **0** **PG #249**
- 1
- 2
- 3

DL – 0 for current Drive

20. Blocks for FAT can be accessed using _____.

- BPB
- DPB
- **FCB**
- Both BPB and DPB.

21. In main memory smallest addressable unit is ____.

- **Byte**
- Nibble
- Word
- None of the given

22. Using the root directory entry and the _____, we can access the contents of File.

- Reserved Blocks
- Number of FAT copies
- **File Allocation Table(FAT)** **PG # 241**
- None of the given

23. If a file is having more than one cluster then it will be managed by _____.

- FAT
- BPB
- DPB
- **None of the given**

24. Practically in FAT 32, total number of entries are _____.

- 2^{26}
- 2^{28}
- 2^{30}
- **2^{32}**

25. _____ keeps the back up of its boot block.

- FAT 12
- FAT 16
- **FAT 32**
- None of the given

26. NTFS can store file name in _____ form.

- **UNICODE** **Click Here For More Detail**
- Both ASCII and UNICODE
- None of the given
- ASCII

27. FAT based file system can store file name in _____ form.

- ASCII
- UNICODE
- **Both ASCII and UNICODE**
- None of the given

28. The LSN of first logical sector of NTFS partition is _____.

- **0** **PG # 240**
- 1
- 2
- 3

29. To minimize _____ defragmenter is used.

- **Seek time** **PG # 316**
- Access time
- Rotational delay
- Both seek and access time

30. In memory map of first 1 MB of RAM, the higher _____ is called system memory.

- 64 KB
- **384 KB** **PG # 317**
- 640 KB
- None of the given

Memory map of the first 1MB of RAM. The first 640KB is called conventional RAM and the higher 384KB is called system memory.

جھوٹ انسان اور ایمان دونوں کا دشمن ہے

31. Extended memory can be accessed in _____ mode.

- Real
- **Protected**
- Both real and protected
- None of the given

32. Extended memory can be accessed only after installing the driver _____ .

- EMM386.EXE
- **HIMEM.SYS** **PG # 318**
- IO.SYS
- None of the given

33. DOS uses _____ memory allocation system.

- **Contiguous** **PG # 325**
- Non Contiguous
- Both Contiguous and Non Contiguous
- None of the given

34. In protected mode segment register will be called _____.

- **Selector** **PG # 326**
- Descriptor
- IDT
- Both Selector and Descriptor

35. There are _____ Control registers in 80386 and above systems.

- 2
- **4** **PG # 331**
- 8
- 16

36. If the CHS (cylinder, head, and sector) address of a disk is known it can be translated in to the _____ address and vice versa.

- Logical
- **LBA** **PG # 213**
- Physical
- None of the Given

37. _____ is a data structure maintained by DOS in the boot block for each drive.

- DPB
- **BPB** **PG # 242**
- FCB
- None of the Given

BIOS parameter block



38. System Stores the DPBs (Drive Parameter Block) for different drives in the form of _____.

- Chain / link list
- Array
- **Table** **Not sure**
- None of Given

39. The extension of a compiled TSR (Terminate and Study Resident) program should be _____

- .EXE
- **.COM**
- .CPP
- .C

40. A single interrupt controller can arbitrate among ____ different devices.

- 4
- 6
- **8** **PG # 47**
- 10

41. The microprocessor package has many signals for data. Below are some in Correct priority order (Higher to Lower).

- Reset,Hold,NMI,INTR
- **NMI, INTR,Hold,Reset** **PG # 46**
- INTR,NMI,Reset,Hold
- None of the Given

42. A standard PC can have _____PPI.

- 1
- **4** **PG # 84**
- 8
- 16

A standard PC can have 4 PPI named LPT1, LPT2, LPT3 and LPT4

43. BIOS DO NOT support _____.

- LPT1
- LPT2
- LPT3
- **LPT4**

44. _____ is used to set date.

- 1A/02H
- 1A/03H
- 1A/04H
- **1A/05H** **PG # 139**

45. If real time clock interrupts microprocessor then interrupt ____ will occur.

- 13h
- 17h
- **70h**
- 71h

46. _____ is LED control byte.

- 0xF3
- **0xED** **PG # 181**
- 0xE5
- 0xFF

47. The DMA requests to acquire buses through the _____ signal.

- **HOLD** **PG # 186**
- ACR
- ACK
- All of the given

48. _____ will specify if the next DMA transfer will happen as a single transfer, block transfer or demand transfer

- DMA Request register
- DMA Mask register
- **DMA Mode register**
- DMA Command register

49. _____ will specify if the channel is to cascade two DMA controllers.

- DMA Request register
- DMA Mask register
- **DMA Mode register**
- DMA Command register

50. _____ are the longitudinal division of the disk.

- Tracks
- **Sectors** **PG # 201**
- Blocks
- None of given

Tracks are the circular division of the disk and the sectors are the longitudinal division of the disk

51. Highest physical capacity of the disk according to the IDE interface is_____.

- 504 MB
- 80 GB
- **127 GB** **PG # 212**
- 300 GB

52. In partition table, first _____ bytes contain code which loads the boot block of active partition.

- 128
- 286
- **446** **PG # 218**
- 500

53. On a single disk there can be _____ different Operating Systems.

- 2
- **4** **PG # 219**
- 6
- 8

عقل مند کہتا ہے میں کچھ نہیں جانتا جبکہ بے وقوف کہتا ہے کہ میں سب کچھ جانتا ہوں

54. _____ is physical or absolute address.

- **LBA** **PG # 240**
- LSN
- CHS
- None of the given

55. The _____ block is the first block on disk.

- LSN =0
- **LBA=0** **PG # 240**
- Both LBA=0 and LSN=0
- None of the given

56. A directory is collection of _____.

- Bios Parameter Blocks
- Drive Parameter Blocks
- **File Control Blocks**
- None of the given

57. First cluster in user data area is numbered _____ in a FAT based systems.

- 0
- 1
- **2** **PG # 258**
- 3

58. Blocks for FAT can be accessed using _____.

- BPB
- DPB
- **FCB**
- Both BPB and DPB.

59. In FAT12 to calculate the address or offset from index, we need to multiply it with ___.

- 1/2
- **3/2** **PG 267**
- 5/2
- 7/2

60. Using the root directory entry and the _____, we can access the contents of File.

- Reserved Blocks
- Number of FAT copies
- **File Allocation Table(FAT)** **PG #269**
- None of the given

61. In FAT16 if FAT entry is between FFF0H to FFF6H then _____.

- Cluster is available
- **It is a Reserved cluster**
- It is next file cluster
- It is a last file cluster

62. _____ needed to store a UNICODE character.

- Nibble
- **2 Bytes**
- 4 Bytes
- Byte

63. _____ used for FCB in FAT 12 and FAT 16.

- Nibble
- Byte
- **2 Bytes**
- 4 Bytes

64. _____ file system is used in CD's.

- None of the given
- **Contiguous**
- Chained
- Indexed

65. FAT based file system can store file name in _____ form.

- ASCII
- UNICODE
- **Both ASCII and UNICODE**
- None of the given

66. In case of _____ contents, the indexes of clusters of file are stored in MFT.

- Small files
- Large files
- **Both small and large files**
- None of the given

67. In memory map of first 1 MB of RAM, the higher _____ is called system memory.

- 64 KB
- **384 KB**
- 640 KB
- None of the given

Memory map of the first 1MB of RAM. The first 640KB is called conventional RAM and the higher 384KB is called system memory

68. PC's operated on 20 bit address bus can operate in _____ mode.

- **Real** **PG # 319**
- Protected
- Both real and protected
- None of the given

خود کو تمہیں سے بڑھ کر کوئی اچھا مشورہ نہیں دے سکتا

69. Int 21H/52H returns the address of DOS internal data structures in ES: BX, the far address of the first MCB in memory lies _____ behind the address returned.

- 2-bytes
- **4-bytes**
- 6-bytes
- 8-bytes

70. In memory management, descriptor entry is of _____ bytes.

- **64**
- 32
- 8
- 16

71. There are _____ Control registers in 80386 and above systems.

- 2
- **4** **PG # 331**
- 8
- 16

72. Control Registers are used for conveying certain control information for _____ Addressing and Co-Processors.

- Real Mode
- **Protected Mode** **PG # 331**
- Virtual Mode
- None of the given

73. The least significant bit of Control register _____ is PE-bit which can be set to enable Protected Mode Addressing.

- **CR0**
- CR1
- CR2
- CR3

74. CD.. command gives the _____ of directory.

- None of the given
- One previous level
- One next level
- **Current path** **PG #282**

75. Cluster size is variable and can be measured in power of _____.

- **2**
- 16
- 4
- 8

کسی انسان کی خوبی کو پہچانو اور اسے بیان کرو، لیکن اگر کسی کی خامی مل جائے تو یہاں تمہاری خوبی کا امتحان ہے۔

فرمان حضرت علیؓ

76. Who notifies the end of interrupt (EOI) to the programmable interrupt controller?

- Programmer
- **Microprocessor**
- Operating system
- None of the given

77. Choose the correct statement to notify the interrupt controller about end of interrupt (EOI).

- Inportb(0x20,0x20)
- **Outportb(0x20,0x20)**
- Outportb(0x30,0x22)
- None of the given

78. Information about Pending interrupts are held by _____.

- Interrupt Service Register
- Interrupt Mask Register
- None of the Given
- **Interrupt Request Register**

79. BPB stands for _____.

- **BIOS parameter block**
- BIOS processing block
- Base processing block
- BIOS partition block

80. The input frequency used by the interval timer is the _____signal generated by the clock generator.

- None of Given
- **PCLK**
- ACL
- ICP

81. BIOS DO NOT support _____.

- LPT1
- LPT2
- LPT3
- **LPT4**

82. DTE is _____.

- **Data terminal equipment** **PG #109**
- Data transmitting equipment
- Dual terminal equipment
- None of the given.

83. Int 14H _____ can be used to send a byte.

- Service # 0
- **Service # 1**
- Service # 2
- Service # 3

84. A single DMA can transfer _____ operands to and from memory in a single a bus cycle.

- **8 bit**
- 16 bit
- 32 bit
- 64 bbit

85. DMA does not work in _____ mode.

- **Double Mode**
- Block Transfer
- Demand Transfer
- None of the given

86. An addressable unit on disk can be addressed by three parameters _____.

- **head #, sector # and track #**
- HD #, AH# and track #
- Hd0 #, sector # and track #
- None of the given

87. Each addressable unit has a unique combination of sec#, head# and track# as its _____ address.

- Both Physical and Logical
- None of the given
- **Physical**
- Logical

جو لوگوں کے سامنے فخر کرتا ہے وہ لوگوں کی نظروں سے گر جاتا ہے

88. In partition table, first _____ bytes contain code which loads the boot block of active partition.

- 128
- 286
- **446**
- 500

89. On a single disk there can be _____ different Operating Systems.

- 2
- **4**
- 6
- 8

90. The last partition table within the chain contains _____ entry/entries about the logical drive.

- **Single**
- Double
- Three
- Four

91. Partition Table can be read using the extended _____ Services.

- 13H
- **14H**
- 15H
- 17H

92. File control block (FCB) is _____ byte long.

- 8
- 16
- **32**
- 64

93. _____ is collection of contagious blocks.

- **Cluster**
- Sector
- File
- Directory

94. A FAT 12 table can have maximum _____ values.

- 1024
- 2048
- **4096**
- None of given

2^12 entries maximum = 4096

95. The first cluster number of file can be found in _____.

- BPB
- DPB
- **FCB** **PG # 265**
- None of the given

96. In FAT12 if FAT entry is 000H then _____.

- It is a last file cluster
- Cluster is available
- **It is a Reserved cluster**
- It is next file cluster

97. Total number of clusters of FAT12 are _____.

- FF0 H
- FFF H
- FEF H
- **FEE H** **PG # 266**

98. When we mark a file as deleted by placing 0xE5 then the chain of clusters in FAT is also replaced by _____.

- E5
- **0**
- 1
- N

99. To store a cluster in FAT 32 _____ is/are needed.

- Byte
- 2 Bytes
- **4 Bytes**
- Nibble

100. _____ keeps the back up of its boot block.

- FAT 12
- FAT 16
- **FAT 32**
- None of the given

101. Up to _____ bytes can be used to store a file name in NTFS.

- 30
- 126
- **254**
- 510

102. First logical sector of NTFS partition is _____.

- DPB
- MFT
- **Boot sector**
- None of the given

103. The LSN of first logical sector of NTFS partition is _____.

- **0**
- 1
- 2
- 3

104. In case of _____ the contents of file are stored in MFT.

- Small files
- Large files
- **Both small and large files**
- None of the given

105. Interrupt Descriptor Table can have up to _____ entries.

- 64
- 128
- **256**
- 512

106. There can be _____ different descriptors privilege levels.

- 2
- **4**
- 8
- 10

107. The least significant bit of Control register _____ is PE-bit which can be set to enable Protected Mode Addressing.

- **CR0** **PG # 331**
- CR1
- CR2
- CR3

108. Entry point of execution in EXE File can be _____.

- From first instruction
- From last instruction
- **Anywhere in the program** **PG # 335**
- From anywhere in the middle

109. In NTFS, Block # ____ is the safest block to store the back up of boot block.

- 8
- **10** **Not sure**
- 20
- 6

110. Size of single entry in Partition Table is _____.

- **512 bytes**
- 128 bytes
- 64 bytes
- 16 bytes

111. MBR partition table can have _____ entries at maximum.

- 1
- 2
- 3
- **4** **Not sure**

112. _____ is a data structure maintained by DOS in the boot block for each drive.

- DPB
- **BPB** **PG # 242**
- FCB
- None of the Given

113. Hidden blocks between the first physical block on each partition are used by _____ for storing of data

- User
- Operating System
- BIOS
- **None of the Given**

114. Which of the following data structure is NOT used in Operating System's File management?

- **Memory Control Block** **PG # 06**
- File Allocation Table
- Drive Parameter Block
- File Control Block

115. Interrupt procedures are _____ procedures.

- **Reentrant** **PG # 38**
- Non-Reentrant
- Recursive
- Virtual

116. The extension of a compiled TSR (Terminate and Study Resident) program should be _____

- .EXE
- **COM**
- .CPP
- .C

to make a .C program a TSR program, you are required to generate a .COM file by using the following command in DOS prompt.

bcc –EEXAMPLE.COM –IC:\BORLANDC\INCLUDE –LC:\BORLANDC\LIB EXAMPLE.C

Here, we are generating EXAMPLE.COM for of EXAMPLE.C program.

117. Choose the correct statement to notify the interrupt controller about end of interrupt (EOI).

- Inportb(0x20,0x20)
- **Outportb(0x20,0x20)**
- Outportb(0x30,0x22)
- None of the given

118. Information about Pending interrupts are held by _____.

- Interrupt Service Register
- Interrupt Mask Register
- **Interrupt Request Register**
- None of the Given

عقل مند اپنے عیب خود دیکھتا ہے اور بیوقوفوں کے عیب دنیا دیکھتی ہے

119. _____ Procedure can have parameters.

- Hardware Interrupt
- Software Interrupt
- **Both Hardware and software Interrupt** **PG # 10**
- None of the Given

120. The Address of partition block on hard disk is _____.

- head # =0, track # = 0 and sector # = 0
- head # =0, track # = 0 and sector # = 1
- head # =0, track # = 1 and sector # = 1
- **head # =1, track # = 0 and sector # = 1** **PG # 42**

121. NMI Stand for _____

- **Non Maskable Interrupt** **PG #46**
- Non Multitude Interrupt
- Non Maskable Instruction
- None of Given

122. The interval timer is used to divide an input frequency.

- **True** **PG # 68**
- False

بدصورت چہرہ بدصورت دماغ سے بہتر ہے

123. The PPI acts as an interface between the CPU and a parallel _____ .

- **I/O device** **PG # 86**
- CPU
- BUS
- None of Given

124. BIOS DO NOT support _____.

- LPT1
- LPT2
- LPT3
- **LPT4**

125. _____ is used to identify the cause of interrupt.

- Interrupt Enable register
- **Interrupt ID register**
- Interrupt Status register
- None of the given

126. The bit _____ of Line control register in UART, if cleared will indicate that DLL is the data register.

- 1
- 3
- 5
- **7**

127. Int 14H _____ can be used to send a byte.

- Service # 0
- **Service # 1** **PG # 121**
- Service # 2
- Service # 3

Service of 14h include service #1 which is used to send a byte and service #2 which is used to receive a byte

128. To access bettery powered RAM, only _____ ports are important from programming point of view.

- **70 and 71H** **PG # 141**
- 71 and 72H
- 70 and 72H
- 72 and 73H

129. The DMA requests to acquire buses through the _____ signal.

- **HOLD** **PG # 186**
- ACR
- ACK
- All of the given

130. _____ used to program various common parameters of transfer for all the channels.

- None of the above
- DMA Status Register
- **DMA Command Register** **PG # 191**
- DMA Request Register

131. Bit # _____ of mode register in DMA determine the direction of a transfer.

- **2**
- 3
- 4
- 5

132. _____ will specify if the next DMA transfer will happen as a single transfer, block transfer or demand transfer.

- DMA Request register
- DMA Mask register
- **DMA Mode register**
- DMA Command register

133. Each addressable unit has a unique combination of sec#, head# and track# as its _____ address.

- Both Physical and Logical
- None of the given
- **Physical** **PG # 202**
- Logical

134. Highest capacity of disk can be accessed using BIOS functions is _____.

- 128 MB
- 256 MB
- **504 MB** **PG # 211**
- 127 GB

عقل مند آدمی اس وقت تک نہیں بولتا جب تک خاموشی نہیں ہو جاتی

135. DOS has built in limit of _____ blocks per cluster.

- 32
- 64
- **128** **PG # 242**
- 256

136. BIOS Parameter block is situated in _____ Block.

- **Boot** **PG # 242**
- Data
- Extended Data
- None of Given

137. The _____ block is the first block on disk.

- LSN =0
- **LBA=0** **PG # 240**
- Both LBA=0 and LSN=0
- None of the given

138. Inside a boot block jump code part occupies _____ byte.

- 5
- 8
- 11
- **3** **PG # 244**

بہترین تجربہ وہ ہے جس سے نصیحت حاصل ہو

139. In boot block BIOS parameter block starts from _____.

- 03H
- 05H
- 08H
- **0BH**

140. In main memory smallest addressable unit is ____.

- **Byte**
- Nibble
- Word
- None of the given

141. Total number of clusters of FAT12 are _____.

- FF0 H
- FFF H
- FEF H
- **FEE H** **PG # 266**

142. In FAT12 to calculate the address or offset from index, we need to multiply it with ___.

- 1/2
- **3/2** **PG# 267**
- 7/2
- 5/2

143. Total number of cluster of FAT16 is _____.

- FFF0 H
- FFFF H
- FFEF H
- **FFEE H** **Page 272**

144. Total ____ fragments can be supported for storing long file names.

- 26
- 28
- **32**
- 48

145. _____ used for FCB in FAT 12 and FAT 16.

- Nibble
- Byte
- **2 Bytes**
- 4 Bytes

146. To store a cluster in FAT 32 _____ is/are needed.

- Byte
- 2 Bytes
- **4 Bytes**
- Nibble

147. Practically in FAT 32, total number of entries are _____.

- 2^{26}
- 2^{28}
- 2^{30}
- **2^{32}**

148. First logical sector of NTFS partition is _____.

- DPB
- MFT
- **Boot sector**
- None of the given

149. Extended memory can be accessed in _____ mode.

- Real
- **Protected**
- Both real and protected
- None of the given

150. MCB is a _____ bytes data structure.

- 8
- **16** **PG # 321**
- 32
- 64

151. Real mode does not support _____ memory allocation system.

- **Contiguous**
- Non Contiguous
- Both Contiguous and Non Contiguous
- None of the given

152. There can be _____ different descriptors privilege levels.

- 2
- **4**
- 8
- 10

153. In memory management, descriptor entry is of _____ bytes.

- 8
- 16
- 32
- **64**

154. For virus to propagate itself, it has to intercept interrupt _____.

- 9H
- 11H
- **13H**
- 17H

خوبصورتی علم و ادب سے ہوتی ہے لباس و حسن سے نہیں

155. If virus wants to be in memory independently, it should have its own _____.

- MCB
- PSP
- EB
- **Both MCB and PSP** **PG #334**

156. Size of single entry in Partition Table is _____.

- **512 bytes** **PG#218**
- 128 bytes
- 64 bytes
- 16 bytes

157. _____ is a data structure maintained by DOS in the boot block for each drive.

- DPB
- **BPB** **PG#242**
- FCB
- None of the Given

158. Cluster size is variable and can be measured in power of _____.

- **2** **PG#242**
- 16
- 4
- 8

159. Bit # _____ of Eflag is used for alignment check

- 12
- 14
- 15
- **18** **PG # 164**

160. Each addressable unit has a unique combination of sec#, head #, track # as its _____ address.

- **Physical** **PG #202**
- Logical
- Both
- None

161. First cluster in user data is numbered in a FAT based system.

- 0
- 1
- **2** **PG # 258**
- 3

162. BIOS services understand --------.

- **LBA** **PG # 212**
- LSN
- Cluster #
- None

163. The first cluster number of a file can be found in---------

- BPB
- DPB
- **FCB** **PG # 265**
- None

164. The size of FS Info block is

- 64byte
- 128 byte
- 256 byte
- **512 byte** **PG # 300**

165. In NTFS first -------- entries are reserved.

- 4
- 6
- **16** **PG # 303**
- 32

166. In memory map of first 1 MB of ram ,the first -------- is called conventional RAM.

- 64kb
- 384kb
- **640kb** **PG # 317**
- None

167. In memory map of first 1 MB of ram ,the higher -------- is called system memory.

- 64kb
- **384kb** **PG # 317**
- 640kb
- None

168. The -------- of boot block constitutes of BPB.

- Code part
- **Data part** **PG # 242**
- Both
- None

169. Extended BIOS function make use of --------- address

- **LBA** **PG # 212**
- CHS
- LSN
- None

170. LBA address can be used in place of the CHS address.

- **True** **PG # 235**
- False

171. In FAT12, the maximum range of clusters is ……...

- 0 ~ FEFH
- 1~ FEFH
- **2 ~ FEFH** **PG # 266**
- 3 ~ FEFH

172. NTFS volume can be accessed directly in DOS.

- True
- **False** **PG # 310**

173. Each partition information chunk is 16 bytes long and the last two bytes at the end of the partition table data part is the partition table signature whose value should be _____ indicating that the code part contains valid executable code.

- 00AA
- 0055
- 050A
- **AA55** **PG # 219**

174. Service 21H/52H service returns the address of DOS internal data structures in ES: BX _____behind the address returned lies the far address of the first MCB in memory.

- 2-bytes
- **4-bytes** **PG # 322**
- 6-bytes
- 8-bytes

175. 80386 can have _____ control registers.

- 2
- 5
- 3
- **4** **PG # 331**

176. The partition table uses the extended _____ service.

- **13H** **PG # 234**
- 14H
- 15H
- 16H

177. The entry point of execution in EXE File can be

- Start of the first instruction
- Start of the last instruction
- **Anywhere in the Program** **PG # 335**
- Can be in the middle of the program

انسان کے لئے بری صحبت سے بڑھ کر بری کوئی چیز نہیں

178. Using the____ entry and the FAT we can access the contents of file.

- Reserved blocks
- **Root Directory** **PG # 269**
- Number of FAT copies
- None of the given

179. Control information in files is maintained using

- BPB
- DPB
- **FCB** **PG # 256**
- FPB

180. What will happen if NTFS volume is accessed in DOS?

- Convert it to FAT volume
- Nothing will happen
- **Error of invalid media** **PG # 310**
- None of the given

181. LSN of FS Info block is available at

- BPB
- **FAT**
- Root Directory
- None of the given

خاموشی غصے کا بہترین علاج ہے

182. DOS device drivers do not understand the_____ data structures.

- FAT12
- FAT16
- FAT32
- **NTFS** **PG # 310**

183. _____ is a collection of contagious blocks.

- **Cluster** **PG # 242**
- Sector
- Byte
- None of Given

184. _____ used to determine the amount of conventional memory interfaced with the processor in kilobytes.

- INT 10 H
- INT 11 H
- **INT 12 H** **PG # 162**
- INT 13 H

185. Bit number _____ of coprocessor control word is the Interrupt Enable Flag.

- **7** **PG #168**
- 8
- 9
- 10

186. To distinguish 486 with Pentium CPUID Test is used.

- **True** **PG # 166**
- False

187. Practically _____ entries are there in FAT 32.

- 2^26
- 2^28
- 2^30
- **2^32** **PG # 265**

188. BPB stands for _____.

- **BIOS parameter block** **PG # 243**
- BIOS processing block
- Base processing block
- BIOS partition block

189. The keyboard input character scan code is received at ___ port.

- **60H** **PG # 179**
- 61H
- 62H
- 63H

190. _____ is LED control byte.

- 0xFD
- **0xED** **PG #181**
- 0xFF
- 0xEE

جھوٹ رزق کو کھا جاتا ہے

191. _____means typematic rate will be sent in next byte.

- **0xF3 PG # 180**
- 0xF4
- 0xF5
- 0xF6

192. Keyboard uses port ____ as status port.

- **64H PG # 177**
- 66H
- 67H
- 69H

193. The keyboard can perform _____ serial I/O.

- asynchronous
- **synchronous**
- Multiple
- Single

194. Bit number 2 of port 64H Status register used for output buffer full.

- True
- **False**

195. Bit number _____ can declares the parity error of port 64H Status register.

- 4
- 5
- 6
- **7**

196. Bit number _____ of port 64H Status register used for input buffer full.

- **0**
- 1
- 2
- 3

197. Disadvantage of FAT32 is _____.

- Large disk size can be managed in FAT32
- Cluster size is reduced
- Internal fragmentation is reduced
- **Very large table** **PG # 299**

198. Maximum possible entries in FAT12 are _____.

- 1024
- 2048
- **4096** **PG # 264**
- 65536

199. What will be the value of the word located at 1Fh in DPB when number of free clusters on drive is not known?

- 0000H
- 1111H
- **FFFFH** **PG # 250**
- None of the given.

افضل انسان وہ ہے جو اپنی اصلاح کی کوشش کرتا ہے

200. Jump code part contains ____ bytes in boot block.

- **3** **PG # 302**
- 5
- 8
- 11

201. Operating system name contains ____ bytes in boot block.

- 3
- 5
- **8** **PG # 257**
- 11

202. File can be _____ viewed as organization of data.

- Physically
- **Logically** **PG # 256**
- Both logically and physically
- None of the give

203. _____ is used to read a block against its LSN.

- **absread()** **PG # 247**
- abswrite()
- lsnread()
- None of the given

اطمینان قلب چاہتے ہو تو حسد سے دور رہو

204. File can be_____ viewed as collection of clusters or blocks.

- **Physically** **PG # 256**
- Logically
- Both physically and logically
- None

205. When we talk about FAT based file system, in user data area first cluster number is _____.

- 0
- 1
- **2** **PG # 258**
- None of the given

206. Cluster number can also be referred as block number.

- True
- **False** **PG # 258**

207. To access the block within cluster using BIOS services the cluster number should be converted into _____.

- CHS
- LBA
- **LSN** **PG # 258**
- None of the given

208. What will be the value of DL register when we are accessing C drive using undocumented service 21H/32H?

- 0
- 1
- 2
- **3** **PG # 249**

209. The directory structure of DOS is like _____.

- Array
- **Tree** **PG # 256**
- Linked list
- None of the given

210. Control information about files is maintained using _____.

- BPB
- DPB
- **FCB** **PG # 256**
- FPB

211. When LSN is equal to zero (0), it means _____.

- First block of the disk
- **First block of the logical drive** **PG # 240**
- First block of hidden blocks
- None of the given

212. In FAT32, lower _____ bits are used.

- 26
- **28** **PG # 292**
- 30
- 32

213. _____ is relative address with respect to the start of Logical Drive.

- LBA
- **LSN** **PG #240**
- CHS
- None of the given

214. The practical limit of blocks per cluster is _____.

- 32 blocks per cluster
- **64 blocks per cluster** **PG #242**
- 128 blocks per cluster
- 256 blocks per cluster

215. In dos we have limit of _____.

- **128 blocks per cluster** **PG #242**
- 256 blocks per cluster
- 32 blocks per cluster
- 64 blocks per cluster

216. Highest capacity physical capacity of the disk according to the IDE interface is _____.

- **127 GB** **PG # 212**
- 100 GB
- 80 GB
- 300 GB

217. Partition Table can be read using the extended _____ Services.

- **13 H** **PG # 234**
- 14 H
- 15 H
- None of given

218. In Protected Mode, the segment registers are used as _____

- Descriptor
- **Selector** **PG # 326)**
- All of the given choices
- None of the given choices

219. To access drive parameter block we use undocumented service _____

- 09H/32H
- 11H/32H
- 17H/32H
- **21H/32H** **PG # 249**

220. _____ is an absolute address relative to the start of physical drive.

- **LBA** **PG # 240**
- LSN
- CHS
- None of the above

221. Boot block consists of _____ bytes.

- 64
- 128
- 256
- **512** **PG # 242**

222. The DMA requests to acquire buses through the _____ signal.

- **HOLD** **PG #186**
- ACR
- ACK
- None of Given

اس سے پہلے کہ تمہیں شہوت فتنے میں ڈالے نکاح کرلو

223. The keyboard device writes a code 0xFA on the port 60H to indicate that the _____.

- Input buffer is full
- **Byte has been received properly** **PG # 179**
- Output buffer is full
- None of the given

224. A single DMA can transfer _____ operands to and from memory in a single a bus cycle.

- **8-bits** **PG # 186**
- 16-bits
- 32-bits
- 12-bits

225. In FAT12, to calculate the address or offset from index, we need to multiply it with ___.

- 1/2
- **3/2** **PG #267**
- 5/7
- 7/2

226. _____ Register can be used to show that the channel is single transfer, block transfer or demand transfer mode.

- DMA Command register
- DMA Request Register
- DMA Mode Register
- **DMA controller Register** **PG #187-188**

اپنی مرضی اور اللہ کی مرضی میں فرق کا نام غم ہے

227. When we mark a file as deleted by placing 0xE5 then the chain of clusters in FAT is also replaced by _____.

- E5
- 1
- **0** **PG # 79**
- N

228. Cluster size is reduced in _____.

- FAT12
- FAT16
- **FAT32** Click here for detail
- None of the given

229. In FAT32 _____ root directory entries are there.

- 128
- 256
- 512
- **None of the given** Click here for detail

230. If a file is having more than one cluster then it will be managed by _____.

- FAT
- BPB
- DPB
- **None of the above**

بری صحبت سے تنہائی بہتر ہے اور تنہائی سے نیک صحبت بہتر ہے

231. Internal fragmentation is reduced in _____.

- FAT12
- **FAT16** **Click here for detail**
- FAT32
- None of the given

232. For supporting long file names, _____ fragments can be supported.

- 12
- 20
- 26
- **32**

233. To store a cluster in FAT 32 _____ is/are needed.

- Nibble
- Byte
- 2 Bytes
- **4 Bytes** **Click here for detail**

234. If a file size is 12K and the size of the cluster is 4K then _____ clusters are used for the file.

- 2
- **3**
- 4
- 5

ایماندار کو غصہ دیر سے آتا ہے اور جلدی دور ہو جاتا ہے

235. We can access the contents of File by using the root directory entry and _____.

- Reserved Blocks
- Number of FAT copies
- **File Allocation Table (FAT)** **PG # 269**
- None of the given

236. FAT based file system can store file name in _____ form.

- ASCII
- UNICODE
- **Both ASCII and UNICODE**
- None of the given

237. Drive parameter block is derived from _____.

- FCB
- FAT
- **BPB** **PG # 249**
- CPB

238. We can access Blocks for FAT using _____.

- BPB
- DPB
- **FCB**
- Both BPB and DPB

جھوٹ انسان اور ایمان دونوں کا دشمن ہے

239. If we know the cluster number, we can access the blocks within the cluster using BIOS services directly.

- **True** **PG # 258**
- False

240. _____ is an internal data structure of DOS and resides in main memory.

- BPB
- DPB
- CPB
- **None of the given**. **Click here for detail**

241. The size of DPB data structure is _____ bytes.

- 16
- 32
- 64
- **128** **click here for detail**

242. The size of FCB data structure is _____ bytes.

- **16** **Click here for detail**
- 32
- 64
- 128

243. Advantages of FAT32 is/are _____.

- Large disk size can be managed in FAT32
- **Cluster size is reduced** Click here for detail
- Internal fragmentation is reduced
- All of the given

ایماندار کو غصہ دیر سے آتا ہے اور جلدی دور ہو جاتا ہے

244. _____ file system keeps the backup of its boot block.

- FAT12
- FAT16
- **FAT32** **Click here for detail**
- None of the given

245. To store a UNICODE character ____ is/are needed.

- Nibble
- Byte
- **2 Bytes** **Click here for detail**
- 4 Bytes

246. _____ is the first block on disk.

- LSN =0
- **LBA=0** **PG # 240**
- LBA=1
- Both LBA=0 and LSN=0

247. If FAT entry is between FFF0H to FFF6H in FAT16 then _____.

- Cluster is available
- **It is a Reserved cluster** **PG # 272**
- It is next file cluster
- It is a last file cluster

خود کو تمھیں سے بڑھ کر کوئی اچھا مشورہ نہیں دے سکتا

248. File system used in CD's is_____ file system

- **Contiguous** **Click here for detail**
- Chained
- Indexed
- None

249. A file has 2 clusters and the size of cluster is 4K. What will be the size of file?

- 2K
- **8K**
- 16K
- 32K

250. In NTFS, Backup of boot block is stored at block # _____.

- 2
- 6
- **8**
- 10

251. The interval timer can operate in _____ modes.

- Five
- Seven
- Four
- **Six** **PG # 72**

252. File control block (FCB) is _____ byte long.

- **32** **Click here for detail**
- 64
- 16
- 128

253. DOS command _____ which gives the status of the memory and also points out which memory area occupied by which process.

- **mem/d** **PG # 13**
- mem/e
- mem/m
- None of the given

254. Each entry in the IVT is _____ in size.

- **4-bytes** **PG # 12**
- 6-bytes
- 8-bytes
- 2-bytes

255. The tail value of the keyboard buffer should be examined to get to the _____ of the buffer.

- **Start** **PG # 56**
- End
- Middle
- None of given

256. Usually interrupt procedures are reentrant procedures especially those interrupt procedure compiled using C language compiler are reentrant.

- **True** **PG # 38**
- False

جو شخص ناکامیوں سے ڈر کر بھاگتا ہے کامیابی اُس سے ڈر کر بھاگتی ہے

257. _____ is Disk interrupt.

- 10H
- 11H
- **13H** **PG # 42**
- 14H

258. In parallel communication, the maximum numbers of bits we can send between two computers are _____.

- 2-bits
- 4-bits
- 6-bits
- **8-bits**

259. 14h include _____ which is used to send a byte.

- Service #0
- **Service #1** **PG # 121**
- Service #2
- Service #3

260. The status register _____ is the main control register.

- **B** **PG # 146**
- A
- C
- D

261. _____ is used to identify the cause of interrupt.

- **Interrupt ID Register** **PG # 116**
- PC Register
- AC Register
- None of All These

262. In NTFS, up to _____ characters are used to store files names,

- 30
- 48
- **255** **PG # 283**
- 510

263. A cluster is a collection of contiguous _____.

- **Blocks** **PG # 242**
- Sectors
- Bytes
- None of Given

264. In BPB, root directory is saved in _____.

- **Cluster#0**
- Cluster#1
- Cluster#2
- Cluster#3

265. In NTFS, total sizes of MFT entries are _____.

- **16-bytes** **PG # 303**
- 20-bytes
- 26-bytes
- 32-bytes

266. In NTFS, _____ store the contents of file.

- **Both small & large file Record**
- Small record
- Large Record
- None of given

267. In NTFS, contents and indexed of file is stored in _____.

- Small record
- Large Record
- Both small & large file Record
- **None of given**

268. Total No. of bytes that can be stored in Keyboard Buffer is____.

- 16
- **32 PG #54**
- 64
- 128

269. BIOS support _____UARTS as COM ports.

- 6
- **4 PG #113**
- 3
- 2

270. DCE stands for _____.

- **Data communication equipment PG # 109**
- Distributed Computing Environment
- Data Communications Equipment
- Data Carrier Equipment

271. In counter register bit no. 3 changes its value between 0 and 1 with in ____clock cycles

- 1
- 2
- 4
- **16 PG #69**

272. In _____each byte is needed to be encapsulated in start and end.

- Synchronous communication
- **Asynchronous communication** **PG # 106**
- Both
- None of given

273. The _____ service # is not used in any interrupt.

- 01
- 02
- 03
- **FF**

274. If we want to send printing on the printer then we have to perform following steps.

- Initialize printer
- Read Status
- Check Error
- **All of the given**

275. If printer is _____ then printer sends back the ACK signal to the printer interface

- **idle** **PG # 97**
- busy
- Out of paper
- None of the given

276. DSR stands for _____.

- **Data set ready** **PG # 111**
- Data service ready
- Data stock ready
- None of the given

277. At IRQ 7 Interrupt # ___ is used.

- 0x0A
- 0x0B
- 0x0C
- **0x0F PG # 95**

278. The memory addresses of COM ports remain same for all computers

- True
- **False**

279. In keyboard status byte bit no. 2 and 3 are used for ctrl and alt keys respectively. which of the following condition is used to check that Ctrl + Alt keys are pressed. Where: unsigned char far * scr = (unsigned char far *)(0x00400017);

- **if (((*scr)&12)==12)**
- if (((*scr)&8)==8)
- if (((*scr)&4)==4)
- if (((*scr)&2)==2)

280. In case of synchronous communication a timing signal is required to identify the start and end of a bit.

- **True PG # 105**
- False

281. The baud rate is set in accordance with the divisor value loaded within the UART internal registers base +0 and base +1.

- **True PG # 114**
- False

عقل مند آدمی اس وقت تک نہیں بولتا جب تک خاموشی نہیں ہو جاتی

282. Software based flow control make use of -------- control characters

- Xon
- XOFF
- **Both** **PG # 135**
- None

283. _____ is used to read time from RTC

- **1A\02H** **PG # 137**
- 1A\03H
- 1A\04H
- 1A\05H

284. Int_____ service 0 can be used to set the line parameter of the UART or COM port.

- **14H** **PG # 119**
- 15H
- 13H
- None of the given option

285. When LBA is equal to zero (0), it means _____.

- **First block of the disk** **PG # 240**
- First block of the logical drive
- First block of the hidden block
- None of the given

286. In IRQ2 and IRQ3 which one has the highest priority?

- Can't be determined
- Both have same priority
- IRQ3
- **IRQ2** **PG # 47**

287. Following is not a method of I/O

- Programmed I/O
- Interrupt driven I/O
- **Hardware Based I/O** **PG # 4**
- None of given

288. It is possible to perform I/O operations from three different methods.

- **True** **PG # 7**
- False

289. The Function of I/O controller is to provide _____.

- I/O control signals
- Buffering
- Error Correction and Detection
- **All of given** **PG # 5**

290. Which of the following are types of ISR _____.

- BIOS (Basic I/O service) ISR
- DOS ISR
- ISR provided by third party device drivers
- **All of the given** **PG # 13**

291. Interrupt service number is usually placed in _____ register.

- CH
- CL
- **AH** **PG # 26**
- AL

خوبصورتی علم و ادب سے ہوتی ہے لباس و حسن سے نہیں

292. NMI Stand for

- **Non Maskable Interrupt** **PG # 46**
- Non Multitude Interrupt
- Non Maskable Instruction
- None of Given

293. A single interrupt controller can arbitrate among ____ different devices.

- 4
- 6
- **8** **PG # 47**
- 10

294. The microprocessor package has many signals for data. Below are some incorrect priority order (Higher to Lower).

- **Reset,Hold,NMI,INTR** **PG # 46**
- NMI, INTR,Hold,Reset
- INTR,NMI,Reset,Hold
- None of the Given

295. The following command "outportb (0x61,inportb(0x61) & 0xFC);" will

- Turn on the speaker
- **Turn off the speaker** **PG # 75**
- Toggle the speaker
- None of the given

296. The following command "outportb (0x61,inportb(0x61) | 3);" will _____ .

- **Turn on the speaker** **PG # 74**
- Turn off the speaker
- Toggle the speaker
- None of the above

297. The PPI acts as an interface between the CPU and a parallel _____ .

- **I/O device** **PG # 83**
- CPU
- BUS
- None of Given

298. BIOS DO NOT support _____.

- LPT1
- LPT2
- LPT3
- **LPT4** **PG # 91**

299. _____ bit is cleared to indicate the low nibble is being sent.

- D1
- D2
- D3
- **D4** **PG # 104**

300. The bit _____ of Line control register in UART, if cleared will indicate that DLL is the data register.

- 1
- 3
- 5
- **7** **PG # 114**

301. _____ file system is used in NTFS based systems.

- Contiguous Chained
- **Indexed**
- None of the given

302. Communication between keyboard and keyboard controller is _____.

- Asynchronous serial
- **Synchronous serial** **PG # 77**
- Parallel communication
- None of the given

303. In NTFS, boot sector is stored at

- First and 6th sector
- First and Last sector
- Only at Last sector
- **Only at First sector**

304. Standard PC operates in two modes in terms of memory which are

- Real mode and Extended Mode
- Base mode and Memory Mode
- None of the given
- **Real mode and protected mode** **PG # 6**

305. IVT is a table containing _____ byte entries each of which is a far address of an interrupt service routine.

- 2
- **4** **PG # 20**
- 8
- 16

306. Each paragraph in keep function is ____ bytes in size.

- 4
- 8
- **16** **PG # 24**
- 32

307. A software interrupt does not require EOI (End of interrupt.

- **True** **PG # 49**
- False

308. To store each character in keyboard buffer ____ bytes are required.

- **2** **PG # 54**
- 4
- 6
- 8

309. Interrupt _____ is empty; we can use its vector as a flag.

- 9H
- 13H
- 15H
- **65H** **PG # 65**

310. Command register is an _____ bit register

- 4
- **8** **PG # 71**
- 16
- 32

311. How many bytes can be used to store a file name in NTFS?

- 128
- **255**
- 510
- 1024

ہر چیز کی ایک پہچان ہوتی ہے اور عقلمند کی پہچان غوروفکر کرنا ہے اور غوروفکر کی پہچان خاموشی ہے

312. ____ is the first logical sector of NTFS partition.

- DPB
- MFT
- **Boot sector**
- None

313. In boot block BIOS parameter block starts from 03H

- 05H
- 08H
- **0BH** **PG # 302**

314. IN NTFS, FAT and root directory is replaced by

- FCB
- **MFT** **PG # 301**
- Hidden blocks
- Boot sector

315. Block # 2 is the safest block to store the backup of boot block.

- True
- **False**

316. The keyboards interface as discussed earlier uses the IRQ0 and the port 64H as data port.

- True
- **False**

317. FAT12 will have 12-bit wide entries and can have 2^12=4096 entries maximum

- **True**
- False

318. In order to produce the sound from PC internal Speaker we have to load the___bit divisor value at the ___port.

- 8, 0x21
- **16, 0x42**
- 32, 0x22
- 64, 0x32

319. DMA stands for_____

- **Direct Memory Access** **PG # 4**
- Distinct Memory Access
- Direct Module Access
- Direct Memory Allocation

320. UART stands for_____

- **Universal Asynchronous Receiver Transmitter** **PG # 107**
- Universal Adjustment and Realigning Tool
- Unconventional Assisted Recovery Team
- None of these

321. Interrupt Vector Table (IVT) in short is a _____ bytes sized table.

- **1024** **PG # 10**
- 2048
- 3072
- 4096

322. Hardware Interrupts are _____.

- Preemptive
- **Non-Preemptive** **PG # 48**
- Both Preemptive and Non-Preemptive
- None of Given

323. Timer interrupt is a _____.

- **Hardware Interrupt** **PG # 28**
- Software Interrupt
- Both of these
- None of These

324. The keyboard makes use of interrupt number _____ for its input operations.

- **9** **PG # 34**
- 10
- 11
- 12

325. Register can be used to divide frequency is _____

- **Counter Register** **PG # 69**
- Accumulator Register
- None of these

326. Which port is known as Data Port_____

- **60H** **PG # 177**
- 61H
- 64H
- 69H

327. LPTs can be swapped.

- **True** **PG # 92**
- False

328. PPI is used to perform parallel communication

- **True** **PG # 81**
- False

-

329. _____is used to control the printer via the BIOS

- Int 16H
- **Int 17H PG # 84**
- Int 18H
- Int 19H

330. There are two main types of interrupts namely _____.

- PC based and Window based
- Hardware based and Kernal based
- **Hardware interrupts and Software interrupts PG # 10**
- None of the given

331. To set the interrupt vector means is to change the double word sized interrupt vector within the IVT.

- **True PG # 22**
- False

332. The service number is usually placed in the _____ register.

- AL
- CL
- **AH PG # 26**
- AX

333. The keyboard makes use of interrupt number _____ for its input operations.

- **9 PG # 34**
- 10
- 11
- 12

ایماندار کو غصہ دیر سے آتا ہے اور جلدی دور ہو جاتا ہے

334. The service _____ is called the keyboard hook service.

- 15H/2FH
- **15H/4FH** **PG # 44**
- 15H/FFH

335. The BIOS interrupt _____ can be used to configure RTC.

- **1AH** **PG # 136**
- 2AH
- 3AH
- 4AH

336. PPI stands for

- Parallel Programmable interface
- **Peripheral Programmable interface** **PG # 76**
- Port Programmable interface
- None of the given

337. Int _____ is used to control the printer via the BIOS.

- **17H** **PG # 84**
- 18H
- 20H
- 21H

338. Counter register can be used to divide clock signal.

- **True** **PG # 69**
- False

دنیا میں سب سے مشکل کام اپنی اصلاح اور سب سے آسان کام دوسروں پر نکتہ چینی کرنا ہے

339. The bit number _____ of the coprocessor control word is the interrupt enable flag.

- **7 PG # 168**
- 8
- 9
- 6

340. There are _____ kinds of serial communication.

- **2 PG # 105**
- 3
- 4
- 5

341. _____ store the base address for LPT1.

- 40:00H
- 40:02H
- **40:08H PG # 92**
- 40:1AH

342. The amount of memory above conventional memory (extended memory can be determined using the service _____.

- **15H/88H PG # 162**
- 16H/88H
- 17H/88H
- 21H/88H

343. The output on the monitor is controlled by a controller called _____within the PC.

- **Video controller PG # 30**
- Bus controller
- Ram controller
- None of the given

344. Interrupt 9 usually reads the _____ from keyboard.

- ASCII code
- **Scan code** **PG # 34**
- Both ASCII and Scan code
- None of the above

345. NMI Stand for

- **Non Maskable Interrupt** **PG # 46**
- Non Multitude Interrupt
- Non Maskable Instruction
- None of Given

346. A single interrupt controller can arbitrate among ____ different devices.

- 4
- 6
- **8** **PG # 47**
- 10

347. The microprocessor package has many signals for data. Below are some in Correct priority order (Higher to Lower).

- **Reset,Hold,NMI,INTR** **PG # 46**
- NMI, INTR,Hold,Reset
- INTR,NMI,Reset,Hold
- None of the Given

348. The _____function initialize the COM port whose number is passed as parameter using BIOS services.

- Initializecom()
- **Initialize()** **PG # 125**
- Recievechar()
- None of these option

349. There are two type of communication synchronous and Anti Synchronous

- True
- **False** **PG # 105**

350. REGS is a Union

- **True**
- False

351. Keyboard Status Byte is located at the address

- 0040:0000H
- 0040:0013H
- 0040:0015H
- **0040:0017H** **PG # 29**

352. If we use keep (0, 1000) in a TSR program, the memory allocated to it is

- 64000 bytes
- 32000 bytes
- **16000 bytes**
- 80000 bytes

353. Maximum number of interrupts in a standard PC is

- 64
- 128
- **256** **PG #1 0**
- 512

354. The -------- function receive a byte and COM port number is passed as parameter using BIOS service

- Receivebyte ();
- Receive ();
- **Receivechar ();** **PG # 125**
- None of the given option

355. _____ whenever receive indicates the start of communication …….. whenever receive indicates the end of communication

- **XON\XOFF** **PG #135**
- XOFF\XON
- XON\YOFF
- YON\XOFF

356. _____ is used to set time from RTC

- 1A\02H
- **1A\03H** **PG #138**
- 1A\04H
- 1A\05H

357. Set the Interrupt vector means to change the double word sized interrupt vector within IVT.

- **True** **PG #22**
- False

358. If keyboard buffer is empty the head and tail points at the same location.

- **True** **PG #55**
- False

359. Standard PC can have _____ PPI.

- 1
- **4** **PG #84**
- 8
- 16

360. By cascading two DMAs ____ bits can be transferred.

- 4
- 8
- **16** **PG #186**
- 32

361. PPI interconnection _____ bits is cleared to indicate low nibble is being sent.

- D1
- D2
- D3
- **D4** **PG # 101**

362. Display device (Monitor) performs _____ **I**/O.

- **memory mapped** **PG #30**
- Isolated
- Both of above
- None of these

363. Timer interrupt occurs _____ times every second by means of hardware.

- **18.2** **PG # 28**
- 16.2
- 15.2
- 14.2

364. An I/O device cannot be directly connected to the busses so controller is placed between CPU and I/O.

- **True** **PG # 83**
- False

365. Tail of keyboard should get to get the start of buffer.

- **True** **PG # 55**
- False

366. ____ No. of bytes are used to store the character in the keyboard buffer.

- 1
- **2** **PG # 54**
- 4
- 8

367. We have set the bit No. 7 of IMR(Interrupt Mask Register) to unmask the Interrupt so that interrupt _____ can occur at ____ line.

- **0xf ,IRQ 7**
- 0xa, IRQ 6
- 0x8, IRQ 5
- 0x6, IRQ 2

368. If we want to produce the grave voice from speaker phone then we have to load the ____ divisor values at Port ____.

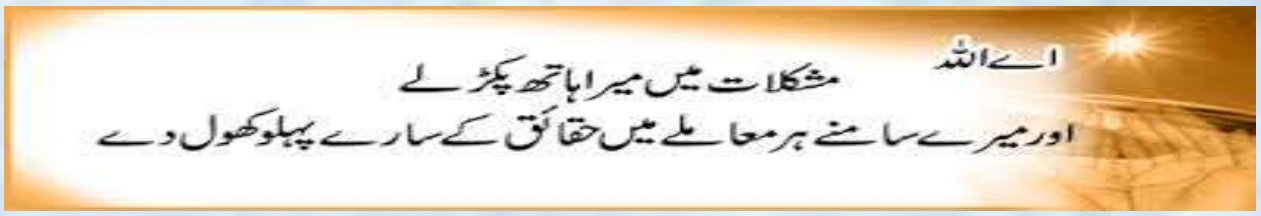
- high, 0x42
- low, 0x22
- high, 0x22
- **low, 0x42**

369. On the execution of IRET instruction, number of bytes popped from stack is

- 4 bytes
- 6 bytes
- **8 bytes** **Click here for detail**
- 10 bytes

370. If CPUID instruction is not present then the processor can be a

- **486 processor** **PG # 166**
- 386 processor
- 286 processor
- All of the above



371. Extended memory is available if the processor is of the type _____

- **AT** **PG # 171**
- XT
- All of the given choices
- None of them

372. The built in mechanism within the UART for error detection is_____

- hamming code
- **parity** **PG # 107**
- CRC16 (cyclic redundancy check 16 bit)
- CRC32 (cyclic redundancy check 32 bit)

373. If three Programmable interrupt controllers are cascaded then how many interrupt driven hardware IO devices can be attached _____

- 12
- 18
- 23
- **24** **PG #48**

374. Int 14H_____ can be used to send a byte

- Service#0
- **Service#1** **PG #121**
- Service#2
- None of the given option.

375. Int 14H _____ can be used to set the line parameter of the UART or COM port.

- **Service # 0** **PG # 119**
- Service # 1
- Service # 2
- None of the given options

376. Int 14H _____ can be used to receive a byte.

- Service # 0
- Service # 1
- **Service # 2 PG # 121**
- None of the given options

377. The _____ function simply enables the self test facility within the modem control register

- STOn()
- SelfTest()
- **SelfTestOn() PG # 127**
- None of these

378. _____ is a device incorporated into the PC to update time even if the computer is off.

- Clock counter
- ROM
- Clock
- **Real time clock PG # 136**

379. Interrupt _____ is used to get or set the time.

- 0AH
- **1AH PG #136**
- 2AH
- 3AH

380. _____ is used to set time.

- 1A/02H
- **1A/03H PG # 138**
- 1A/04H
- 1A/05H

381. ---------- is used to read date from RTC

- 1A\02H
- 1A\03H
- **1A\04H** **PG # 138**
- 1A\05H

382. ____whenever received indicates the start of communication and _____ whenever received indicates a temporary pause in the communication.

- **XON & XOFF** **PG # 135**
- XOFF & XON
- XON & YOFF
- YON & XOFF

383. The_____ function uses the COM port number to receive a byte from the COM port using BIOS services.

- recievebyte()
- initialize ()
- receive()
- **recievechar()** **PG # 125**

384. In self test mode the output of the UART is routed to its input.

- **True** **PG # 117**
- False

385. Only _____ ports are important from programming point of view.

- **70 and 71H** **PG # 141**
- 71 and 72H
- 70 and 72H
- 72 and 73H

386. The BIOS interrupt 0x1AH can be used to configure real time clock

- **True** **PG # 136**
- False

Note: If you found any mistake in mcqz please mail at above mentioned email address. And tell me your answer with references.







THE END

Campus (AKMP01)

ARSLAN ARSHAD
Arslan.arshad01@gmail.com

Reading Contents

Topic 1: Windows Operating System

Software can be categorized into two main types: **System Software** and **Application Software**. System Software is concerned to the system or its resources while Application Software is related to some application. Operating System is one of the good examples of System software that acts as a manager for the System's resources. It arbitrates and schedules the resources among the processes to avoid any kind of conflicts.
There are varieties of Operating Systems available in the market but Windows is one of the important Operating Systems developed by Microsoft. It is widely used inside the PCs, Laptops, enterprise servers, handheld devices and cell phones etc.

Core Features of Windows Operating System

1. **Memory Management**. Memory Management is the vital or key feature of Windows Operating System. The OS is responsible for managing primary as well as secondary memory. Virtual address space on secondary memory is also managed by OS to transfer data between primary and secondary storage and accommodate a large process in small memory space. All kinds of services and related data structures are supported by Operating Systems to efficiently manage the Virtual memory space.
2. **File systems**. Windows OS manages all kinds of files and folders on disk in hierarchical form. Large file naming space of 255 characters is supported by the OS. Number of APIs are provided by the OS to access the files in both Sequential and Random mode.
3. **Processors**. Windows Operating System provides support to multiprocessors and multi cores systems. It also arbitrates among the cores or processors to efficiently divide and allocate computational tasks to them.
4. **Resource naming and location**. In Windows OS, the resources like processes or devices are treated as objects and each object is assigned a unique name to identify, locate and access it.
5. **Multitasking**. Windows OS supports multitasking. It manages processes, threads, and other independent units, and their asynchronous execution. Tasks can be preempted and scheduled according to dynamically calculated priorities.
6. **Communication and Synchronization**. The Windows OS provides constructs to manage inter-process communication and synchronization within a computer or networked computers.
7. **Security and Protection**. The Windows OS has a strong security mechanism to protect resources from illegal and accidental access. A user cannot access other user data without assigning privileges.

Topic 2: Windows Evolution

Windows exist in several versions. New versions of Windows are introduced from time to time. Actually certain new APIs are included in the new version to improve or extend its functionalities. The following major themes or features are considered while developing a new version.

- **Scalability**. A new version of Windows OS runs on different platforms including PCs, enterprise servers, multiprocessing systems, mobiles and systems having large memory space.
- **Performance**. A newer version of Windows certainly improves performance compared to previous versions.
- **Integration**. A newer version must integrate with new technologies like web services, .NET technologies, multimedia etc.
- **Ease of use**. Certain new APIs and improved GI in the new version can ensure ease of use.
- **Enhanced API**. Introducing new APIs or enhancements in existing APIs should be the main theme of the new version.

Disk Operating System (DOS)

In the 1980s, Microsoft Disk Operating System was used inside the IBM PCs incorporating Intel Processor. It was a text based and command line operating system. It was a single user OS. Its filing system was based on FAT and was able to access up to the maximum of 4GB files.

Windows History

Keeping the demand of graphical user interface, Microsoft developed its first version Windows 3.1. In this version, DOS Kernel and FAT based file systems were used.

After that in the 1990s, certain new versions of Windows named Windows 95, 97 and 98 were introduced supporting the 32-bit architecture of Intel's processors.

Later on **Windows NT** versions were introduced supporting a file system based on new technology called NTFS. Its security and file system was better than the previous versions.

Windows Server 2008 OS was developed for professional use to manage enterprise and server applications. Support for multi-core technology and 64-bit applications was provided in this OS. Other Windows versions supporting 32-bit, 64-bit architecture, multi-core and multiprocessing were also introduced including Windows XP, Windows Vista, Windows 7, 8 and Windows 10.

Topic 3: Windows Market Role

Several competitors of Windows OS like UNIX, Linux etc. exist in the market; however, Windows has its own unique status in the market. It has several significant advantages over other operating systems.

Above 90% PCs are based on Intel's processors and Windows is the most appropriate OS for Intel PCs. In the world of desktop, the most dominant OS is the Microsoft Windows which enjoys a market share of above 80%. Windows is not confined to the desktop, it also has support for diverse platforms including multi-core, multiprocessing, servers and mobiles etc.

Due to its dominance role, certain applications and software development tools are available in the market that can easily integrate with Windows OS and can develop windows applications ranging from small scale to enterprise level.

One of the key features of Windows OS is its rich GUI that makes its use very convenient. This interface can be easily customized according to the local setup. The size, color and visibility of graphical interface objects can also be changed by the user.

Compared to other operating systems, certain modern features exist in Windows due to which most of the developers develop their applications for Windows targeting the huge market of Windows.

Topic 4: Windows, Standards and Open Systems

Open Source Software is a software that is publically available with its source code to use, modify and distribute with original rights. It is developed by the community rather than a single company or vendor. In contrast, proprietary software is copyrighted and only available to use under a license.

Windows Operating system is a proprietary and copyrighted software of Microsoft corporation. It is provided for use only under a License agreement. Without purchasing a license, its use is illegal and is an act of copyright infringement.

Being a closed system, Windows has the following strengths.

· As Windows components are provided and updated only by a single vendor, its implementation remains uniform throughout the world. Further, extensions in Window components or APIs are only vendor-specific and so no non-standard extension is possible except for platform differences.

Windows also support various types of hardware platforms like open systems.

Interoperability of Windows: Windows provide interoperability with non-window components.

- Windows OS provides support to the Standard C and C++ libraries. We can install and use any C compilers on Windows systems.
- Socket is a resource that is required for interface when two computers are interconnected to each other. Windows also supports sockets to communicate among devices having different computer architectures and access to TCP/IP and other networking protocols.
- It also supports the Remote Procedure Calls (RPCs) architecture to call the remote functions in distributed client-server based applications.
- Windows also supports the X Windows system which is open source, cross platform software providing GUI in a distributive network environment.

Topic 5: Windows Principles

- In Windows OS, all the system resources including processes, threads, memory, pipes, DLL etc. are represented by objects which are identified and referenced by a handle. These objects cannot be directly accessed. In case, if any application approaches to access these objects directly, Windows throws an appropriate exception. The only way to access and operate on these objects is a set of APIs provided by Windows. Several APIs can be related to a single object to manipulate it differently.
- A long list of parameters is associated with each API where each parameter has its own significance but only few parameters are specified for a specific operation.
- To perform the task of multitasking and multi-threading efficiently, Windows provides a number of synchronization constructs to arbitrate among the resources.
- The names of Windows APIs are long and descriptive for its proper and convenient use.
- Some pre-defined data types required for Windows APIs are:
    - BOOL (for storing a single logical value)
    - HANDLE (a handle for object)
    - LPTSTR (a string pointer)
    - DWORD (32-bit unsigned integer)
- Windows Data types avoid the pointer operator (*).
- Some lowercase prefix letters with variable names are used to identify the type of variable. This notation is called Hungarian notation. For example, in the variable name **lpszFilename**, 'lpsz' is Hungarian notation representing a long pointer to zero terminated string.
- **windows.h** is a header file including all the APIs prototypes and data types

Topic 6: 32-Bit and 64-Bit Source Code Portability

Windows keeps two versions of each API, one for 32-bit and other for 64-bit. A 32-bit code can be run on 64-bit hardware but will be unable to exploit some features of 64-bit like accessing large disk space or using large pointer or 64-bit operation.

Latest versions of Windows support both 32 and 64-bit architectures by keeping two versions of each API, one for 32-bit and other for 64-bit.

Interoperability of 32 and 64-bit: A single source code can be built for 32-bit as well as 64-bit versions. To decide whether executable code of 32 or 64-bit is generated by the compiler at runtime, it depends on its settings or configuration. Further, to decide which version of API is used, it is also based on the compiler's configuration.

A 32-bit code can run on 64-bit hardware successfully but will be unable to use some features of 64-bit like large disk space, large pointer etc.

A source code developed for 64-bit architecture cannot easily run on a 32-bit machine. For this purpose, re-compilation of the program is required and suitable configuration is made in the compiler to generate a 32-bit executable code.

Topic 7: When to use Standard C Library for File Operations

Windows provides a set of built-in APIs to perform I/O operations. A related API with specific parameters is invoked for the concerned resource and I/O operation is performed.

Similarly, certain C/C++ standard functions are available to perform I/O operations. For example, fopen(), fclose(), fread(), fwrite() etc. are C functions that can be used to perform I/O operations related to files.

Differences between Windows APIs and Standard Functions

Standard C functions can be used inside the source code to run on Windows platform because Windows has system calls at low level to support C/C++ functions for I/O operations.

If it is required to run a program on cross-platform, then it is preferred to use the standard C/C++ library functions inside the source code. However, in this case, the advanced Windows

features like locking, synchronization, asynchronous I/O, inter process communication etc. cannot be achieved.

In case, if portability is not focused and required to avail the advanced Windows features, then it is preferred to use the Windows APIs.

Topic 8: A Simple File Copy Program Using Standard C Library

```
// cpC.  Basic File Copy Program:  C Library Implementation
// Copy File1 to File2
#include<stdio.h>
#include<errno.h>
#define BUF_SIZE 256

int main(int argc, char *argv[])     {

        FILE *inFile, *outFile;
        char rec[BUF_SIZE];
        size_t  bytesIn, bytesOut;

        if(argc!=3)    {
          printf("Usage: cp file1 file2\n");
          return 1;     }

        inFile=fopen(argv[1], "rb");
        if(inFile==NULL)  {
          perror(argv[1]);
          return 2;
                                }

        outFile=fopen(argv[2], "wb");
        if(outFile==NULL)   {
          perror(argv[2]);
          return 3;                }

        /* Process the input file a record at a time */
        while((bytesIn = fread(rec, 1, BUF_SIZE, inFile)) > 0)
        {
          bytesOut=fwrite(rec, 1, bytesIn, outFile);
```

```
        if(bytesOut != bytesIn)   {
                perror("Fatal write error");
                return 4;               }
        }
        fclose(inFile);
        fclose(outFile);
        return 0;
}
```

This program is used to copy one file to another using C standard functions. In this program, a buffer of size 256 bytes is used in which the chunks of file are copied one by one.
The source file is opened in read binary mode and the destination file in write binary mode using the C **fopen()** function.
If both are successfully opened, then a file is read inside a loop chunk by chunk using **fread()** function and written onto the destination file using **fwrite()** function. After a few iterations, the file will be written to the destination file and both files are closed.

Topic 9: A Simple File Copy Program Using Windows APIs

Program 1 - 2

```
// cpC.  Basic File Copy Program:  Windows Implementation
// Copy File1 to File2
#include<stdio.h>
#include<windows.h>
#include<stringapiset.h>
#define BUF_SIZE 16384
int main(int argc, char *argv[])     {
HANDLE  hIn, hOut;
DWORD  nIn, nOut;
        CHAR buffer[BUF_SIZE];
        LPWSTR lpwszFile1, lpwszFile2;
        INT iLen1, iLen2;

if(argc !=3)   {
          fprintf(stderr, "Usage: cp file1 file2\n");
          return 1;   }
        lpwszFile1 = (LPTSTR)malloc(510);
```

```
        lpwszFile2 = (LPTSTR)malloc(510);
        iLen1 = MultiByteToWideChar(CP_ACP, 0, argv[1], -1, lpwszFile1, 510);
        iLen2 = MultiByteToWideChar(CP_ACP, 0, argv[2], -1, lpwszFile2, 510);
        hIn=CreateFile(lpwszFile1, GENERIC_READ, FILE_SHARE_READ, NULL,
                OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL, NULL);
        if (hIn==INVALID_HANDLE_VALUE)  {
            fprintf(stderr, "Cannot open input file. Error: %x\n", GetLastError());
            return 2;       }

        hOut=CreateFile(lpwszFile2, GENERIC_WRITE, 0, NULL, CREATE_ALWAYS,
                FILE_ATTRIBUTE_NORMAL, NULL);
        if (hOut== INVALID_HANDLE_VALUE)   {
                fprintf(stderr, "cannot open output file. Error: %x\n", GetLastError());
                CloseHandle(hIn);
            return 3;
        }

        while((ReadFile(hIn, buffer, BUF_SIZE, &nIn, NULL) && nIn > 0)  {
          WriteFile(hOut, buffer, nIn, &nOut, NULL);
          if (nIn != nOut)   {
                fprintf("Fatal write error: %x\n", GetLastError());
                CloseHandle(hIn);
                CloseHandle(hOut);
                return 4;
                }
        }
        CloseHandle(hIn);
        CloseHandle(hOut);
        return 0;
}
```

In this program, Windows APIs are used instead of C standard functions. In the main program, the words in capital letters like HANDLE, DWORD, CHAR, LPWSTR and INT are Windows data types. As the file paths given in command line parameters are in ASCII format, the parameters will be first converted to Unicode using the **MultiByteToWideChar()** function.

After that, a file is opened for reading purposes and its handle is stored in hIn. Similarly, another file is created for writing purposes and its handle is stored in hOut.

If files are successfully opened, then the source file is read in a loop and written to the destination file. At the end both the handles for files are closed.

Topic 10: A Simple File Copy Program Using Windows Convenience Function

Numerous Windows functions are used to perform various tasks at low level. However, Windows has a set of **Convenience functions** that combine several functions to perform a common task. In most cases, these functions improve the performance because several tasks are performed by a single function.

For example, **CopyFile()** is a convenience function that replaces the algorithms used for creating, opening, reading and writing one file to another.

Program 1 - 3

```
/* cpC.  Basic File Copy Program:  Windows Implementation
                        using convenience function CopyFile() */
// Copy File1 to File2

#include<stdio.h>
#include<windows.h>
#define BUF_SIZE 256
LPWSTR lpwszFile1, lpwszFile2;
INT iLen1, iLen2;

int main(int argc, char *argv[])        {
if(argc !=3)  {
          fprintf(stderr, "Usage: cp file1 file2\n");
          return 1;
               }
        lpwszFile1 = (LPTSTR)malloc(510);
        lpwszFile2 = (LPTSTR)malloc(510);
        iLen1 = MultiByteToWideChar(CP_ACP, 0, argv[1], -1, lpwszFile1, 510);
        iLen2 = MultiByteToWideChar(CP_ACP, 0, argv[2], -1, lpwszFile2, 510);

if (!CopyFile(lpwszFile1, lpwszFile2, FALSE)
        {
                        fprintf(stderr, "CopyFile Error: %x\n", GetLastError());
                        return 2;
        }
```

```
    return 0;
}
```

Lecture 11: Windows File System

Windows supports various file systems.

- **NT File System**

NTFS is an important file system supported by Windows, its main features are:

  - **Security**: One user cannot access other user data without privileges.
  - **Fault tolerance** (if a portion of disk corrupts, it works because different copies of information are maintained in this files system).
  - **Encryption** (data encrypted/decrypted, provide security)
  - **Compression** (Space capacity increased due to data compression)
  - Supports very huge file size

- **File Allocation Tables**
  - FAT12, FAT16, FAT32 (Start version of File Systems, also supported by Windows)
- Compact Disk File System (CDFS)
  - Support CD
- Supports Universal Disk Format (UDF) & Live File System (LFS) also

Lecture 12: File Naming Conventions

- Windows OS supports a number of File Systems. Each file system has its own mechanism for naming files.

Certain Limitations for File Naming

- Letters like A, B, C etc. are used to represent Drive, Network Drives are represented by higher letters like N, K, L etc.
- Double slash (\\) in the start of path represents remote resource
- Forward slash (\ ) or backslash (/) is used as a path name separator
- The first 31 ASCII characters (control characters) cannot be used in file names
- Special symbols like \, /, colon (:), pipe (|) etc. cannot be used in filenames
- File and directory names are case insensitive
- File name and extensions are separated by (.)
- Max size for file name is 255 and for path is 260 characters.

- File extension takes 2 – 4 bytes
- Single dot (**.**) represents current directory while double dot (**..**) represents one step back (up) directory

Lecture 13: Creating and Opening Files (Using Windows API)

The CreateFile() API

CreateFile() API with a list of parameters is used to open or create a new file. Its return type is HANDLE to an open file object in case of successful opening or creation. The parameters are:

lpFileName: It is a string pointer that points to a filename to be opened or created.

dwDesiredAccess: It is a 32-bit double word which specifies the GENERIC_READ and WRITE access.

dwShareMode: This mode specifies how the file is shared?

- 0 signifies that file will not be shared
- FILE_SHARE_READ allows the file to be shared for concurrent read
- FILE_SHARE_WRITE allows the file to be shared for writing.

lpSecurityAttributes: points to a security attributes structure.

dwCreate: signifies whether to create a new file or overwrite an existing one.

- CREATE_NEW: Creates a new file. If the file already exists, then fail.
- CREATE_ALWAYS: Creates a new file or overwrites an existing one.
- OPEN_EXISTING: open an existing file or fail if the file does not exist
- OPEN_ALWAYS: open an existing file, if it does not exist, then create it.

dwFlagsAndAttributes: It signifies attributes of the newly created file.
hTemplateFile: It is the Handle of an open file that specifies attributes to apply to a newly created file.

Lecture 14: Reading/Writing a File

The **ReadFile()** API is used to read data from file to buffer.

Syntax:

BOOL ReadFile(HANDLE hFile, LPVOID lpBuffer, DWORD nNumberOfBytesToRead, LPDWORD lpNumberOfBytesRead, LPOVERLAPPED lpOverlapped);

- If the file is not opened in concurrent mode, then ReadFile() starts reading from the current position.
- If the current location is End of File, then no Errors occur and *lpNumberOfBytesRead is set to zero
- The function returns FALSE if it fails in case any of the parameter is invalid

Parameters

- HANDLE hFile is the file handle
- LPVOID lpBuffer is the address of the array that stores the data read from the file.
- DWORD nNumberOfBytesToRead is the number of bytes to be read from the file
- LPDWORD lpNumberOfBytesRead is the number of bytes actually read
- LPOVERLAPPED lpOverlapped is used for concurrent processing.

The **WriteFile()** API is used to write data from buffer to file.

Syntax:

BOOL WriteFile(HANDLE hFile, LPCVOID lpBuffer, DWORD nNumberOfBytesToWrite, LPDWORD lpNumberOfBytesWritten, LPOVERLAPPED lpOverlapped);

To write through the current size of file, the file must be opened with FILE_FLAG_WRITE_THROUGH option

Lecture 15: Closing a File

After opening & using a file, it is required to close and invalidate the file handles in order to release the system resources.

The **CloseFile()** API is used to close a file. A file handle is passed as parameter to this API as a result of which the API will return True or False value. If the operation is successful, then it returns True value. In case if the handle is already invalid, then it will return False value.

Lecture 16: Generic Characters

- Windows supports both ASCII (8-bit standard) and Unicode (16-bit standard) characters
- Windows have different API function variants to deal with both Unicode and ASCII characters.
- Unicode is used to represent characters in other languages like Arabic, Chinese, Urdu etc.
- The structure of a program is different for Unicode and ASCII characters.
- Generic program or code works for both the ASCII and Unicode.
- Windows provides a set of data types for ASCII, a set of data types for Unicode and a set of data types for Generic code like TCHAR, LPTSTR, LPCTST etc.
- For Unicode the following two macros should be used before including <windows.h>
    - #define UNICODE: it includes all the function headers
    - #define _UNICODE: it includes variable types
    - These macros will force the generic data type to be replaced by Unicode data types.
- sizeof() operator should be used in coding instead of hard coding if required.
- Use generic C Library functions like _itot(), _stprintf(), _tcscopy() etc.
- Use the _TEXT or T_macro for representing string constants.
- <tchar.h> file should be included for generic C functions.
- Most of the APIs in the latest version of Windows expect to pass the parameters in Unicode, therefore a generic code should be written instead of ASCII code.

Lecture 17: Generic Functions

- Besides Generic data types, Windows also supports generic functions.
- Examples of generic functions are:
  _tcscmp() instead of lstrcmp(), _tcscmpi() instead of lstrcmpi()

- Some functions that deal with Unicode characters and strings and work with locale settings transparently are:
    - CharUpper()
    - IsCharAlphaNumeric()
    - CompareString()
- Generic main() function is modified in terms of data types of its parameters and its name as mentioned below:
  int _tmain(int argc, LPTSTR argv[])
- <windows.h> and <tchar.h> header files must be included before the main() function.
- Windows usually have two versions for each API function. For example, TextOut will have two variants i.e. TextOutA(16 bit API) and TextOutW(32 bit API)
- The use of Generic data types and Unicode macro enables the compiler to decide which version to choose

Lecture 18: Unicode Strategies

While writing a new code or enhancing an existing one, a programmer can adapt any of the following strategies based on requirements.

- Ignore Unicode Policy
- Use 8 bit only
- Ignore Unicode
- Use data types like char or CHAR
- Most of the Windows cannot be used ignoring unicode
- Use standard library functions like printf(), scanf(), atoi() etc.

Using Generic Code Policy

- The Unicode macro is used to switch between 8 bit and Unicode
- Generic functions are used
- Generic data types are used

Using Unicode Policy

- Use Unicode only
- Unicode functions are used
- Wide characters data types are used

Unicode and 8-bit Policy

- Handle Unicode and 8-bit

- Write code for both Unicode and 8-bit
- Decision regarding which option to choose is made at runtime
- Use of Unicode only strategy is now increasing popularly
- Use of generic code makes the code more flexible

Lecture 19: Reporting Errors

- Windows Operating System provides the facility to report errors, if occurs.
- It is an important and salient feature of Windows to display error messages for user convenience.
- The important Windows function for reporting errors is **GetLastError()**. It returns the code for the last system error.
- Another function **FormatMessage()** translates the error code into meaningful English or language selected in preferences.
- A message string reference against the error code is stored in another parameter.
- Another function **LocalFree()** is also used with FormatMessage() to deallocate the allocated local memory.

Lecture 20: Example for Reporting Errors

The following two header files must be included for different functions, prototypes, and data types.

<environment.h> and **<everything.h>**

<environment.h> includes a Unicode macro for the environment of the program.

<everything.h> includes all the header files required for typical Windows program

Example:

```
#include<everything.h>
VOID ReportError(LPCTSTR userMessage, DWORD exitCode, BOOL printErrorMessage)
{
        DWORD eMsgLen, errNum,=GetLastError();
        LPTSTR lpvSysMsg;
   _ftprintf(stderr, _T("%s\n"), userMessage)
    if (printErrorMessage)
```

```
    {
        eMsgLen = FormatMessage(FORMAT_MESSAGE_ALLOCATE_BUFFER  |
        FORMAT_MESSAGE_FORM_SYSTEM, NULL, errNum, MAKELANGID(LANG_NEUTRAL,
        SUBLANG_DEFAULT), (LPTSTR) &lpvSysMsg, 0, NULL);

        if (eMsgLen > 0)
        {
                _fprintf(stderr, _T("%s\n"), lpvSysMsg);
        }
        else
        {
                _ftprintf (stderr, _T("Last Error Number; %d\n"), errNum);
        }
       if (lpvSysMsg != NULL)
       LocalFree(lpvSysMsg);
    }
    if (exitCode>0)
    {
        ExitProcess(exitCode);
        return;
    }
}
```

Here this function **ReportError()** receives three parameters. Inside this function, the result of **GetLast()** function is stored in the variable, errNum. A generic string variable "lpvSysMsg" is declared for storing the error message.

If the user likes to print the error message, then the error code will first format and convert it into a string using the **FormatMessage()** function.
If message length is greater than 0, then it will print the message, otherwise will print the error code and memory is deallocated.

Topic 20:

Example: Reporting Errors

We will discuss an example of reporting errors. Windows have provide certain API's in which the functions of getlasterror() and formatmessage(), we will see how we can use these functions to report errors. For this we will make a generic function that we will use in reporting errors.

To write code we need following customized header files

- Enviornment.h
- Everything.h

Enviornment.h includes macros for the environment of the program which is basically used of UNICODE macro.

```
#if(WIN32_WINNT>=0x600)

#define WIN32_WINNT 0x600  /* Enable use of NT 5 (XP,2000,2003) functions */

#else

#if (WIN32_WINNT>=0x500 ) /*enable use of NT5(XP,2000,2003) functions */

#endif

#endif
```

In environment.h we have also

```
#ifdef UNICODE

#define_UNICODE

#endif

#ifndef UNICODE

#undef_UNICODE
```

```
#endif

#define LANG_DFLT LANG_ENGLISH  // set default language

#define SUBLANG_DFLT SUBLANG_ENGLISH_US
```

Everything.h includes all the header file that will be typically required for all the subsequent window programs.

```
#include “ENVIORNMENT.h

#include<windows.h>  // all headers

#include<tchar.h> // generic functions

#include<stdio.h>  // use for working in CLI

#include<stdlib.h>

#include<malloc.h>// memory allocation

#include<io.h>  // input output operations

#include<WinSock2.h> // for windows socket operations

#include “support.h”

#include _MT

#include <process.h> // for multi tasking and multi threading

#endif
```

Following is the code of reporting errors.

```
#include  “Everything.h”

VOID ReportError{LPCTSTR userMessage, DWORD exitCode, BOOL
printErrorMessage}

{

DWORD eMsgLen, errNum=GetLastError();
```

```
LPSTR lpvSysMsg;
_ftprint(stderr,_t("%s\n"},userMessage};
If (printErrorMessage)
{
 eMsgLen= FormatMessage{FORMAT_MESSAGE_ALLOCATE_BUFFER|
FORMATE_MESSAGE_FROM_SYSTEM,
NULL,errNUM,MAKELANGID{LANG_NEUTRAL, SUBLANG_DEFAULT),
(LPSTR)&lpsSysMSg,0,NULL);
If(eMsgLen>0)
{
    _ftprint(stder,T("%s\n"|,lpvSysMsg));
}
Else
{
_ftprint{stderr,_T{"LastErrorNumber,%d\n"},errNum};
}
If (lpvSysMsg!=NULL) LOCALFree(lpvSysMsg);
}
If(exitCode>0)
ExitProcess(exitCode);
return;
}
```

Topic 21:- Standard IO devices:

Reading Material:

There are three standard IO devices used in Operating system.

- Input
- Output
- Error

An input operation is performed by default on standard IO devices. For example we does not mention in printf command where to print, it print the letter or any string by default on standard Output device and in getch, character is get by default input device. All the IO devices are manipulate in windows through handles.

Certain APIs are used to acquire handle to standard IO devices.

HANDLE GetStdHandle(DWORD nStdHandle);

- It returns a valid handle if the function succeed
- In case of failure it return INVALID_HANDEL_VALUE
- Successive calls to the functions will still run the same handle
- If the handle is closed it makes it subsequently unusable for the process in future.

Three types of values can be pass to the nSTDHandle

1. STD_INPUT_HANDLE
2. STD_OUPUT_HANDLE
3. STD_ERROR_HANDLE

STD_INPUT_HANDLE contains CONIN$(Console input) as an environment variable, STD_OUTPUT_HANDLE contains CONOUT$(Console Output) as an environment variable.

Operating system have also the concept of redirection using the given API

BOOL SetStdHandle(DWORD nStdHandle, HANDLE hHandle);

It also return true if calls succeed and return false in case of fail.

Topic 22: Copying multiple files using windows API

Reading Content:

We will see in this module how to display files using cosonle APIs of windows, in other words display file on screen is actually copying file on console. For this we made utility function “options()”. We will also use this function further. This function take a variable list of parameters and use to parse these variable list. Basically we specify the list of parameters of a program on command prompt, there are number of options in it. Option () is use to parse these options. It identify the “-“ prefix and check all the possible options, and set the flag against the set options.

Following is the code of the program

```
#include “Everything.h”

#include <stdarg.h>

DWORD Options (int argc, LPCTSTR argv [], LPCTSTR OptStr, …)   /*… show the parameters list are variables */

{

  Va_list pFlagList;

LPBOOL pFlag;

Int iFlag = 0, iArg;

Va_start (pFlagList, OptStr);

While ( (pFlag= Va)arg (pFlagList, LPBOOL)) != NULL

        && iFlag<(int)_tcslen (OptStr)){

*pFlag= False;

 For (iArg= 1; !(*(pFlag) && iArg <argc &&argv[iARG] [0]== _T(‘-‘); iArg++)
```

```
        *pFlag = _memtchr (argv [iArg], OptStr [iFlag], _tcslen (argv [iArg]))!=
NULL

iFlag++;

}

Va_end (pFlagList);

For (iArg= 1; !(*(pFlag) && iArg <argc &&argv[iARG] [0]== _T('-'); iArg++);

Retrun iArg;

}
```

This utility or program takes the number of parameters through command prompt, in which we specify option and file names. The Files names which are given , it open these file and try to print the content of these file on console and if no file name is given then it take the data from standard input and send to the standard output. It also use report error function in case of any error occur.

CatFile Function:

```
Static VOID CatFile (HANDLE hInFile, HANDLE hOutFile)

{

   DWORD nIn, nOut;

  BYTE buffer [BUF_SIZE];

While (ReadFile(hInFile, buffer, BUF_SIZE, $nIn, NULL) && (nIn !=0 )

&& WriteFile (hOutFile, buffer, nIn, &Out, NULL));

Return;

}
```

Topic 23 : Encrypting files

Reading Content:

Encryption is a very old technique, and roman empire use to encrypt secret conversation in war days and they use Ceasar Cipher algorithm to encrypt. In this method an alphabet is substituted by another alphabet placed n positions forward in circular manner. The text that is changed using encryption method is called Cipher text.

The text that we are going to encrypt is called plain text so it is denoted by P and after encrypt we present it with C.

File Encryption Program

- if n=1 then A will be replaced by B, B will be replaced by C and so on uptil Z which will be again replaced by A.
- if n=2 then A will be replaced by C, B will be replaced by Dand so on uptil Y which will be again replaced by A
- it uses the function C= (P + n) mod 26

We take mod 26 because total character are 26.

We use mod 256 because of ASCII character

File Encryption Program

- The program uses a slightly different version of Ceasar Cipher adapted for ASCII characters
- C = (P + n) mod 256

This technique is not exactly cipher but little bit similar to cipher. Following is the code of encrypting file

```
#include "Everything.h"

#include <io.h>
```

```
BOOL cci_f (LPCTSTR, LPCTSTR, DWORD);
int _tmain (int argc, LPTSTR argv [])
{
   if (argc != 4)
        ReportError  (_T ("Usage: cci shift file1 file2"), 1, FALSE);

   if (!cci_f (argv [2], argv [3], _ttoi(argv[1])))
        ReportError (_T ("Encryption failed."), 4, TRUE);
   return 0;
}
BOOL cci_f (LPCTSTR fIn, LPCTSTR fOut, DWORD shift)
{
   HANDLE hIn, hOut;
   DWORD nIn, nOut, iCopy;
   BYTE buffer [BUF_SIZE], bShift = (BYTE)shift;
   BOOL writeOK = TRUE;
hIn = CreateFile (fIn, GENERIC_READ, 0, NULL, OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL, NULL);
   if (hIn == INVALID_HANDLE_VALUE) return FALSE;
hOut = CreateFile (fOut, GENERIC_WRITE, 0, NULL, CREATE_ALWAYS, FILE_ATTRIBUTE_NORMAL, NULL);
   if (hOut == INVALID_HANDLE_VALUE) {
        CloseHandle(hIn);
        return FALSE;
```

```
    }
    while (writeOK && ReadFile (hIn, buffer, BUF_SIZE, &nIn, NULL) && nIn > 0) {
        for (iCopy = 0; iCopy < nIn; iCopy++)
            buffer[iCopy] = buffer[iCopy] + bShift;
        writeOK = WriteFile (hOut, buffer, nIn, &nOut, NULL);
    }
    CloseHandle (hIn);
    CloseHandle (hOut);
    return writeOK;
}
```

Topic 24: File and Directory Management

Reading Content:

We will see in this module the types of API for file management. Windows provides lots of function for file and directory management. These functions are pretty straightforward and easy to use.

- Such functions perform operations like
    - Delete
    - Copy
    - Rename

Delete

- Delete function will help to delete the file on a given path. For deleting file the following API is used.
- BOOL DeleteFile(LPCTSTR lpFileName) ;
- Returns TRUE if the file at the given valid file path is deleted

Copy

- For copying file the following API is used .
- BOOL CopyFile(LPCTSTR lpExistingFileName, LPCTSTR lpNewFileName, BOOL bFailIfExists);
- Copies an existing file from lpExistingFileName path to lpNewFileName, If new filename is same as existing then the file is overwritten only if bFailIfExists is false. It returns TRUE if the file at the given file is copied successfully

Hard Copy

- Windows also provides hardlinks. Following API is used for creating hardlinks

BOOL CopyHardLink(LPCTSTR lpFileName, LPCTSTR lpExistingFileName, LPSECURITY_ATTRIBUTES lpSecurityAttributes);

- Creates a hard link for existing file
- Both the files must be on same system volume
- Security attributes will apply on new file name

Move

- Use the following APIs to move files
- BOOL MoveFile(LPCTSTR lpExistingFileName, LPCTSTR lpNewFileName);

- BOOL MoveFileEx(LPCWSTR lpExistingFileName, LPCWSTR lpNewFileName,
- DWORD dwFlags);
- MoveFile() fails if new file already exists. We use MoveFileEx() to overwrite existing file.
- MoveFileEx() is the extended version of MoveFile() It can be used for both files and directories

Topic 25: Directory Management

We will discuss the functions which we can use for directory management. We will do different directory operation like create directory , remove directory and move directory.

For **create directory** we will use the following function

```
BOOL CreateDirectory(LPCTSTR lpPathName,
  LPSECURITY_ATTRIBUTES lpSecurityAttributes
);
```

For **remove directory** we will use the following function

```
BOOL RemoveDirectory(
  LPCSTR lpPathName
);
```

- lpPathName specifies the path of the directory to be created or deleted.

Current Directory

In windows and many other operating system concept of current directory or current working directory is also exist. Each Process has a current working directory which can be changed/retrieved using the following API

```
BOOL SetCurrentDirectory(
  LPCTSTR lpPathName
);
DWORD GetCurrentDirectory(
  DWORD  nBufferLength,
  LPTSTR lpBuffer
);
```

Topic 27: Printing and Prompting on Console

In the previous module we see the certain APIs , which perform input/output operations on console, now we use these APIs. We create two types of utility functions, one is help us to display string on console and other is pass some message to user and also take input from users. Following are names and description of these functions.

Variable parameter list

- PrintStrings() function is designed which take variable list of parameters.
- Variable parameter list is processed using va_start(), va_arg()and va_end() library functions

Console Prompt functions:

- This function devised such that it prompt a given message to the user on console
- Further it also receives user response through the console

Print String Function:

```
#include "Everything.h"

#include <stdarg.h>


BOOL PrintStrings (HANDLE hOut, ...){

        DWORD msgLen, count;

        LPCTSTR pMsg;

        va_list pMsgList;    /* Current message string. */

        va_start (pMsgList, hOut);         /* Start processing msgs. */

        while ((pMsg = va_arg (pMsgList, LPCTSTR)) != NULL) {   // read variable
list

         msgLen = lstrlen (pMsg);   // message length

if (!WriteConsole (hOut, pMsg, msgLen, &count, NULL) // in case of write
console fail

                                && !WriteFile (hOut, pMsg, msgLen * sizeof (TCHAR),
        &count, NULL)) {

                        va_end (pMsgList);

                        return FALSE;

         }

        }

        va_end (pMsgList);
```

```
        return TRUE;
}
BOOL PrintMsg (HANDLE hOut, LPCTSTR pMsg){
        return PrintStrings (hOut, pMsg, NULL);
}
```

Console Prompt Function

```
BOOL ConsolePrompt (LPCTSTR pPromptMsg, LPTSTR pResponse, DWORD
maxChar, BOOL echo){
        HANDLE hIn, hOut;
        DWORD charIn, echoFlag;
        BOOL success;
// create console input file
  hIn = CreateFile (_T("CONIN$"), GENERIC_READ | GENERIC_WRITE, 0,
                        NULL, OPEN_ALWAYS, FILE_ATTRIBUTE_NORMAL, NULL);
// create console output file
        hOut = CreateFile (_T("CONOUT$"), GENERIC_WRITE, 0,
                        NULL, OPEN_ALWAYS, FILE_ATTRIBUTE_NORMAL, NULL);
        /* Should the input be echoed? */
        echoFlag = echo ? ENABLE_ECHO_INPUT : 0;
   success =  SetConsoleMode (hIn, ENABLE_LINE_INPUT | echoFlag |
ENABLE_PROCESSED_INPUT)
&& SetConsoleMode (hOut, ENABLE_WRAP_AT_EOL_OUTPUT |
ENABLE_PROCESSED_OUTPUT)
```

```
                    && PrintStrings (hOut, pPromptMsg, NULL)
                    && ReadConsole (hIn, pResponse, maxChar - 2, &charIn,
NULL);

    /* Replace the CR-LF by the null character. */
    if (success)
     pResponse [charIn - 2] = _T('\0');
    else
     ReportError (_T("ConsolePrompt failure."), 0, TRUE);
    CloseHandle (hIn);
    CloseHandle (hOut);
    return success;
}
```

Topic 28:

In this module we will see a small code which will use get current directory.

Printing Current Directory

This example uses the *GetCurrentDirectory()* function to get the current directory, further it uses the console IO to print the path.

```
#include "Everything.h"

#define DIRNAME_LEN (MAX_PATH + 2)

int _tmain (int argc, LPTSTR argv [])
```

```
{


    TCHAR pwdBuffer [DIRNAME_LEN];

    DWORD lenCurDir;

//calling getcurrent directory and name stored in pwdbuffer

 lenCurDir = GetCurrentDirectory (DIRNAME_LEN, pwdBuffer);

    if (lenCurDir == 0)  // in case of failure

            ReportError (_T ("Failure getting pathname."), 1, TRUE);

    if (lenCurDir > DIRNAME_LEN)

            ReportError (_T ("Pathname is too long."), 2, FALSE);

        PrintMsg (GetStdHandle (STD_OUTPUT_HANDLE), pwdBuffer);  // in case of
correction

    return 0;

}
```

Topic 29:

If we see historically, there were some file system which was of 12-bit after that we have 32-bit system and still somewhere 32-bit file system are used. FAT based system allowed a maximum file size of 2^{32} bytes which is 4GB. NTFS theoretically provides the file size limit of 2^{64} which is very huge.

Files of such proportion are called huge files. Although for most of the application 32 bit file space is sufficient. However due to rapid technological changes leading to increased disk spaces its useful to know how to deal with 64

bit huge file spaces and windows facilitate with some API's that support 64-bit file system.

Topic 30:

File Pointer APIs

Whenever a file is opened using CreateFile() the file pointer is placed at the start of file. The file pointer changes as ReadFile() or WriteFile() operations are performed. Every subsequent read/write operation is performed at the current file pointer position.

SetFilePointer() and **SetFilePointerEx()** functions are used to change file pointer position. For random file accessSetFilePointer() is clumsy to used for 64 bit operations. SetFilePointerEx() can be more readily used for 64 bit operations

Use of SetFilePointer() used is difficult as compared to SetFilePointerEx().

SetFilePointer()

```
DWORD SetFilePointer(
  HANDLE hFile,
  LONG   lDistanceToMove,  // for 32-bit file
  PLONG  lpDistanceToMoveHigh,  // pointer to a long for NTFS
  DWORD  dwMoveMethod
);
```

hFile

A handle to the file.

lDistanceToMove

The low order 32-bits of a signed value that specifies the number of bytes to move the file pointer.

lpDistanceToMoveHigh

A pointer to the high order 32-bits of the signed 64-bit distance to move.

dwMoveMethod

The starting point for the file pointer move.

FILE_BEGIN // file pointer move number of bytes w.r.t the start of file

FILE_CURRENT // file pointer move with respect to the current position

FILE_END // file pointer move w.r.t the end of file

Topic 31:

For large files that may have size 2^{64} , we need to understand the 64 bit arithmetic. To facilitate 64-bit integer arithmetic windows provide a union LARGE_INTEGER. This union has structure for dealing with **lower and higher double** words Moreover it also has a field to deal with whole quadword of type LONGLONG.

Definition of Structure of large integer

```
typedef union _LARGE_INTEGER {
  struct {
        DWORD LowPart;
        LONG  HighPart;
```

```
  } ;
  struct {
        DWORD LowPart;
        LONG  HighPart;
  } u;
  LONGLONG QuadPart;
} LARGE_INTEGER;
```

Extension of SetFilePointerEx

```
BOOL SetFilePointerEx(
  HANDLE      hFile,
  LARGE_INTEGER  liDistanceToMove,
  PLARGE_INTEGER lpNewFilePointer,
  DWORD       dwMoveMethod
);
```

hFile

A handle to the file. The file handle must have been created with the GENERIC_READ or GENERIC_WRITE access right

liDistanceToMove

The number of bytes to move the file pointer.

lpNewFilePointer

A pointer to a variable to receive the new file pointer.

dwMoveMethod

he starting point for the file pointer move.

FILE_BEGIN

FILE_CURRENT

FILE_END

Topic 32:

Overlap structure is a structure which is defined in the 'windows.h' and can be used with read file and write file API's. This structure can also be used to set file pointer position, Although the name is overlapped but it is not necessary that can used only for overlapped operations, we also used in multitasking.

SetFilePointer() or SetFilePointerEx() need not be invoked we can achieve this goal using readfile and write file. Also, you can append to the file by specifying 0xffffffff in both Offset and OffsetHigh fields

Definition of Overlap structure:

```
typedef struct _OVERLAPPED {
  ULONG_PTR Internal;  // reserved filed
  ULONG_PTR InternalHigh;    // reserved filed
  union {
        struct {
```

```
        DWORD Offset;
        DWORD OffsetHigh;
        } DUMMYSTRUCTNAME;
        PVOID Pointer;
  } DUMMYUNIONNAME;
  HANDLE     hEvent;
} OVERLAPPED, *LPOVERLAPPED;
```

Implementation:

```
filePos.QuadPart=x;   // large integer variable
ov.Offset=filePos.LowPart;   // place low part of file position
ov.OffsetHigh=filePos.HighPart;  // place high part of file position
ReadFile(hFile, buf, sizeof(buf), &nRead, &ov);
…..
…
WriteFile(hFile, buf, sizeof(buf), &nWrite, &ov);
```

Topic 33:

One method of getting file size is already exist and that is to open a file first using create file, once file is open the file pointer is pointing to the first byte then we move file pointer to the end of file (eof). So file pointer is move from

starting to end of file is give use the size of the file. Windows also provides an API to get file size GetFileSizeEx()

GetFileSizeEx()

```
BOOL GetFileSizeEx(
  HANDLE      hFile,
  PLARGE_INTEGER lpFileSize
);
```

hFile

A handle to the file. The handle must have been created with the FILE_READ_ATTRIBUTES access right

lpFileSize

A pointer to a LARGE_INTEGER structure that receives the file size, in bytes.

Windows also give option to set the size. The file size can also be changed, reducing the file size truncate data. Increasing the file size can be useful where the size of file is expected to grow. We use SetEndOfFileEx() to change the file size.

Topic 34:

In this topic we discuss the example of creates a file with a capacity of specified records. The file has a header followed by equal size records. The feature of this example is, user can modify any record randomly and get the total count of records in the file.

```
#include "Everything.h"

        #define STRING_SIZE 256

        typedef struct _RECORD { /* File record structure */

                 DWORD                          referenceCount;   /* 0 meands an
empty record */

                SYSTEMTIME                  recordCreationTime;

                SYSTEMTIME                  recordLastRefernceTime;

                SYSTEMTIME                  recordUpdateTime;

                TCHAR                         dataString[STRING_SIZE];

        } RECORD;

        typedef struct _HEADER { /* File header descriptor */

                DWORD                         numRecords;

                DWORD                         numNonEmptyRecords;

        } HEADER;

        int _tmain (int argc, LPTSTR argv[])

        {

                HANDLE hFile;

                LARGE_INTEGER currentPtr;

                DWORD OpenOption, nXfer, recNo;

                RECORD record;
```

```
        TCHAR string[STRING_SIZE], command, extra;
        OVERLAPPED ov = {0, 0, 0, 0, NULL}, ovZero = {0, 0, 0, 0, NULL};
        HEADER header = {0, 0};
        SYSTEMTIME currentTime;
        BOOLEAN headerChange, recordChange;

int prompt = (argc <= 3) ? 1 : 0;
 if (argc < 2)
        ReportError (_T("Usage: RecordAccess file [nrec [prompt]]"), 1, FALSE);
OpenOption = ((argc > 2 && _ttoi(argv[2]) <= 0) || argc <= 2) ? OPEN_EXISTING : CREATE_ALWAYS;
        hFile = CreateFile (argv[1], GENERIC_READ | GENERIC_WRITE,
                        0, NULL, OpenOption, FILE_FLAG_RANDOM_ACCESS, NULL);
        if (hFile == INVALID_HANDLE_VALUE)
           ReportError (_T("RecordAccess error: Cannot open existing file."), 2, TRUE);
        if (argc >= 3 && _ttoi(argv[2]) > 0)  {
                header.numRecords = _ttoi(argv[2]);
                        if (!WriteFile(hFile, &header, sizeof (header), &nXfer, &ovZero))
```

```
                                        ReportError (_T("RecordAccess Error: WriteFile
header."), 4, TRUE);

                                    currentPtr.QuadPart = (LONGLONG)sizeof(RECORD) *
_ttoi(argv[2]) + sizeof(HEADER);

                                if (!SetFilePointerEx (hFile, currentPtr, NULL, FILE_BEGIN))

                                                ReportError (_T("RecordAccess Error: Set
Pointer."), 4, TRUE);

                                if (!SetEndOfFile(hFile))

                                        ReportError (_T("RecordAccess Error: Set End of
File."), 5, TRUE);

                                    if (prompt) _tprintf (_T("Empty file with %d records
created.\n"), header.numRecords);

                                return 0;

                        }

if (!ReadFile(hFile, &header, sizeof (HEADER), &nXfer, &ovZero))

                                ReportError (_T("RecordAccess Error: ReadFile header."),
6, TRUE);

                            if (prompt) _tprintf (_T("File %s contains %d non-empty records
of size %d.\n Total capacity: %d\n"),

                                argv[1], header.numNonEmptyRecords, sizeof(RECORD),
header.numRecords);

while (TRUE) {

                                headerChange = FALSE; recordChange = FALSE;
```

```
                                                    if (prompt) _tprintf (_T("Enter r(ead)/w(rite)/d(elete)/qu(it) record#\n"));
                        _tscanf (_T("%c%u%c"), &command, &recNo, &extra);
                        if (command == _T('q')) break;
                        if (recNo >= header.numRecords) {
                                    if (prompt) _tprintf (_T("record Number is too large. Try again.\n"));
                                continue;
                        }
                                      currentPtr.QuadPart = (LONGLONG)recNo * sizeof(RECORD) + sizeof(HEADER);
                        ov.Offset = currentPtr.LowPart;
                        ov.OffsetHigh = currentPtr.HighPart;
                           if (!ReadFile (hFile, &record, sizeof (RECORD), &nXfer, &ov))
                                ReportError (_T("RecordAccess: ReadFile failure."), 7, FALSE);
                         GetSystemTime (&currentTime); /* Use to update record time fields */
                        record.recordLastRefernceTime = currentTime;
if (command == _T('r') || command == _T('d')) { /* Report record contents, if any */
                                if (record.referenceCount == 0) {
```

```
                                        if (prompt) _tprintf (_T("record
Number %d is empty.\n"), recNo);
                                    continue;
                                } else {
                                        if (prompt) _tprintf (_T("record
Number %d. Reference Count: %d \n"),
                                            recNo, record.referenceCount);
                                    if (prompt) _tprintf (_T("Data: %s\n"),
record.dataString);
                                    /* Exercise: Display times. See ls.c for an
example */
                                }
                                if (command == _T('d')) { /* Delete the record */
                                    record.referenceCount = 0;
                                    header.numNonEmptyRecords--;
                                    headerChange = TRUE;
                                    recordChange = TRUE;
                                }
} else if (command == _T('w')) { /* Write the record, even if for the first time */
                                 if (prompt) _tprintf (_T("Enter new data string for
the record.\n"));
```

```
                                    _fgetts (string, sizeof(string), stdin); // Don't use
_getts (potential buffer overflow)

                                    string[_tcslen(string)-1] = _T('\0'); // remove the
newline character

                                    if (record.referenceCount == 0) {

                                                        record.recordCreationTime =
currentTime;

                                            header.numNonEmptyRecords++;

                                            headerChange = TRUE;

                                    }

                                    record.recordUpdateTime = currentTime;

                                    record.referenceCount++;

                                                _tcsncpy (record.dataString, string,
STRING_SIZE-1);

                                    recordChange = TRUE;

} else {

                                     if (prompt) _tprintf (_T("command must be r, w,
or d. Try again.\n"));}

                                  /* Update the record in place if any record contents
have changed. */

                                 if (recordChange && !WriteFile (hFile, &record, sizeof
(RECORD), &nXfer, &ov))
```

```
                                    ReportError (_T("RecordAccess: WriteFile update
failure."), 8, FALSE);

                                /* Update the number of non-empty records if required
*/

                               if (headerChange) {

                                        if (!WriteFile (hFile, &header, sizeof (header),
&nXfer, &ovZero))

                                                ReportError (_T("RecordAccess: WriteFile
update failure."), 9, FALSE);

                               }

                        }

                            if (prompt) _tprintf (_T("Computed number of non-empty
records is: %d\n"), header.numNonEmptyRecords);

                        if (!ReadFile(hFile, &header, sizeof (HEADER), &nXfer, &ovZero))

                                ReportError (_T("RecordAccess Error: ReadFile header."),
10, TRUE);

                           if (prompt) _tprintf (_T("File %s NOW contains %d non-empty
records.\nTotal capacity is: %d\n"),

                                                argv[1],  header.numNonEmptyRecords,
header.numRecords);

                        CloseHandle (hFile);

                        return 0;

                }
```

Topic 35:

Windows provide a certain set of APIs for search files/folders within the hierarchical structure of Directories/folders. These APIs include:

FindFirstFile(), FindFirstFileEx(), FindNextFile() and FindClose(). Detail of these APIs are:

FindFirstFile() API

Handle FindFirstFile(LPCSTR lpFileName, LPWIN32_FIND_DATA *A lpFindFileData);*

Where **lpFileName** represents the directory or path, and the filename. The name can include wildcard characters, for example, an asteristk (*) or a question mark (?).

lpFindFileData: A pointer to the WIN32_FIND_DATA structure that receives information about a found file or directory.

Structure or Detail of WIN32_FIND_DATA structure:

typedef struct _WIN32_FIND_DATA {

DWORD dwFileAttributes;

FILETIME ftCreationTime;

FILETIME ftLastAccessTime;

FILETIME ftLastWriteTime;

DWORD nFileSizeHigh;

DWORD nFileSizeLow;

```
    DWORD  dwReserved0;
    DWORD  dwReserved1;
    CHAR cFileName[Max_Path];
    CHAR cAlternateFileName[14];
    DWORD  dwFileType;
    DWORD  dwCreatorType;
    DWORD   WFinderFlags;
};
```

FindNextFile() API:

BOOL FindNextFile(HANDLE hFindFile, LPWIN32_FIND_DATA *A lpFindFileData*);

Where hFindFile represents the search handle returned by a previous call to the FindFirstFile or FindFirstFileEx function &

lpFindFileData is a pointer to the WIN32_FIND_DATA structure that receives information about the found file or sub-directory.

FindClose() API

BOOL FindClose(HANDLE hFindFile);

Topic 36:

Certain other APIs are also used for getting the file attributes but these API need to have an open file handle rather than scan a directory or use a filename.

GetFileTime() API

BOOL GetFileTime(HANDLE hFile, LPFILETIME lpCreationTime, LPFILETIME lpLastAccessTime, LPFILETIME lpLastWriteTime);

Where **hFile** is a handle to the file or directory.

lpCreationTime is a pointer to a FILETIME structure to receive the data and time the file or directory was created.

lpLastAccessTime is a pointer to a FILETIME structure to receive the data and time the file or directory was last accessed.

lpLastWriteTime is a pointer to a FILETIME structure to receive the data and time the file or directory was last written to truncated or overwritten.

FileTimeToSystemTime() API

It converts the file time into system time.

SystemTimeToFileTime() API

It converts the system time into file time.

CompareFileTime() API

It compares file times of two files. It returns -1 if less, 0 if equal and +1 if greater.

SetFileTime() API

It sets the three time of file. NULL used if the file time is not to be changed.

FileTimeToLocalFileTime() and LocalFileTimeToFileTime() APIs converts UTC and local File time.

GetFileType API

It provides information regarding the type of file (disk file or pipes)

GetFileAttributes() API

DWORD GetFileAttributes(LPCTSTR lpFileName);

Where lpFileName is the name of a file or directory. Its return value is:

FILE_ATTRIBUTE_DIRECTORY, FILE_ATTRIBUTE_READONLY, FILE_ATTRIBUTE_NORMAL, FILE_ATTRIBUTE_TEMPORARY.

In case of failure, the return value is INVALID_FILE_ATTRIBUTES whose value is 0xFFFFFFFF.

Topic 37:

Windows provide the facility of creating temporary files for storing the intermediate results. These files are assigned unique names in a directory with extension .tmp. Certain APIs are used for creating temporary files. These include:

GetTempFileName API

UINT GetTempFileName(LPCTSTR lpPathName, LPCTSTR lpPrefixString, UINT uUnique, LPSTR lpTempFileName);

Where **lpPathName** represents the directory path for the filename. The string cannot be longer than 14 characters.

lpPrefixString represents the null-terminated prefix string. The first 3 characters of this string is used as a prefix of the filename.

uUnique represents an unsigned integer to be used in creating the temporay filename.

lpTempFileName is a pointer to the buffer that receives the temporary filename

Topic 38:

We can get the attributes of a file, listing of files and can traverse the directory structure using certain windows APIs.

An application called **lsW** is used for showing files and listing their attributes. It uses two option switched that is -l and -R where -l option is used to list the attributes of files in a folder and -R is used for recursive traversal through subfolders.

This application or program will work with a relative pathname; it will not work with absolute pathname.

Program: File Listing and Directory Traversal

```
#include<everything.h>

BOOL TraverseDirectory(LPTSTR, LPTSTR, DWORD, LPBOOL);

DWORD FileType(LPWIN32_FIND_DATA);

BOOL ProcessItem(LPWIN32_FIND_DATA, DWORD, LPBOOL);

Int  _tmain(int argc, LPTSTR argv[])

{

  BOOL flags[MAX_OPTIONS], ok=TRUE;

  TCHAR searchPattern[MAX_PATH +1], currPath[MAX_PATH_LONG +1],
parrentPath[MAX_PATH_LONG +1];

LPTSTR pSlash, pSearchPattern;

int i, fileIndex;
```

```
DWORD pathLength;

fileIndex = Options(argc, argv, _T("RI"), &flags[0], &flags[1], NULL);

/* parse the search pattern into two parts: the parent and the filename or wild card
expression. The filename is the longest suffix not containing a slash. The parent is the
remaining prefix with a slash. This is performed for all command line search pattern. If
no file is specified, use * as the search pattern */

pathLength = getCurrentDirectory(MAX_PATH_LONG, currPath);

if(pathLength==0 || pathLength>= MAX_PATH_LONG)

{    /* pathLength>= MAX_PATH_LONG (32780) should be impossible  */

  ReportError(_T("GetCurrentDirectory failed", 1, TRUE);
```

Topic 39: Setting File Times

UNIX Touch command changes file access and changes the time to current system time.

For changing file time the GetSystemTimeAsFileTime is more convenient than calling Get SystemTime followed by SystemTimeToFileTime.

These two functions are provided by sysinfoapi.h api used to retrieve system time.
GetSystemTimeAsFileTime() take file pointer as argument and retrieves the current system date and time in Coordinated Universal Time (UTC) format.

GetSystemTime() A pointer to a SYSTEMTIME structure to receive the current system date and time. Retrieves the current system date and time in Coordinated Universal Time (UTC) format.

SetFileTime() Sets the date and time that the specified file or directory was created, last accessed, or last modified.

Topic 40: File Processing Strategies

C Library for file processing have following features:

- Code is portable to Operating system running other than windows.
- Convenient line and character oriented functions are simply string processing.
- The functions are higher level and easy to use than windows functions.
- Line and stream character-oriented functions can be migrated to generic calls.

C library also have some limitations but some performance advantages are also present when compared to windows functions.

Topic 41: File Locking

Windows OS can lock files completely or some part of a file.There are two functions avalible LockfileEx and LockFile.

The syntax and parameters list of LockfileEx is given below:

```
BOOL LockFileEx(
        HANDLE hFile,
        DWORD dwFlags,
        DWORD dwReserved,
        DWORD nNumberOfBytesToLockLow,
        DWORD nNumberOfBytesToLockHigh,
        LPPVERLAPPED lpOverlapped
)
```

To Unlock a file UnlockFileEx() function is used.Parameters list is highly similar to LockFileEx.

Topic 42: Releasing file locks

If a program does not release a lock or holds the lock longer, other programs will not be able to proceed and their performance will be negatively impacted.

Therefore, programs should be designed in such a way that locks are released after the usage is completed using appropriate functions.

Topic 43: Lock Logic consequences

- File locking can generate deadlocks just like deadlocks generated by mutual exclusion locks.
- A read or write may be able to complete its request before encountering a conflicting lock.

- The read or write will return false, and the byte transfer count will be less than the number requested.
- Shared locks if not used properly also generate unexpected errors, as if one process have a shared lock on a file other processes can only ready the file but cannot write into it.

Topic 44: The Registry

The Windows Registry is a hierarchical database that stores low level settings for the Microsoft Windows operating system and for applications that opt to use the registry. The OS kernel, device drivers, services, Security Accounts Manager, and user interfaces all can read and update the registry values.

In other words, the registry or Windows Registry contains different information e.g. settings, options, and other values for programs and hardware installed on the system. For example, when a program is installed, a new key containing settings such as its version is added to the Windows Registry.

Topic 45: Registry Keys

The registry contains two basic elements: keys and values. Registry keys are container objects similar to folders. Registry values are non-container objects similar to files. Keys may contain values and sub-keys. Keys are referenced with a syntax similar to Windows' path names, using backslashes to indicate levels of hierarchy. Keys must have a case insensitive name without backslashes.

The hierarchy of registry keys can only be accessed from a known root key handle (which is anonymous but whose effective value is a constant numeric handle) that is mapped to the content of a registry key pre-loaded by the kernel from a stored "hive", or to the content of

a sub-key within another root key, or mapped to a registered service or DLL that provides access to its contained sub-keys and values.

E.g. HKEY_LOCAL_MACHINE\Software\Microsoft\Windows refers to the sub-key "Windows" of the sub-key "Microsoft" of the sub-key "Software" of the HKEY_LOCAL_MACHINE root key.

There are seven predefined root keys, traditionally named according to their constant handles defined in the Win32 API, or by synonymous abbreviations (depending on applications):[4]

- HKEY_LOCAL_MACHINE or HKLM
- HKEY_CURRENT_CONFIG or HKCC
- HKEY_CLASSES_ROOT or HKCR
- HKEY_CURRENT_USER or HKCU
- HKEY_USERS or HKU
- HKEY_PERFORMANCE_DATA (only in Windows NT, but invisible in the Windows Registry Editor)[5]
- HKEY_DYN_DATA (only in Windows 9x, and visible in the Windows Registry Editor)

Topic 46: Key Management

Service Manager stores many settings in the registry. You seldom have to edit the registry yourself, because most of those settings are derived from entries that you make in day-to-day use. However, some changes to settings might occasionally be required. Service Manager stores most registry values in the following locations:

HKEY_CURRENT_USER\Software\Microsoft\System Center<version>\Service Manager\Console
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\System Center<version>

Topic 47: Listing Registry Keys

You can show all items directly within a registry key by using Get-ChildItem. Add the optional Force parameter to display hidden or system items. For example, this command displays the items directly within PowerShell drive HKCU:, which corresponds to the HKEY_CURRENT_USER registry hive:

PowerShell

```
Get-ChildItem -Path HKCU:\ | Select-Object Name
```

You can also specify this registry path by specifying the registry provider's name, followed by ::. The registry provider's full name is Microsoft.PowerShell.Core\Registry, but this can be shortened to just Registry. Any of the following commands will list the contents directly under HKCU:.

PowerShell

```
Get-ChildItem -Path Registry::HKEY_CURRENT_USERGet-ChildItem -Path Microsoft.PowerShell.Core\Registry::HKEY_CURRENT_USERGet-ChildItem -Path Registry::HKCUGet-ChildItem -Path Microsoft.PowerShell.Core\Registry::HKCUGet-ChildItem HKCU:
```

Topic 48: Exception and their Handlers

Structured exception handling (SEH) is a Microsoft extension to C to handle certain exceptional code situations, such as hardware faults, gracefully. Although Windows and Microsoft C++ support SEH, we recommend that you use ISO-standard C++ exception handling. It makes your code more portable and flexible. However, to maintain existing code or for particular kinds of programs, you still might have to use SEH.

Topic 49: Try and Except Blocks

Different languages have different implementation for exception handling.

For example Python handles the exception following way:
The try block lets you test a block of code for errors.
The except block lets you handle the error.
The else block lets you execute code when there is no error.
The finally block lets you execute code, regardless of the result of the try- and except blocks.

Topic 50: Filter Expressions

Writing an exception filter in windows

You can handle an exception either by jumping to the level of the exception handler or by continuing execution. Instead of using the exception handler code to handle the exception and falling through, you can use a filter expression to clean up the problem. Then, by returning EXCEPTION_CONTINUE_EXECUTION (-1), you may resume normal flow without clearing the stack.

It's a good idea to use a function call in the filter expression whenever filter needs to do anything complex. Evaluating the expression causes execution of the function, in this case, Eval_Exception.

This handler passes control to another handler unless the exception is an integer or floating-point overflow. If it is, the handler calls a function (ResetVars is only an example, not an API function) to reset some global variables. The __except statement block, which in this example is empty, can never be executed because Eval_Exception never returns EXCEPTION_EXECUTE_HANDLER (1).

Using a function call is a good general-purpose technique for dealing with complex filter expressions. Two other C language features that are useful are:

- The conditional operator
- The comma operator

Topic 52: Exceptions Handling Sequence

Exception Handling sequence for python is given below:

try:
  You do your operations here;
  .....
except ExceptionI:
  If there is ExceptionI, then execute this block.
except ExceptionII:
  If there is ExceptionII, then execute this block.
  .....
else:
  If there is no exception then execute this block.

Topic 53: Floating-Point Exceptions

IEEE defines a standard for floating-point exceptions it is called IEEE Standard for Binary Floating-Point Arithmetic (IEEE 754). This standard have defined five types of floating-point exception:

- Invalid operation
- Division by zero
- Overflow
- Underflow
- Inexact calculation

Topic 54: Errors and Exceptions

Errors	Exception
Errors are usually raised by the environment in which the application is running.	Exceptions are caused by the code of the app the code belongs to.
It is not possible to recover from an error.	The use of **try-catch** blocks can handle exceptions and recover the system from exception.
Errors occur at run-time and are unknown by the compiler.	Exceptions may or may not be caught by the compiler.
Programmers include an explicit test to check for error,for example whether a file read/write operation has failed.	An exception could occur nearly anywhere, and it is practical to test for an exception.

Topic 55: Treating Errors as Exceptions

Programmer can use ReportError function to report the error and let windows treat it and terminate the process.But it have its own limitations.

• A fatal error terminates the entire process when only a single thread should terminate.

• The programmer may require to continue program execution rather than terminate the process.

• Synchronization resources, such as events or semaphores, will not be released in most of the circumstances.

Topic 56: Termination Handlers

Termination handler provides same functionality as an exception handler,but it is executed when a thread leaves a block as a result normal program flow as well as when an exception occurs. A termination handler cannot diagnose an exception.
A termination handler consists of the following elements.

- A guarded body of code
- A block of termination code to be executed when the flow of control leaves the guarded body

Termination handlers are declared in language-specific syntax. Using the Microsoft C/C++ Optimizing Compiler, they are implemented using **__try** and **__finally**.

The guarded body of code can be a block of code, a set of nested blocks, or an entire procedure or function.
The termination block is executed when the flow of control leaves the guarded body, regardless of whether the guarded body terminated normally or abnormally.

Topic 57: Better Programs with Termination Handlers

Termination and exception handlers allow you to make your program more robust by both simplifying recovery from errors and exceptions and helping to ensure that resources and file locks are freed at critical junctures.

Consider the below given code where we are checking for invalid handle and ReportingException.

If (hin == INVALID_HANDLE_VALUE)

ReportException(argv[iFile],1);
If(!GetFileSizeEx(hIn, &fsize) || fsize.HighPart > 0)
ReportException(_T("FIle is Too Large",1));

Topic 58: Filter Functions

The filter function identifies the exception type and based on the type of the exception, the handler can treat each exception differently. In the following program, the exception handling and termination of a program are illustrated using a filter function.

The program generates an exception based on the type of the exception entered by the user. The floating point exceptions are enabled with the help of controlfp function and old status is saved in the fpOld. The try block mentions the different cases of exception generation with the help of a switch statement.

In the except clause, there is a filter function calling another function GetExceptionInformation(). eCategory is a reference variable whose value determines the type of exception being thrown. For the outer try, there is a finally clause used for any abnormal termination of the program. The old value of the floating point mask is also restored at this stage.

Now we will see how the values in the ecategory reference variable are placed.

User generated exceptions are identified based on the masking operation generating zero as a result.

Topic 59: Console Control Handles

Console control handlers are quite similar to the mechanism of exception handlers. Normal exceptions respond to several asynchronous events like division by zero, invalid page fault etc., but they do not respond to console related events like Ctrl+C. Console control handlers can detect and handle the events that are console related. The API SetConsoleCtrlHandler() is used to add console handlers.

BOOL SetConsoleCtrlHandler(PHANDLER_ROUTINE HandlerRoutine, BOOL Add);

The API takes the address of the HandlerRoutine and Add Boolean as parameters. There can be a number of handler routines if the Add parameter is set as TRUE. If the HandlerRoutine parameter is set as NULL and Add is TRUE, then the Ctrl-C signal will be ignored.

BOOL HandlerRoutine (DWORD dwCtrlType);

The return type of HanlderRoutine should be Boolean taking only one parameter of type DWORD. The HandlerRoutine() function takes a DWORD as a parameter and returns a Boolean value.

dwCtrlType identifies the signal and its values can be:

1. CTRL_C_EVENT: Indicates that a Ctrl-C sequence was entered through keyboards.

2. CTRL_CLOSE_EVENT: Indicates that the console window is being closed.

3. CTRL_BREAK_EVENT: Indicates Ctrl-Break sequence.

4. CTRL_LOGOFF_EVENT: Indicates the user is trying to logoff.

5. CTRL_SHUTDOWN_EVENT: Indicates the user is trying to shut down.

The handler routine will be invoked if a console exception is detected. Handler routine runs in an independent thread within the process. Raising an exception within the handler routine will not interfere with the working of the original routine that created the handler routine. Signals apply to the whole process, while exception applies to a single thread.

Usually signal handlers are used to perform cleanup tasks whenever a shutdown, close or logoff events are detected. Signal Handler would return a TRUE value in case it takes care of the task or it may return FALSE. In case of FALSE, the next handler in the chain is invoked. Signal handler chain is invoked in the reverse order of which they are set up in and the system signal handler is the last one in this chain.

Topic 60: Console Control Handler Example

We have a simple program that set up a console control handler and starts beeping in a loop. The control handler is invoked whenever a console event occurs. The handler handles the event likewise and clears an exitFlag to end the loop of the main function.

```
/* Chapter 4. CNTRLC.C */
/* Catch Cntrl-C signals. */
#include "Everything.h"
static BOOL WINAPI Handler(DWORD cntrlEvent);
static BOOL exitFlag = FALSE;
int _tmain(int argc, LPTSTR argv[])

/* Beep periodically until signaled to stop. */
{
        /* Add an event handler. */
    if (!SetConsoleCtrlHandler(Handler, TRUE))
```

```
                ReportError(_T("Error setting event handler"), 1, TRUE);

        while (!exitFlag) { /* This flag is detected right after a beep, before a handler exits */
                Sleep(4750); /* Beep every 5 seconds; allowing 250 ms of beep time. */
                Beep(1000 /* Frequency */, 250 /* Duration */);
        }
}
        _tprintf(_T("Stopping the main program as requested.\n"));
        return 0;
}
BOOL WINAPI Handler(DWORD cntrlEvent)
{
        switch (cntrlEvent) {
                /* The signal timing will determine if you see the second handler message */
                case CTRL_C_EVENT:
                        _tprintf(_T("Ctrl-C received by handler. Leaving in 5 seconds or
                                less.\n"));
                        exitFlag = TRUE;
                        Sleep(4000); /* Decrease this time to get a different effect */
                        _tprintf(_T("Leaving handler in 1 second or less.\n"));
                        return TRUE; /* TRUE indicates that the signal was handled. */
                case CTRL_CLOSE_EVENT:
                        _tprintf(_T("Close event received by handler. Leaving the handler in 5
                                seconds or less.\n"));
                        exitFlag = TRUE;
                        Sleep(4000); /* Decrease this time to get a different effect  */
                        _tprintf(_T("Leaving handler in 1 second or less.\n"));
                        return TRUE; /* Try returning FALSE. Any difference? */
                default:
                        _tprintf(_T("Event: %d received by handler. Leaving in 5 seconds or
                                less.\n"), cntrlEvent);
                        exitFlag = TRUE;
                        Sleep(4000); /* Decrease this time to get a different effect */
                        _tprintf(_T("Leaving handler in 1 seconds or less.\n"));
                        return TRUE; /* TRUE indicates that the signal was handled. */
        }
}
```

The program can be terminated by the user either by closing the console or with a Ctrl-C. The handler will register with windows using the SetConsoleCtrlHandler(Handler, TRUE) function. The handler will be activated upon the occurrence of any console event. If the registration of any handler fails due to any reason, then an error message will be printed.

Topic 61: Vectored Exceptions Handling

Exception handling functions can be directly associated with an exception just like console handlers. In case a vectored exception is set up, then windows first looks for Vectored Exception Handlers (VEH) and then unwinds the stack.

No __try and __catch keywords are required with VEH and they are like console control handlers. Windows provides a set of APIs for VEH management as follows.

PVOID WINAPI AddVectoredExceptionHandler(ULONG FirstHandler, PVECTORED_EXCEPTION_HANDLER VectoredHandler);

The given API has two parameters; FirstHandler is the parameter used to specify the order in which the handler executes. Non-zero value indicates that it will be the first one to execute and zero specifies it to be the last. If there are more than one handlers setup with zero value, then they will be invoked in the order they are added using AddVectoredExceptionHandler(). Return value is NULL in case of failure, otherwise it returns the Handle to the Vectored Exception Handler.

A call to RemoveVectoredExceptionHandler() is used to remove the vectored exception from a chain. Following is the structure of a vectored exception handler to be set up by the AddVectoredExceptionHandler().

LONG WINAPI VectorHandler(PEXCEPTION_POINTERS ExceptionInfo);

PEXCEPTION_POINTER is a pointer to EXCEPTION_POINTER structure that was previously retrieved by GetExceptionInformation().

Pitfalls of Vectored Exceptions

The exception handler function should be fast and must return as quickly as possible, therefore it should not have a lot of code. The VEH should neither perform any blocking operation like the Sleep() function nor use any synchronization objects. Typically a VEH would access exception information structure, do some minimal processing, and set a few flags.

VEH uses the same return values as SHE.

EXCEPTION_CONTINUE_EXECUTION : No more handlers invoke, exception stack does not unwind, and the execution continues from the point where the exception occurred.

EXCEPTION_CONTINUE_SEARCH : The next VEH handler is probed, if there are no additional VEH handlers, then SEH stack is unwound.

Topic 62: Memory Management

Dynamic Memory

Need of dynamic memory arises whenever dynamic data structures like search tables, trees, linked lists etc. are used. Windows provides a set of APIs for handling dynamic memory allocation.

Windows also provides memory mapped files which allows direct movement of data to and from user space and files without the use of file APIs. Memory Mapped files can help conveniently handle dynamic data structure and make file handling faster because they are treated just like memory. It also provides a mechanism for memory sharing among processes.

Windows Memory Management

Windows essentially uses two API platforms i.e. Win32 and Win64.

The Win32 API uses pointers of size 32 bits, hence the virtual space is 2^32. All data types have been optimized for 32 bit boundaries. Win64 uses a virtual space of 2^64 (16 Exabytes).

A good strategy is to design an application in such a way that it could run in both modes without any change in code.

Win32 makes at least half of the virtual space (8GB) accessible to a process and the rest of the space is reserved by the system for shared data, code, and drivers etc. Overall Windows provides a large memory space available to user programs and hence requires optimal management.

Topic 63: Windows Memory Management Overview

Dynamic Memory

Need of dynamic memory arises whenever dynamic data structures like search tables, trees, linked lists etc. are used. Windows provides a set of APIs for handling dynamic memory allocation.

Windows also provides memory mapped files which allows direct movement of data to and from user space and files without the use of file APIs. Memory Mapped files can help conveniently handle dynamic data structure and make file handling faster because they are treated just like memory. It also provides a mechanism for memory sharing among processes.

Windows Memory Management

Windows essentially uses two API platforms i.e. Win32 and Win64.

The Win32 API uses pointers of size 32 bits, hence the virtual space is 2^32. All data types have been optimized for 32 bit boundaries. Win64 uses a virtual space of 2^64 (16 Exabytes).

A good strategy is to design an application in such a way that it could run in both modes without any change in code.

Win32 makes at least half of the virtual space (8GB) accessible to a process and the rest of the space is reserved by the system for shared data, code, and drivers etc. Overall Windows provides a large memory space available to user programs and hence requires optimal management.

Further information about the parameters of Windows Memory Management can be probed using the following API.

VOID GetSystemInfo(LPSYSTEM_INFO lpSysInfo)

The API returns a pointer to SYSTEM_INFO structure. The structure contains various information regarding the system like page size, granularity, and application's physical memory address space.

Topic 64: Introduction to Heaps

A programmer allocates memory dynamically from a heap. Windows maintains a pool of heaps and a process can have many heaps. Traditionally, one heap is considered enough. But several heaps may be required to make a program more efficient.

In case a single heap is sufficient, then a runtime library function for heap allocation like malloc(), free(), calloc(), realloc() might be enough.

Heap is a windows object and hence is accessed by a handle. Whenever you require allocating memory from heap, you need a heap handle. Every process in windows has a default heap which can be accessed through the following API.

HANDLE GetProcessHeap(VOID)

The API returns a handle to the process heap. NULL is returned in case of failure and not INVALID_HANDLE_VALUE.

However, due to a number of reasons it would be desirable to have more than one heap. Sometimes it is convenient to have distinct heaps for different data structures.

Separate Heaps

Following are the benefits of separate heaps:

1. If a distinct heap is assigned to each thread, then each thread will only be able to use the memory allocated to each thread.

2. Separation of memory space among threads reduces memory contention.

3. Fragmentation is reduced when one fixed size data structure is allocated from a single heap.

4. Allocating a single heap among each thread simplifies synchronization.

5. If a single heap contains complex data structures, then they can be easily de-allocated with a single API call by de-allocating the entire heap. We will not need complex de-allocation algorithms in such cases.

6. Small heaps for a single data structure reduces the chances of page faults as per the principle of locality.

Topic 65: Creating Heaps

We can create a new heap using HeapCreate() API and its size can be set to zero. The API adjusts the heap size to the nearest multiple of page size. Memory is committed to the heap initially, rather than on demand. In case the memory requirements increase than the initial, more pages will automatically be allocated to the heap up to maximum size allowed.

If the required memory is not known, then deferring memory commitment is a good practice as heap is a limited resource. Following is the syntax of the API used to create new heaps.

HANDLE HeapCreate(DWORD flOptions, SIZE_T dwInitialSize, SIZE_T dwMaximumSize);

dwMaximumSize if non-zero, determines the maximum limit of the heap memory set by the user. Heap is not grow-able beyond this point. In case it's zero, then the heap is grow-able to the extent of the virtual memory space available for the heap.

dwInitialSize is the initial size of the heap set by the programmer. SIZE_T is used to enable portability. Based on the win32 or win64 platforms, SIZE_T will be 32 or 64 bit wide.

flOptions is usually a combination of the following three flags predefined in Windows:

1. HEAP_GENERATE_EXCEPTIONS: This flag will raise exceptions in case there is any failure while allocating memory to heap. Exceptions are not generated by CreateHeap(), rather they may occur at the time of allocation.

2. HEAP_NO_SERIALIZE: This option provides a slight performance improvement when serialization of data is not required.

3. HEAP_CREATE_ENABLE_EXECUTE: It is an advanced option that allows you to execute code from heap. Usually an exception will occur if data from heap is interpreted as code due to the data execution prevention (DEP) restriction by Windows.

CloseHandle() is not appropriate to dispose of a heap handle rather HeapDestroy() is used as follows.

BOOL HeapDestroy(

HANDLE hHeap

);

hHeap is the handle to a previously created heap. Do not use the handle obtained from GetProcessHeap() because it may raise an exception. This is an easy way to get rid of all the contents of the heap including complex data structures.

Topic 66: Managing Heap Memory

Once a heap is created, it does not allocate memory that is directly available to the program. Rather, it only creates a logical structure of heap that will be used to allocate new memory blocks. Memory blocks are allocated using heap memory allocation APIs like HeapAlloc() and HeapReAlloc().

LPVOID HeapAlloc(HANDLE hHeap, DWORD dwFlags, SIZE_T dwBytes);

hHeap is the handle of the heap from which memory is to be allocated. dwFlags are quite similar to the flags used in HeapCreate().

HEAP_GENERATE_EXCEPTIONS: This flag will raise exceptions in case there is any failure while allocating memory to heap. Exceptions are not generated by CreateHeap(), rather they may occur at the time of allocation.

HEAP_NO_SERIALIZE: This option provides a slight performance improvement when serialization of data is not required.

The first flag HEAP_GENERATE_EXCEPTION is ignored if it is already set in HeapCreate().

HEAP_ZERO_MEMORY: Specifies that the allocated memory will be initialized to zero.

dwBytes is the size of the memory block to be allocated. For non-grow-able heap, its 0x7FFF8 approximately equivalent to 0.5 MB.

The return value of the function is LPVOID. This is the address of the allocated memory block. Use this pointer in a formal way and there is no need to make any reference to the Heap handle.

In case HEAP_GENERATE_EXCEPTION flag is set, an exception is generated if failure occurs. The exception is either STATUS_NO_MEMORY or STATUS_ACCESS_VIOLATION.

If the exception flag is not set, then NULL is returned by HeapAlloc() and the GetLastError() does not work on HeapAlloc().

BOOL HeapFree(HANDLE hHeap, DWORD dwFlags,

_Frees_ptr_opt_ LPVOID lpMem);

hHeap is the heap handle from which memory is to be allocated. dwFlags should be 0 or set to HEAP_NO_SERIALIZE. lpMem should be the pointer previously returned by HeapAlloc() or HeapReAlloc().

Return value of FALSE will indicate a failure. GetLastError() can be used to get the error.

LPVOID HeapReAlloc(HANDLE hHeap, DWORD dwFlags,

_Frees_ptr_opt_ LPVOID lpMem, SIZE_T dwBytes);

hHeap is the heap handle from which memory is to be allocated. dwFlags: HEAP_GENERATE_EXCEPTION and HEAP_NO_SERIALIZE are quite similar to the flags used in HeapAlloc().

HEAP_ZERO_MEMORY: only the newly allocated memory is set to zero (in case dwBytes is greater than the previous allocation).

HEAP_REALLOC_IN_PLACE_ONLY: It indicates the newly allocated block should lie contiguously to the previously allocated block.

lpMem: specifies the pointer to the block previously allocated to the same heap hHeap.

dwBytes: It refers to the block size to be allocated that can be lesser or greater than the previous allocation. But the same restriction as HeapAlloc applies i.e. the block size cannot be greater than 0x7FFF8.

Topic 67: Heap Size and Serialization

Some programs may require to determine the size of allocated blocks in the Heap. The size of the allocated block is determined using the API HeapSize() as follows.

```
SIZE_T HeapSize(
  HANDLE  hHeap,
  DWORD   dwFlags,
  LPCVOID lpMem
);
```

The function returns the size of the block or zero in case of failure. The only valid dwFlag is HEAP_NO_SERIALIZE.

Serialization

Serialization is required when dealing with concurrent threads using some common resource. Serialization is not required if threads are autonomous and there is no possibility of concurrent threads disrupting each other.

The flag HEAP_NO_SERIALIZE provides an estimated performance advantage of 16%. Using this flag can improve performance if there are no conflicts. This flag can be conveniently used if:

a. The program is single threaded.

b. Each thread has its own heap that is insulated from other threads.

c. The program has its own mutual exclusion mechanism.

Heap Exceptions

Heap exceptions are enabled using the flag HEAP_GENERATE_EXCEPTION. This allows the program to close open "handles" before a program terminates. There can be two scenarios with this option:

a. STATUS_NO_MEMORY: It means there is no more memory to be allocated because the heap has reached its limits.

b. STATUS_ACCESS_VIOLATION: It indicates that the heap has been corrupted. Either the heap has been accessed as code, or has been accessed beyond its bounds.

There are some other functions that can be used while working with heaps. For example,

HeapSetInformation() can be used to enable low fragmentation mode. It can also be used to allow termination of a thread upon heap's corruption.

HeapCompact() allows finding the largest allocated block in the heap.

HeapValidate() allows detecting heap corruption.

HeapWalk() enumerates the allocated blocks on heap.

HeapLock() and HeapUnLock() allow serializing access to the heap.

Topic 68: Using Heaps

Using Memory Management APIs

Till now we have used the Memory Management APIs for allocating, reallocating and deallocating heaps. We can also get the size of a heap through an API.

A typical methodology of dealing with heaps should be to get a heap handle either using HeapCreate() or GetProcessHeap(). Use the handle obtained from the above to allocate memory blocks from the heap using HeapAlloc(). If some block needs to be deallocated, use HeapFree(). Before the program is terminated or when the heap is not required, use HeapDestroy() to dispose of the heap.

It is convenient not to mix up windows heap API and Run Time Library functions. Anything allocated with C library functions should also be deallocated with C library functions.

Topic 69: Working with Heaps

A Sorting File example using heaps

The example is formulated using two heaps. The first one will be a node heap, while the other is a record heap. Node heap will be used to build a tree, while the data heap will be used to store keys.

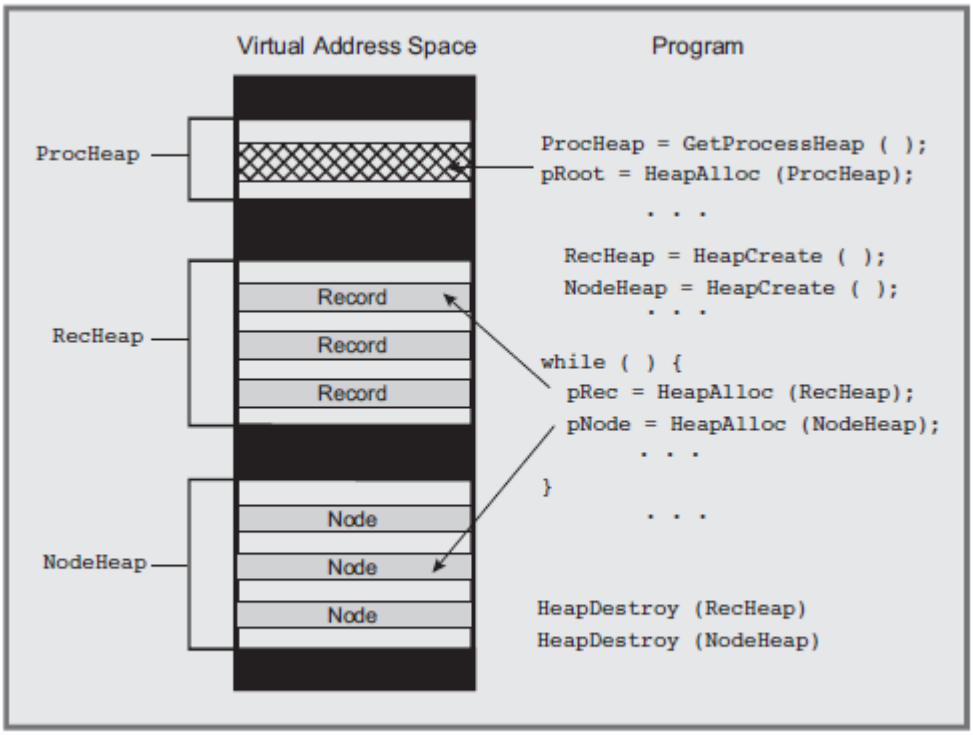


Figure 5–2 Memory Management in Multiple Heaps

Following are the three heaps as shown in the figure:

1. ProcHeap

2. RecHeap

3. NodeHeap

ProcHeap contains the root address, but RecHeap stores the records. NodeHeap on the other hand stores the nodes when they are created. Sorting will be performed in the NodeHeap that gives a reference to be searched in the RecHeap. The data structure will be maintained in the NodeHeap. At the end, all the heaps will be destroyed except ProcHeap because it is created using GetProcessHeap().

Topic 82: **Exporting and Importing Interfaces**

For a DLL function to be useful for an application exporting it, it must be declared as exportable. This can be done by using .DEF file or by using __declspec (dllexport) storage modifier.

__declspec (dllexport)DWORD MyFunction (…)

The build process will generate a .LIB file and .DLL file. The .LIB files will a stub for function calls while DLL file will hold the actual code. Similarly the calling program should use the __declspec (dllimport) Storage modifier.

Following Macro can be used with the library file code

#if defined (MYLIBRARY_EXPORTS)

#define LIBSPEC _declspec(dllexport)

#elif defined (__cplusplus)

#define LIBSPEC extern “C” __declspec(dllimport)

#else

#define LIBSPEC __declspec (dllimport)

#endif

LIBSPEC SWORD MyFunction (…)

If you are using Visual C++ compiler then this task is automatically performed by the compiler. When building the calling program you need to specify the .LIB file. When executing the calling program the .DLL file must be placed in the same directory as the application. Following is the default DLL search safe order for explicit and implicit linking.

- Directory containing the application
- The system directory. This path can be determined by GetSystemDirectory()
- The windows directory (GetWindowsDirectory())
- The current directory
- Directories specified by the PATH environment variable

Topic 83: **Explicit Linking**

Explicit linking requires the program to explicitly a specific DLL to be loaded or freed. Once the DLL is loaded then the program obtains the address of the specific entry point and uses that address as a pointer in function call.

The function is not declared in the calling program. Rather a pointer to a function is declared. The functions required are:

HMODULE LoadLibrary(LPCTSTR lpLibFileName);

HMODULE LoadLibraryEx(LPCTSTR lpLibFileName,

HANDLE hFile,

DWORD dwFlags);

HMODULE is NULL in case of failure.

File name need not mention the extension. The file path must be valid. See MSDN for details of dwFlags Once the DLL is loaded. The programmer needs to obtain an entry point (procedure address) into the DLL. This is done using:

FARPROC GetProcAddress(HMODULE hModule, LPCSTR lpProcName);

hModule is the module handle obtained for the DLL. lpProcName cannot b UNICODE. It is the name of the function whose entry point is to be obtained. Its return type is FARPROC. This is the far address of the function. A C type function pointer declaration can be easily used to invoke the function in DLL and pass its parameters.

Topic 84: **Explicit Linking a file conversion function**

Previously, we have studied many file conversion functions. Some used memory mapped IO, some used file operations. Some performed file conversion some performed file encryption.

Now, we take a look at how we can encapsulate these functions in a DLL and invoke them explicitly.

Following example shows explicit linking of these functions.

HMODULE hDLL;

FARPROC pcci;

TCHAR YNResp[5] = YES;

if (argc < 5)

ReportError (_T("Usage: cciEL shift file1 file2 DllName"), 1, FALSE);

/* Load the cipher function. */

```
hDLL = LoadLibrary (argv[4]);

 if (hDLL == NULL)

ReportError (_T("Failed loading DLL."), 4, TRUE);

 /* Get the entry point address. */

 pcci = GetProcAddress (hDLL, _T("cci_f"));

if (pcci == NULL)

ReportError (_T ("Failed of find entry point."), 5, TRUE);

cci_f = (BOOL (__cdecl *)(LPCTSTR, LPCTSTR, DWORD)) pcci;

 /* Call the function. */
```

Topic 85: **DLL Entry Point**

DLL Entry point can be specified optionally when you create a DLL. The code at entry point executes whenever a process attaches to the DLL. In case of implicit linking the DLL attaches at the time of process start and detaches when process ends. In case of explicit linking the process attaches when LoadLibrary()/LoadLibraryEx() is invoked. Also the process detaches when FreeLibrary() is called.

LoadLibraryEx() can also suppress the execution of entry point. Further, entry point is also invoked whenever a thread attaches to the DLL.

The Entry point in DLL is defined as:

```
BOOL DllMain(

HINSTANCE hDll,

DWORD Reason,

LPVOID lpReserved)
```

hDll corresponds to the handle returned by LoadLibrary(). lpReserved if NULL, represent explicit attachment else it represents implicit attachment. Reason will have on the four values

DLL_PROCESS_ATTACH

DLL_THREAD_ATTACH

DLL_THREAD_DETACH

DLL_PROCESS_DETACH

Based on the value of Reason the programmer can decide whether to initialize or free resources. All the calls to DllMain are serialized by the system. Serialization is critically important as DllMain() is supposed to perform initialization operations for each thread. There should be no blocking calls, IO calls or wait functions as they will indefinitely block other threads. A call to other DLLs from DllMain() cannot be performed except for few exceptions. LoadLibrary() and LoadLibraryEx() should never be called from DllMain() as it will create more DLL entry points. DisableThreadLibraryCalls() can be used to disable thread attach/detach calls for a specified instance.

Topic 86: DLL Version Management

Complications

Many times a DLL is upgraded to provide more features and new symbols. Also, multiple processes share a single implementation of DLL. This strength of DLL may also lead to some complications.

- If the new DLL has changed interfaces this render problems for older programs that have not been updated for newer version.
- Application that require newer updated functionality may sometime link with older DLL version.

Version Management

One intuitive resolution to the problem can be by using different directory for each version. But there are several solutions. Use DLL version number as part of the .DLL and .LIB file names eg. Utils_4_0.DLL and Utils_4_0.LIB. Applications requiring DLLs can determine the version requirements and subsequently access files with distinct filename using implicit or explicit linking.

Microsoft has introduced a concept of side by side DLLs or assemblies. This solution requires application to declare its DLL requirements using XML.

Querying DLL Information

Microsoft DLLs support a callback function that version information and more regarding a DLL. This callback function can be used dynamically to query the information regarding DLL. The works as follows:

HRESULT CALLBACK DllGetVersion(

DLLVERSION *pdvi

)

Information regarding the DLL is placed in the DLLVERSION structure. It contains a field cbSize which is the size of the structure,

dwMajorVersion

dwMinorVersion

dwBuilderNumber

dwPlatformID

dwPlatformID can be set of DLLVER_PLATFORM_NT if DLL cannot run on windows 9x or to DLLVER_PLATFORM_WINDOWS if there are no restrictions. cbSize should be set to sizeof(DLLVERSION)

Topic 87: Windows Processes and Threads

Windows Process Management is all about:

- Multitasking Systems
- Multiprocessor systems
- Thread as a unit of execution

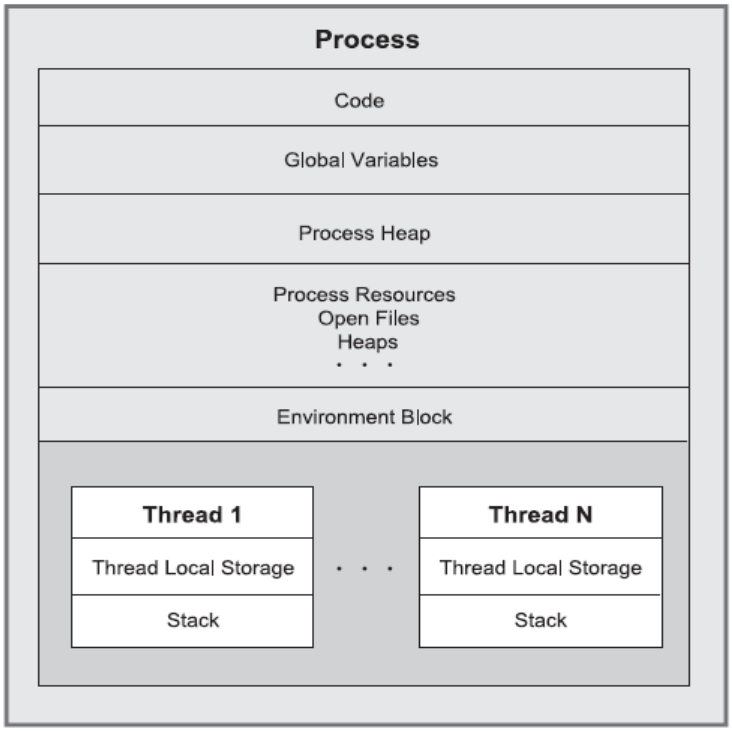
Resources of Processes

From programmer's perspective each process has resources such as one or more threads distinct virtual address space. Although, processes can share memory and files but the process itself lie in an individual virtual memory space.

- One or more code segment including DLLs
- One or more data segment including global variables
- Environment block containing environment variables (current path)

Resources of Threads

Each thread is a unit within the process. Nevertheless, it has resources of its own. Stack: for procedure calls, interrupts, exception handling, and auto variables. Thread Local Storage (TLS): An array like collection of pointers enabling a thread to allocate storage to create its unique data environment. An argument on stack unique for the created thread. A structure containing the context (internal registers status) of the thread.



Topic 88: **Process Creation**

The most fundamental process management function in windows is CreateProcess(). Windows does not have any structure that keeps account of the parent-child processes. The process that creates a child process is considered as parent process. CreateProcess() has 10 parameters. It does not returns a HANDLE. Rather two handles, one for process and one for thread is returned in a parameter struct.

It creates a process with single primary thread. Windows does not have any structure that keeps account of the parent-child processes. The process that creates a child process is considered as parent process.

CreateProcess() has 10 parameters. It does not return a HANDLE. Rather two handles, one for process and one for thread is returned in a parameter struct. One must be very careful while closing both these handles when they are not needed. Closing the thread handle does not terminate the thread it only deletes the reference to the thread within the process.

BOOL CreateProcess(LPCTSTR lpApplicationName, LPTSTR lpCommandLine,

LPSECURITY_ATTRIBUTES lpProcessAttributes, LPSECURITY_ATTRIBUTES

lpThreadAttributes, BOOL bInheritHandles, DWORD dwCreationFlags, LPVOID

lpEnvironment, LPCTSTR lpCurrentDirectory, LPSTARTUPINFO lpStartupInfo,

LPPROCESS_INFORMATION lpProcessInformation);

lpApplicationName and lpCommandLine specify the program name and the command line parameters. lpProcessAttributes and lpThreadAttributes points to the process and thread's security attribute structure. bInheritHandles specifies whether the new process inherits copies of the calling process's inheritable handles. dwCreationFlags combines several flags

CREATE_SUSPENDED: indicates that the primary thread is a suspended thread and will continue if the process invokes ResumeThread()

DETACHED_PROCESS and CREATE_NEW_CONSOLE are mutual exclusive, First flag creates a process without console and second one creates a process with console

CREATE_UNICODE_ENVIRONMENT: should be set if Unicode is being used.

CREATE_NEW_PROCESS_GROUP : Indicates that the new process is the root of new process group. All processes in the same root group receive the control signal if they share the same console.

lpEnvironment : points to an environment block for the new process. If NULL, the process uses the parent's environment.

lpCurDir Specifies the drive and directory of new process. If NULL parent working directory is used.

lpStartupInfo : specifies the main window appearance and standard device handles for new programs.

lpProcInfo is the pointer to the structure containing handles and id for process and thread.

Topic 89: Why Processes need IDs and Handles

IDs are unique to processes for their entire lifetime. ID is invalidated when a process is destroyed. Although, it may be reused by other newly created processes. Alternately, a process can have many handles with different security attributes.

Some process management functions require handles while others require IDs. Handles are required for general purpose handle based functions. Just like file handles and handles for other resources the process handles need to be closed when not required.

The process obtains environment and other information from CreateProcess() call. Once the process has been created then changing this information may have no effect on the child. For example the parent process may change its working directory but it will not have effect on child unless the child changes its own working directory. Processes are entirely independent.

Topic 90: Specifying the Executable Image and the Command Line

Executable Image

Either lpApplicationName or lpCommandLine specifies the executable image. Usually lpCommandLine is used, in which case lpApplicationName is set to NULL. However if lpApplicationName is specified there are rules governing it.

lpApplicationName

lpApplicationName should not be NULL. It should specify the full name of the application including the path. Or use the current path in which case current drive and directory will be used. Include the extension i.e. .EXE or .BAT in the file name. For long names quotes within the string are not required.

lpCommandLine

lpApplicationName should be NULL. Tokens within string are delimited by spaces. First token is the program image name. If the name does not contain the path of the image then the following search sequence is followed.

- Directory of current process image
- The current Directory
- Windows System directory which can be retrieved from GetSystemDirectory()
- Windows Directory which is retrievable from GetWindowsDirectory(). Directories as specified in Environment variable PATH

Command Line

A process can obtain its command line from the usual argv mechanism. Alternately, in windows it can call GetCommandLine(). It's important to know that command line is not a constant string. A program can change the command line. It's advisable that the program makes changes on a copy of command line.

Topic 91: Inheritable Handles

Mostly, a child process requires access to a resource referenced in parent by a handle. If the handle is inheritable then the child can directly receive a copy of open handles in parent. For example standard input and output handles are shared in this pattern. This inheriting of handles is accomplished in this manner. The bInheritHandles flag in the CreateProcess() call determines whether the child process will inherit copies of parent open handles. Also it is necessary to make individual handles inheritable. Use the SECURITY_ATTRIBUTES structure to specify this. It has a flag bInheritFlag which should be set to TRUE. Also the nLength should be set to sizeof(SECURITY_ATTRIBUTES)

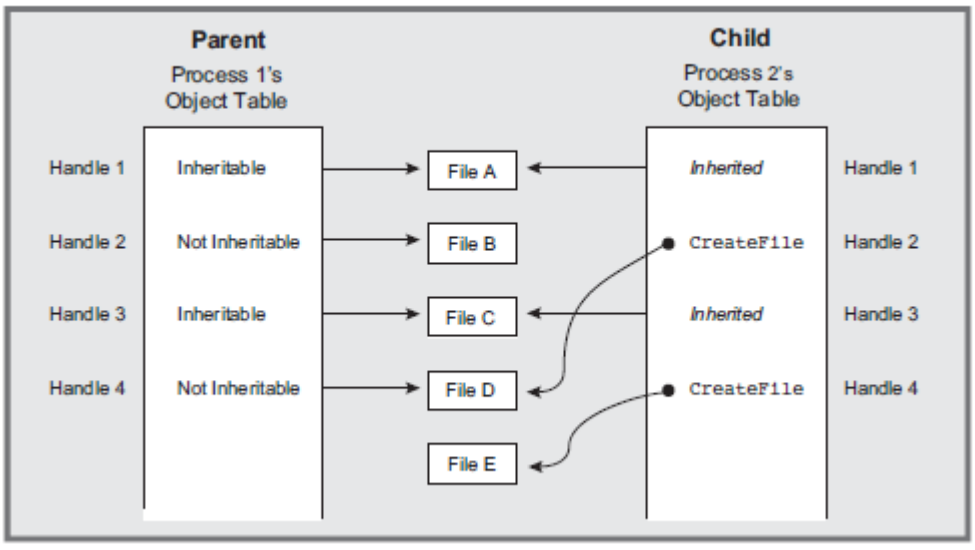
The following code illustrates how this is done programmatically.

```
HANDLE h1, h2, h3;
SECURITY_ATTRIBUTES sa =
    {sizeof(SECURITY_ATTRIBUTES), NULL, TRUE };
...
h1 = CreateFile (..., &sa, ... ); /* Inheritable. */
h2 = CreateFile (..., NULL, ... ); /* Not inheritable. */
h3 = CreateFile (..., &sa, ...);
    /* Inheritable. You can reuse sa. */
```

Topic 92: Passing Inheritable Handles

We have only made the handles inheritable. However, we need to pass the actual values of handles to the child process. Either this is accomplished through InterProcess Communication (IPC) Or it is passed on the child process by setting it up in the STARTUPINFO struct. The latter is a preferred policy as it allows IO redirection and no changes are required in child process. Another approach is to convert the file handles into text and pass them through the command line to the child process.

The handles if are already inheritable then they will readily accessible to the child. Inherited handles are distinct copies. Parent and child might be accessing same file with different file pointer. Each process should close handles.



Content Development

Topic No - 115

More on Thread Management

The functions of thread management that are discussed above are enough to program any useful threading application. However, there are some more functions introduced in the later versions of Windows i.e. Windows XP and Windows 2003 to write a useful and robust program. These functions are described below.

GetProcessIdOfThread()

This function was not available in the earlier version of windows and requires Windows 2003 or later versions. When a thread is specified, this function tells us, to which process a specified thread is linked while returning the id of that process. This function is useful for, mapping thread and process and program that manages or interacts with threads in another process

GetThreadIOPendingFlag()

This function determines whether the thread, specified by its handle, has any outstanding I/O requests. For example, the thread might be blocked for some IO operations. The result is the status at the time that the function is executed; the actual status could change at any time if the target thread completes or initiates an operation.

Suspending and Resuming Threads

In some cases, we might require to pause any running thread or resume any paused thread. For these purposes, Windows maintains a suspend count. Every thread has its separate suspend count. A thread will only run if the suspend count is 0 and if it is not, then the thread is paused and execution of that thread will be stopped until the suspend count becomes 0. One thread can increment or decrement the suspend count of another thread using SuspendThread and ResumeThread. Recall that a thread can be created in the suspended state with a count of 1

DWORD ResumeThread (HANDLE hThread)
DWORD SuspendThread (HANDLE hThread)

Both functions, if successful, return the previous suspend count. 0xFFFFFFFF indicates failure.

Topic No - 116

Waiting for Threads

One thread can wait for another thread to complete or terminate in the same way that threads wait for process termination. Threads and actions should be synchronized so that action can be performed after completing execution. The wait function can be used with the thread handle to check whether a particular thread has completed its execution or not. Window treats thread as an object, and there are two different types of wait function in the windows that can be used for threads as well (Since the thread is also an object), i.e. WaitForSingleObject() or WaitForMultipleObjects(). WaitForSingleObject() is used to wait for a single specified object and WaitForMultipleObjects can be used for more than 1 object, those objects are defined in the form of an array.

For WaitForMultpleObjects, there is a defined limit in windows to wait for execution i.e. MAXIMUM_WAIT_OBJECTS(64), Usually, it is 64 objects. If there are more than 64 objects, we can use multiple calls as well i.e. if there are 100 objects, we can create 2 arrays of 64 and 36 objects and call WaitForMultipleObjects() two times.

The wait function waits for the object, indicated by the handle, to become signaled. In the case of threads, ExitThread() and TerminateThread() set the object to the signaled state, releasing all other threads waiting on the object, including threads that might wait in the future after the thread terminates. Once a thread handle is signaled, it never becomes nonsignaled.

Note that multiple threads can wait on the same object. Similarly, the ExitProcess() function sets the process state and the states of all its threads to signaled.

Topic No – 117

C Library in Threads

Let's assume a scenario, where we are using the window threading function with C library functions concurrently, there might arise a problem of thread safety. For example, strtok() is a C library function, used to extract the token from the specified string, this function uses the global memory space for its internal processing, and if we are using the different copies of that string, they all might use global space. The result achieved from this operation might get compromised and unsatisfactory. In such cases, these types of problems are resolved by using C Library threading functions rather than Windows threading functions. For C Library, Windows C provides a thread-safe library named LIBCMT. This library can be used for thread-relevant functions to program a multithreaded program. So far, we were using the CreateThread and ExitThread functions to create and exit the threads, this library provides us with these equivalent functions i.e. _beginthreadex() and endthreadex() respectively. These C library functions are quite simpler but are not diverse as compared to Windows threading functions. i.e. _beingthreadex() is a simpler function but it does not allow users to specify the security attributes. _endthreadex() does not allow return values and does not pass information regarding the status of the thread. If we are using the windows program and using this C Libray thread function the return value must be type cast to HANDLE to process it further, since the original return type of _beginthreadex() is not HANDLE.

Topic No - 118

Multithreaded Pattern Searching

In this example of pattern searching with multithreading, the program is managing concurrent I/O to multiple files, and the main thread, or any other thread, can perform additional processing before waiting for I/O completion. In this way, we can manage several files in a very simple and efficient way. In the previous examples of pattern searching, we used the multitasking approach, but here we will use the multithreading technique, that enables us to write an optimal program.

In Synchronous Input-Output, suppose an example of the keyboard, when the user presses any keyboard button the execution stops until the button is released, but in asynchronous input-output, a Sound card plays the song at the same time other operations are also performed.

When a read operation is performed, this can be performed concurrently, a file can be read by several processes at the same time or we can also read several files at the same time. But the problem arises when several processes attempt to write a single file. For now, we will be limited to read operation only. This program can be provided several files in which a specific pattern is to be searched. For every file, a separate thread will be created which will search the pattern in that file. Once the pattern is found in the file, it will be reported in a temporary file

```
/* grepMT. */
/* Parallel grep-- multiple thread version. */

#include "Everything.h"
typedef struct { /* grep thread's data structure. */
        int argc;
        TCHAR targv[4][MAX_PATH];
} GREP_THREAD_ARG;

typedef GREP_THREAD_ARG * PGR_ARGS;
static DWORD WINAPI ThGrep (PGR_ARGS pArgs);

int _tmain(int argc, LPTSTR argv[])
{
        GREP_THREADED_ARG *gArg;
        HANDLE *tHandle;
        DWORD threadIndex, exitCode;
        TCHAR commandLine[MAX_COMMAND_LINE);
        int iThrd, threadCount;
        STARTUPINFO startUp;
        PROCESS_INFORMATION processInfo;

        GetStartupInfo(&startUp);

        /* Boss Thread: create seperate "grep" thread for each file. */
```

```
tHandle = malloc((argc-2) * sizeof(HANDLE));
gArg = malloc((argc-2) * sizeof (GREP_THREAD_ARG));

for(iThrd=0, iThrd<argc-2; iThrd++)
{
_tcscpy (gArg[iThrd].targv[1], argv[1]);  /* Pattern */
_tcscpy (gArg[iThrd].targv[2], argv[iThrd + 2]);
GetTempFileName /* Temp file name. */
        (".", "Gre", 0, gArg[iThrd].targv[3];
gArg[iThrd].argc = 4;

/* Create a worker thread to execute the command line. */
tHandle[iThrd] = (HANDLE)_beginthreadex (
        NULL, 0, ThGrep, &gArg[iThrd], 0, NULL);
}

/* Redirect std Output for file listing process. */
startUp.dwFlags = STARTF_USESTDHANDLES;
startUp.hStdOutput = GetStdHandle (STD_OUTPUT_HANDLE);

/* Worker threads ar eall running . Wait for them to complete. */
threadCount = argc - 2;
while (threadCount>0) {
        threadIndex =  WaitForMultipleObjects (
                threadCount, tHandle, FALSE, INFINITE);
        iThrd = (int) threadIndex - int (WAIT_OBJECT_0);
        GetExitCodeThread ( tHandle[iThrd], &exitCode);
        CloseHandle ( tHandle[iThrd]);
        if ( exitCode == 0) {       /* Pattern Found */
                if ( argc > 3) {
                        /* Print file name if more than one . */
                        _tprintf ( _T("\n**Search results - file : %s\n"),
                                gArg[iThrd].targv[2];
                        fflush(stdout);
                }

        /* Use the "cat" program to list the result files. */
        _stprintf ( commandLine, _T("%s%s"), _T("cat "),
                gArg[iThrd].targv[3]);
        CreateProcess(NULL, commandLine, NULL, NULL,
                TRUE, 0, NULL, NULL, &startUP, &processInfo);
        waitForSingleObject ( processInfo.hProcess, INFINITE);
        CloseHandle (processInfo.hProcess);
        CloseHandle (processInfo.hThread);
        }

DeleteFile ( gArg[iThrd].targv[3]);
```

```
        /* Adjust thread and file name arrays. */
        tHandle[iThrd] = tHandle[threadCount - 1];
        _tcscpy(gArg[iThrd].targv[3], gArg[threadCount -1 ].targv[3]);
        _tcscpy(gArg[iThrd].targv[2], gArg[threadCount -1 ].targv[2]);
        threadCount--;
        }
}

/* The form of grep thread function code is :
static  DWORD WINAPI ThGrep (PGR_ARGS pArgs)
{
        ....
} */
```

- Structure of thread argument is created with argument count (argc) and thread argument value (targv) which will be passed to thread. A Thread prototype is also created named as ThGrep which will be used to search the specific patterns in the file.
- STARTUPINFO and PROCESS_INFORMATION structures are created for startup and for creating process respectively
- stratUp information is placed in GetStratupInfo function
- Files are specified using the command line, in which patterns are to be searched, and for every file inputted, a separate thread will run.
- Loop will run till argc-2 times, in argc, the first two parameters will be process name and pattern and 3rd parameter is inputted file names, though, this loop will run, till the number of files inputted. In this loop for every file, the name of the file is copied and its temporary file is created
- Further, arguments that are to be passed to the thread are also prepared, 1st argument will store the pattern which is to be searched, 2nd argument will store the name of the file in which the pattern is to be searched and 4th argument will store the count i.e. how many arguments are stored.
- Thread is created using _beginthreadex, whose name is ThGrep and is passed all the arguments that were created (gArg). With this, the handle of every thread will be stored in the form of an array (tHandle)
- Standard output files, standard error files, and flags are set for startup information
- Total number of threads are stored in ThreadCount
- Another loop is run, in which the Wait function is called to wait for multiple objects specified with a number of threads (threadCount) and handles of all threads (tHandle array). When the wait function is completed, an exitcode will be generated which will tell us why the wait function is stopped, and further, for garbage collection, the handle of the thread is also closed with the CloseHandle function.
- When the wait function was in execution, a process is created with help of the CreateProcess function which has been provided, startup information(startUp), and process information (processInfo).
- Another wait function is also called but this time WaitForSingleObject is called which has been provided the handle of the process named hProcess.
- Once the execution of the process is completed, the handle of the process as well as the

thread are closed.

Topic No – 119

Boss worker and other thread models

In the example of multithreaded pattern searching, there was one main thread, which was running other threads. Each file was assigned a separate thread, which was finding the pattern in that file. This model is more like a boss worker model. The boss worker model is one in which there is one boss and many workers. The Boss assigns work to workers and each worker report result back to the boss. There are many other models, which are used to write an efficient and more understandable multithreaded program depending on the scenarios.

The work crew model is one in which the workers cooperate on a single task, each performing a small piece. They might even divide up the work themselves without direction from the boss. Multithreaded programs can employ nearly every management arrangement that humans use to manage concurrent tasks.

The Client-Server model is mostly used worldwide, in which a client requests the server and the server runs a thread for that client. For every client, a separate thread is run at the server end. In this way, the work is done concurrently rather than sequentially. Another major model is the pipeline model, where work moves from one thread to the next

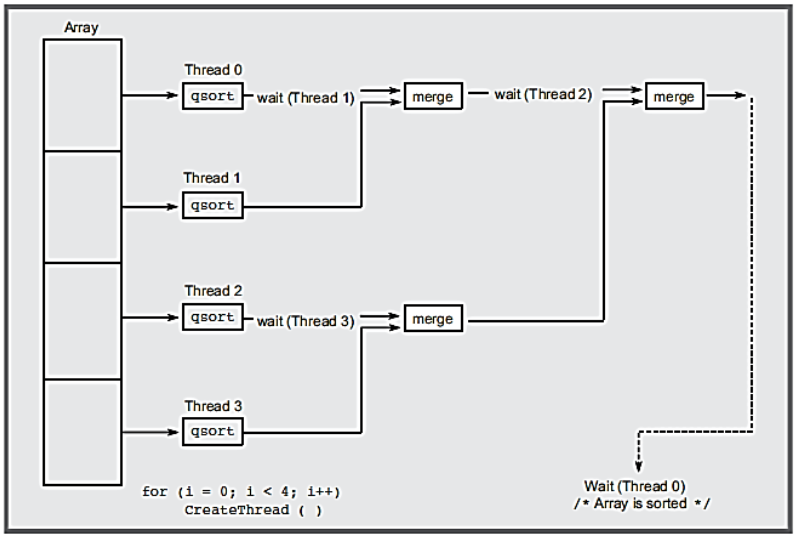
There are many advantages to using these models when designing a multithreaded program, including the following.

- Most models can be used which makes things simpler and expedites programming and debugging efforts.
- Models help you obtain the best performance and avoid common mistakes
- Models naturally correspond to the structure of programming language constructs.
- Maintenance is simplified.
- Troubleshooting is simplified when they are analyzed in terms of a specific model.
- Synchronization and coordination are simplified using well-defined models.

Topic No – 120

MergeSort: Exploiting Multiple Processors

The window is a multiprocessing system i.e. more than 1 process can run simultaneously. This example shows, how multithreading can be used to achieve the optimal performance gain in a multiprocessing system, each process runs a different thread to utilize the optimal resources. The main idea behind this is to subdivide the process into similar tasks so that a separate thread is run for each subtask i.e. A big array is divided into smaller parts, each part is sorted separately with different threads, and all the parts are merged. This will allow parallelism and better performance gain. The strategy implemented in this example is the worker crew model, work is divided into different workers, and all the work is merged in the end. This strategy could also be used, by using multiprocessing instead of multithreading, but the result might not be as efficient as it is with multithreading, because switching overhead is low in multithreading but high in multiprocessing. In the example of MergeSort, each subarray is sorted with qsort() and merged as in the mergesort algorithm. The program code will run most accurately if a number of records are divisible by a number of threads and the number of threads is in the power of two. If the number of processes is equal to the number of threads, this would be the most optimal situation otherwise less optimal. If a list is subdivided into 4 sub-lists and 4 threads are created for these sublists, they all must be created at a suspended state and should only be resumed when all the threads are created. If one thread is completed and the other, which is to be merged, is not created or does not exist, this will occur in a race condition. To avoid the race condition, all the threads should be created with a suspended state and resumed to run all concurrently. The following diagrams explain it further



A Large array is divided into smaller 4 subarrays. For each subarray, a separate thread is created i.e. thread 0, thread 1, thread 2, and thread 3. When thread 0 is sorted it will wait for thread 1 to be sorted, once sorted, they both will be merged. The same happens for thread 2 and thread 3, they are merged when sorted. These 2 merged subarrays are then sorted and merged to form a large sorted array.

Topic No – 121

MergeSort: Exploiting Multiple Processors

```
/* Chapter 7 SortMT. Work crew model
        File sorting with multiple threads and merge sort.
        sortMT [options] nt file. Work crew model.  */
/*  This program is based on sortHP.
        It allocates a block for the sort file, reads the file,
        then creates an "nt" threads (if 0, use the number of
        processors to so sort a piece of the file. It then
        merges the results in pairs. */
/* LIMITATIONS:
        1.      The number of threads must be a power of 2
        2.      The number of 64-byte records must be a multiple
                of the number of threads.
        An exercise asks you to remove these limitations. */
#include "Everything.h"
/* Definitions of the record structure in the sort file. */
#define DATALEN 56  /* Correct length for presdnts.txt and monarchs.txt. */
#define KEYLEN 8
typedef struct _RECORD {
        TCHAR key[KEYLEN];
        TCHAR data[DATALEN];
} RECORD;
#define RECSIZE sizeof (RECORD)
typedef RECORD * LPRECORD;
typedef struct _THREADARG {       /* Thread argument */
        DWORD iTh;          /* Thread number: 0, 1, 3, ... */
        LPRECORD lowRecord;       /* Low Record */
        LPRECORD highRecord;      /* High record */
} THREADARG, *PTHREADARG;
static DWORD WINAPI SortThread (PTHREADARG pThArg);
static int KeyCompare (LPCTSTR, LPCTSTR);
static DWORD nRec; /* Total number of records to be sorted. */
static HANDLE * pThreadHandle;
int _tmain (int argc, LPTSTR argv[])
{

        /* The file is the first argument. Sorting is done in place. */
        /* Sorting is done in memory heaps. */
        HANDLE hFile, mHandle;
        LPRECORD pRecords = NULL;
        DWORD lowRecordNum, nRecTh,numFiles, iTh;
        LARGE_INTEGER fileSize;
        BOOL noPrint;
        int iFF, iNP;
```

```
	PTHREADARG threadArg;
	LPTSTR stringEnd;

	iNP = Options (argc, argv, _T("n"), &noPrint, NULL);
	iFF = iNP + 1;
	numFiles = _ttoi(argv[iNP]);
	if (argc <= iFF)
		ReportError (_T ("Usage: sortMT [options] nTh files."), 1, FALSE);
	/* Open the file and map it */
	hFile = CreateFile (argv[iFF], GENERIC_READ | GENERIC_WRITE,
			0, NULL, OPEN_EXISTING, 0, NULL);
	if (hFile == INVALID_HANDLE_VALUE)
		ReportError (_T ("Failure to open input file."), 2, TRUE);
	// For technical reasons, we need to add bytes to the end.
	/* SetFilePointer is convenient as it's a short addition from the file end */
	if (!SetFilePointer(hFile, 2, 0, FILE_END) || !SetEndOfFile(hFile))
		ReportError (_T ("Failure position extend input file."), 3, TRUE);

	mHandle = CreateFileMapping(hFile, NULL, PAGE_READWRITE, 0, 0, NULL);
	if (NULL == mHandle)
		ReportError (_T ("Failure to create mapping handle on input file."), 4, TRUE);

	/* Get the file size. */
	if (!GetFileSizeEx (hFile, &fileSize))
		ReportError (_T ("Error getting file size."), 5, TRUE);
	nRec = (DWORD)fileSize.QuadPart / RECSIZE;     /* Total number of records. Note
assumed limit */
	nRecTh = nRec / numFiles;    /* Records per thread. */
	threadArg = malloc (numFiles * sizeof (THREADARG));   /* Array of thread args. */
	pThreadHandle = malloc (numFiles * sizeof (HANDLE));

	/* Map the entire file */
	pRecords = MapViewOfFile(mHandle, FILE_MAP_ALL_ACCESS, 0, 0, 0);

	if (NULL == pRecords)
		ReportError (_T ("Failure to map input file."), 6, TRUE);
	CloseHandle (mHandle);
	/* Create the sorting threads. */
	lowRecordNum = 0;
	for (iTh = 0; iTh < numFiles; iTh++) {
		threadArg[iTh].iTh = iTh;
		threadArg[iTh].lowRecord = pRecords + lowRecordNum;
		threadArg[iTh].highRecord = pRecords + (lowRecordNum + nRecTh);
		lowRecordNum += nRecTh;
		pThreadHandle[iTh] = (HANDLE)_beginthreadex (
			NULL, 0, SortThread, &threadArg[iTh],
```

```
CREATE_SUSPENDED, NULL);
        }
        /* Resume all the initially suspened threads. */
        for (iTh = 0; iTh < numFiles; iTh++)
                ResumeThread (pThreadHandle[iTh]);
        /* Wait for the sort-merge threads to complete. */
        WaitForSingleObject (pThreadHandle[0], INFINITE);
        for (iTh = 0; iTh < numFiles; iTh++)
                CloseHandle (pThreadHandle[iTh]);

        /*  Print out the entire sorted file. Treat it as one single string. */
        stringEnd = (LPTSTR) pRecords + nRec*RECSIZE;
        *stringEnd =_T('\0');
        if (!noPrint) {
           _tprintf (_T("%s"), (LPCTSTR) pRecords);
        }
        UnmapViewOfFile(pRecords);
        // Restore the file length
        /* SetFilePointer is convenient as it's a short addition from the file end */
        if (!SetFilePointer(hFile, -2, 0, FILE_END) || !SetEndOfFile(hFile))

        ReportError (_T("Failure restore input file lenght."), 7, TRUE);
        CloseHandle(hFile);
        free (threadArg); free (pThreadHandle);
        return 0;
} /* End of _tmain. */
static VOID MergeArrays (LPRECORD, DWORD);
DWORD WINAPI SortThread (PTHREADARG pThArg)
{

        ReportError (_T("Failure restore input file lenght."), 7, TRUE);
        CloseHandle(hFile);
        free (threadArg); free (pThreadHandle);
        return 0;
} /* End of _tmain. */
static VOID MergeArrays (LPRECORD, DWORD);
DWORD WINAPI SortThread (PTHREADARG pThArg)
{

        DWORD groupSize = 2, myNumber, twoToI = 1;
                        /* twoToI = 2^i, where i is the merge step number. */
        DWORD_PTR numbersInGroup;
        LPRECORD first;
        myNumber = pThArg->iTh;
        first = pThArg->lowRecord;
        numbersInGroup = (DWORD)(pThArg->highRecord - first);
```

```
        /* Sort this portion of the array. */
        qsort (first, numbersInGroup, RECSIZE, KeyCompare);

        /* Either exit the thread or wait for the adjoining thread. */
        while ((myNumber % groupSize) == 0 && numbersInGroup < nRec) {
                                /* Merge with the adjacent sorted array. */
                WaitForSingleObject (pThreadHandle[myNumber + twoToI], INFINITE);
                MergeArrays (first, numbersInGroup);
                numbersInGroup *= 2;
                groupSize *= 2;
                twoToI *=2;
        }
        return 0;
}
static VOID MergeArrays (LPRECORD p1, DWORD nRecs)
{

        /* Merge two adjacent arrays, each with nRecs records. p1 identifies the first */
        DWORD iRec = 0, i1 = 0, i2 = 0;
        LPRECORD pDest, p1Hold, pDestHold, p2 = p1 + nRecs;
        pDest = pDestHold = malloc (2 * nRecs * RECSIZE);
        p1Hold = p1;
        while (i1 < nRecs && i2 < nRecs) {
                if (KeyCompare ((LPCTSTR)p1, (LPCTSTR)p2) <= 0) {
                        memcpy (pDest, p1, RECSIZE);
                        i1++; p1++; pDest++;
                }

                else {
                        memcpy (pDest, p2, RECSIZE);
                        i2++; p2++; pDest++;
                }
        }
        if (i1 >= nRecs)
                memcpy (pDest, p2, RECSIZE * (nRecs - i2));
        else    memcpy (pDest, p1, RECSIZE * (nRecs - i1));
        memcpy (p1Hold, pDestHold, 2 * nRecs * RECSIZE);
        free (pDestHold);
        return;
}
int KeyCompare (LPCTSTR pRec1, LPCTSTR pRec2)
{
        DWORD i;
        TCHAR b1, b2;
        LPRECORD p1, p2;
        int Result = 0;
```

```
        p1 = (LPRECORD)pRec1;
        p2 = (LPRECORD)pRec2;
        for (i = 0; i < KEYLEN && Result == 0; i++) {
                b1 = p1->key[i];
                b2 = p2->key[i];
                if (b1 < b2) Result = -1;
                if (b1 > b2) Result = +1;
        }
        return  Result;
}
```

- A thread structure is defined which is passed to every thread as an argument, this structure has the ‘iTh’ variable which specifies the thread number, ‘lowRecord’ and ‘highRecord’ which are used to define the starting and ending index of the sub list.
- ‘SortThread’ is the thread name that is to be created
- ‘nRec’ is the total number of threads created
- ‘pThreadHandle' is the handle to the thread
- The file, which has records in it, is opened to read and write. This file is mapped (‘mHandle’) to access that file as memory i.e. in the form of an array.
- Thread argument, which is to be passed as an argument to create a thread, is allocated the memory, that has all three fields that are to be passed
- ‘nRec’ is the total number of records
- ‘nRec/numFiles’ defines the number of records per file
- ‘pRecord’ is the address of the mapped file
- Different threads are created and run to divided the file into chunks and each thread sorts the file. To divide the file, a loop is run till ‘numFiles’ times, and specifies the thread numbers (‘ith’), starting index of a chunk (‘lowRecord’), and ending index of a chunk (‘highRecord’)
- The thread is created with ‘beginThread’ functions which are passed ‘SortThread’ and ‘ThreadArgument’ in a suspended state.
- When all the threads are created, they are resumed with help of a loop.
- For all created threads, we wait till the execution completes. Once the execution is completed handle is closed and the file is unmapped.
- When a thread is run, it is passed ‘QSort’ to sort the records, and then it waits for its adjacent thread to be sorted and completed. Once they are completed, the results are merged. Every thread waits for its adjacent thread to be completed and merges after completion

Performance

- Multithreading gives the best results when the number of processors is the same as the number of threads.
- Additional threads beyond processor count slows down the program
- Performance degrades if there is one processor and memory is low and the array is very large because most of the time threads will be contending for physical memory. But the problem alleviates when the memory is sufficient.

Topic No – 122

Introduction to Parallelism

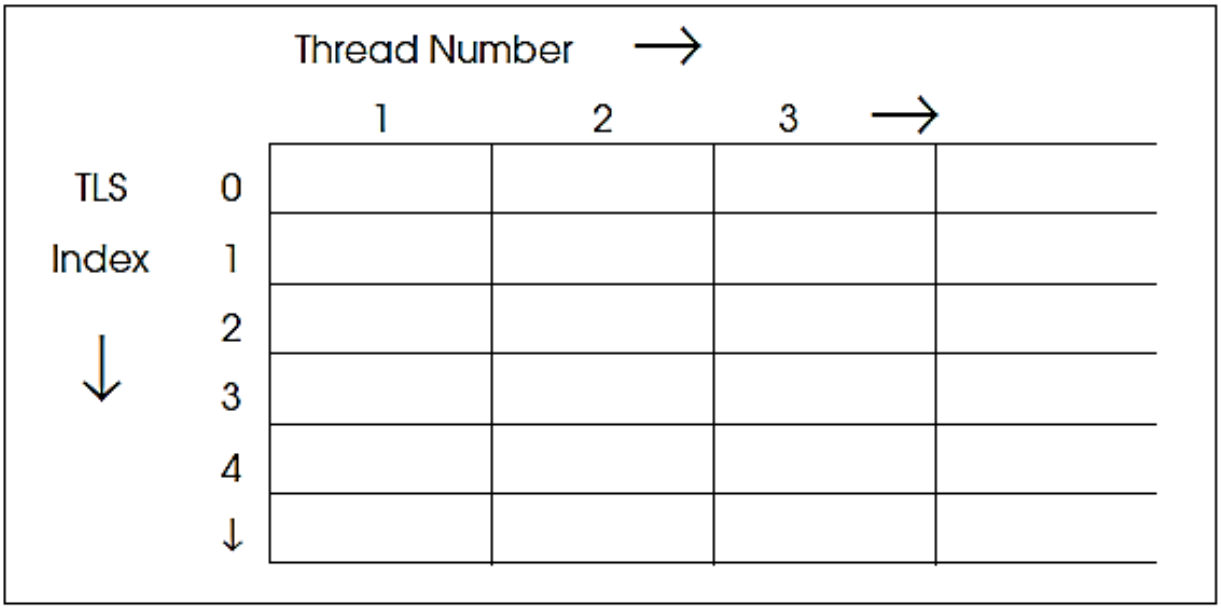
Multiprocessing and multithreading, both are responsible for multiple flows of execution, but the multithreading is optimal. Windows is not only multithreading or multiprocessing it also supports multiprocessors as well. If we have a system with multiprocessors, we should learn to program to use the potential of multiprocessors because the processor's speed has reached its bottleneck i.e. after a certain limit, its speed cannot be enhanced. Parallelization is the key to future performance improvement since we can no longer depend on increased CPU clock rates and since multicore and multiprocessor systems are increasingly common. Previously, various programs have been discussed that unleash the power of parallelism. The properties that enabled parallelism include the following:

- Major task is divided into subtasks and many worker threads were run. Subtasks were divided into worker threads that perform their work. These worker subtasks run independently, without any interaction between them.
- As subtasks are complete, a master program can merge the results of divided subtasks into a single result.
- The programs do not require mutual exclusion of any sort. Only the master worker is synchronized with each worker and waits for them to complete.
- Every worker will work as a separate thread on a separate processor. it is the most optimal situation
- Program performance scales automatically, up to some limit, as you run on systems with more processors; the programs themselves do not, in general, determine the processor count on the host computer. Instead, the Windows kernel assigns worker subtasks to available processors.
- Output remains undisturbed even if the program is serialized.
- If you "serialize" the program the results should get precisely the same as the parallel program. The serialized program is, moreover, much easier to debug.
- The maximum performance improvement is limited by the program's "parallelism," thread management overhead, and computations that cannot be parallelized.

Topic No – 123

Thread Local Storage

A thread is an execution unit. In a multithreading program, one procedure can have several threads. Every thread needs data, that it doesn't want to share with other threads and which is unique i.e. it varies from thread to thread. One technique is to have the creating thread call CreateThread (or beginThreadex) with lpvThreadParm pointing to a data structure that is unique for each thread. The thread can then allocate additional data structures and access them through lpvThreadParm. Windows also provides Thread Local Storage (TLS), which gives each thread its array of pointers. The following figure shows this TLS arrangement.



A function can have many threads i.e. thread 1, thread 2, etc. Every column in the TLS arrangement corresponds to thread numbers and every thread needs variables i.e. TLS index 0,1,2,3 etc. Initially, no TLS indexes (rows) are allocated, but new rows can be allocated and deallocated at any time. Once the row is allocated, it will be allocated to all rows. The primary thread would be a logical choice for TLS space management, however, every thread can access TLS.

- **DWORD TlsAlloc(VOID):** This API is used to allocate the index and it returns the TLS index in the form of the double word. Otherwise returns -1 in case of failure.
- **BOOL TlsFree(DWORD dwIndex)**: Frees the specified index.
- **LPVOID TlsGetValue (DWORD dwTlsIndex) and BOOL TlsSetValue (DWORD dwTlsIndex, LPVOID lpTlsValue):** Provided valid indexes are used. The programmer can access TLS space using these simple GET/SET APIs

Some Cautions

- TLS provides a convenient mechanism for accessing memory that is global within a thread but inaccessible to other threads.

- Global storage of a program is accessible by all threads
- TLS provides a convenient mechanism for accessing memory that is global within a thread but inaccessible to other threads.
- Global storage of a program is accessible by all threads

Topic No – 124

Processes and Thread priorities

In a multitasking or multithreading system, there are a number of processes running, each of which competes for resources like processor, memory, etc. The operating system is responsible for managing the resources. The Windows kernel always runs the highest-priority thread that is ready for execution. A thread is not ready if it is waiting, suspended, or blocked for some reason. Since the threads are dependents on the processes and they receive priority relative to their process priority classes. Process priority classes are set initially when they are created using CreateProcess, and each has a base priority, with values including

- **IDLE_PRIORTY_CLASS** for threads that will run only when the system is idle. This is the lowest priority process.
- **NORMAL_PRIORTY_CLASS** indicating no special scheduling requirements.
- **HIGH_PRIORTY_CLASS** indicating time-critical tasks that should be executed immediately.
- **REALTIME_PRIORTY_CLASS**, the highest possible priority

The priority class of a process can be set and got using **BOOL SetPriorityClass(HANDLE hProcess, DWORD dwPriority) and DWORD GetPriorityClass(HANDLE hProcess)** respectively.

There are some enhances variants of priority levels i.e. **ABOVE_NORMAL_PRIORTY_CLASS (**which is below HIGH_PRIORTY_CLASS) and **BELOW_NORMAL_PRIORTY_CLASS (**which is above IDLE_PRIOTY_CLASS).

PROCESS_MODE_BACKGROUND_BEGIN, which lowers the priority of the process and its threads for background work without affecting the responsiveness of foreground processes and threads. **PROCESS_MODE_BACKGROUND_END** restores the process priority to the value before it was set with PROCESS_MODE_BACKGROUND_BEGIN.

Thread priorities are either absolute or are set relative to the process base priority. At thread creation time, the priority is set to that of the process. The relative thread priorities are in a range of ±2 “points” from the process’s base. The symbolic names of the resulting common thread priorities, starting with the five relative priorities, are:

- THREAD_PRIORTY_LOWEST
- THREAD_PRIORTY_BELOW_NORMAL
- THREAD_PRIORTY_NORMAL
- THREAD_PRIORTY_ABOVE_NORMAL
- THREAD_PRIORTY_HIGHEST
- THREAD_PRIORTY_TIME_CRITICAL is 15, or 31 if the process class is REAL_TIME_PRIORTY_CLASS
- THREAD_PRIORTY_IDLE is 1, or 16 for REAL_TIME_PRIORTY_CLASSES processes

THREAD_MODE_BACKGROUND_BEGIN and **THREAD_MODE_BACKGROUND_END** are similar to **PROCESS_MODE_BACKGROUND_BEGIN** and **PROCESS_MODE_BACKGROUND_END**.

Thread priorities can be set within the range of +-2 of the corresponding process priority using these **BOOL SetThreadPriority(HANDLE hThread, int nPriority)** AND **Int GetThreadPriority(HANDLE hThread).**

Thread priorities are dynamic. They change with the priority of process or windows may also boost thread priority as per need. This feature can be enabled disabled using **SetThreadPriorityBoost()**

Topic No – 125

Thread States

The following figure shows how the executive manages threads and shows the possible thread states. This figure also shows the effect of program actions. Such state diagrams are common to all multitasking OSs and help clarify how a thread is scheduled for execution and how a thread moves from one state to another.

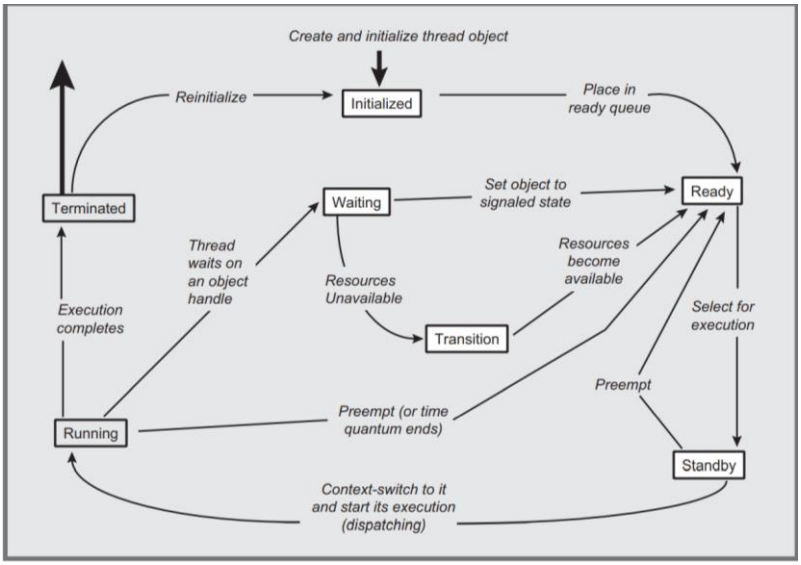


Figure 1 Thread States and Transitions

(From Inside Windows NT, by Helen Custer. Copyright © 1993, Microsoft Press. Reproduced by permission of Micro-soft Press. All rights reserved.)

- A thread is in the running state when it is running on a processor. More than one thread can be in the running state on a multiprocessor computer
- The executive places a running thread in the wait state when the thread performs a wait on a non-signaled handle, such as a thread or process handle. I/O operations will also wait for the completion of a disk or other data transfer, and numerous other functions can cause waiting. It is common to say that a thread is blocked, or sleeping, when in the wait state
- A thread is ready if it could be running. The executive's scheduler could put it in the running state at any time. The scheduler will run the highest-priority ready thread when a processor becomes available, and it will run the one that has been in the ready state for the longest time if several threads have the same high priority. The thread moves through the standby state before entering the ready state.
- The executive will move a running thread to the ready state if the thread's time slice expires without the thread waiting. Executing will also move a thread from the running state to the ready state.
- The executive will place awaiting thread in the ready state as soon as the appropriate handles are signaled, although the thread goes through an intermediate transition state. It is common to say that the thread wakes up.

- A thread, regardless of its state, can be suspended, and a ready thread will not be run if it is suspended. If a running thread is suspended, either by itself or by a thread on a different processor, it is placed in the ready state.
- A thread is in the terminated state after it terminates and remains there as long as there are any open handles on the thread. This arrangement allows other threads to interrogate the thread's state and exit code.
- Normally, the scheduler will place a ready thread on any available processor. The programmer can specify a thread's processor, which will limit the processors that can run that specific thread. In this way, the programmer can allocate processors to threads and prevent other threads from using these processors, helping to assure responsiveness for some threads. The appropriate functions are SetProcessAffinityMask and GetProcessAffinityMas. SetThreadIdealProcessor can specify a preferred processor that the scheduler will use whenever possible; this is less restrictive than assigning a thread to a single processor with the affinity mask.

Topic No – 126

Mistakes while using Threads

There are several factors to keep in mind as you develop threaded programs; lack of attention to a few basic principles can result in serious defects, and it is best to avoid the problems in the first place than try to find them during testing or debugging.

- The essential factor is that the threads execute asynchronously.
- There is no sequencing unless you create it explicitly.
- This asynchronous behavior is what makes threads so useful, but without proper care, serious difficulties can occur.

Here are a few guidelines. There may be a few inadvertent violations, however, which illustrates the multithreaded programming challenges.

- Make no assumptions about the order in which the parent and child threads execute.
- It is possible for a child thread to run to completion before the parent, or, conversely, the child thread may not run at all for a considerable period.
- On a multiprocessor computer, the parent and one or more children may even run concurrently.
- Make sure all the initializations required by a child thread have been performed before calling CreateThread()
- In case a thread has been run but initialization is required then use some technique like thread suspension until data is initialized.
- Failure by the parent to initialize data required by the child is a common cause of “race conditions” wherein the parent “races” the child to initialize data before the child needs it.
- Any thread, at any time, can be preempted, and any thread, at any time, may resume execution
- Do not confuse synchronization and priority. These both are different concepts. Threads are defined as their priorities when created, However Threads are synchronized in such a way that one thread completes its specific purpose, and only then another thread may run its process.
- Even more so than with single-threaded programs, testing is necessary, but not sufficient, to ensure program correctness. It is common for a multithreaded program to pass extensive tests despite code defects. There is no substitute for careful design, implementation, and code inspection.
- Threaded program behavior varies widely with processor speed, number of processors, OS version, and more. Testing on a variety of systems can isolate numerous defects, but the preceding precaution still applies.
- The default stack size for a thread is 1MB. Make sure the size is sufficient as per thread needs.
- Threads should be used only as appropriate. Thus, if there are activities that are naturally concurrent, each such activity can be represented by a thread. If, on the other hand, the activities are naturally sequential, threads only add complexity and performance overhead.

- If you use a large number of threads, be careful, as the numerous stacks will consume virtual memory space and thread context switching may become expensive. In other cases, it could mean more threads than the number of processors.
- Fortunately, correct programs are frequently the simplest and have the most elegant designs. Avoid complexity wherever possible.

Topic No – 127

Timed Waits

In threading, there are some functions that can be used to wait for threads. The Sleep function allows a thread to give up the processor and move from the running to the wait state for a specified period of time. After the completion of the specified time, the thread will move to the ready state and will wait to move on running state. A thread can perform a task periodically by sleeping after carrying out the task.

VOID Sleep (DWord dwMilliseconds)

The time period is specified in milliseconds and can even be INFINITE, in which case the thread will never resume. A 0 value will cause the thread to relinquish the remainder of the time slice; the kernel moves the thread from the running state to the ready state.

The function SwitchToThread() provides another way for a thread to yield its processor to another ready thread if there is one that is ready to run.

Topic No – 128

Fibers

A fiber, as the name implies, is a piece of a thread. More precisely, fiber is a unit of execution that can be scheduled by the application rather than by the operating system. An application can create numerous fibers, and the fibers themselves determine which fiber will execute next. The fibers have independent stacks but otherwise run entirely in the context of the thread on which they are scheduled, having access, for example, to the thread's TLS and any mutexes owned by the thread. Furthermore, fiber management occurs entirely in user space outside the kernel. Fibers can be thought of as lightweight threads, although there are numerous differences. A fiber can execute on any thread, but never on two at one time. Fiber that is meant to run on different threads at different instances should not access thread-specific data from TLS.

Fiber Uses

- Fibers make portability easy.
- Fibers need not wait/block on file locks, mutexes, and pipes. They can pool resources and in case resources are not available they can switch to another fiber.
- Fiber exhibits flexibility. They exist as part of threads but they are not bound to any specific thread. They can run on any thread but only one at a time.
- Fibers are not pre-emptively scheduled. Windows OS is unaware of fibers. They are controlled and managed by fiber DLL.
- Fibers can be used as co-routines. Using this an application can switch between related tasks. Whereas, in the case of threads applications has no control over which thread executes next.
- Major software platform offers the use of fibers and claims its performance advantages.
-

Topic No – 129

Fiber APIs

A set of different API functions are provided that can help create and manage fibers. These functions are

A thread must enable fiber operation by calling

ConvertThreadToFiber() or

ConvertThreadToFiberEx()

After calling this API the thread will now contain a fiber. It will provide a pointer to the fiber data more or less like thread data. This can be used accordingly.

Subsequently, new fibers can be created in this thread using

CreateFiber()

Each new fiber has a start address, stack size, and a parameter. Each fiber is identified by address and not a handle.

Individual fiber can obtain their data by calling

GetFiberData()

Similarly, a fiber can obtain its identity by calling

GetCurrentFiber()

A running fiber gives control to another fiber using

SwitchToFiber()

It uses the address of the other fiber. The context of the current fiber is saved and the context of the other fiber is restored. Fibers must explicitly indicate the next fiber that is to run in the application.

A running fiber gives control to another fiber using

SwitchToFiber()

It uses the address of the other fiber. The context of the current fiber is saved and the context of the other fiber is restored. Fibers must explicitly indicate the next fiber that is to run in the application.

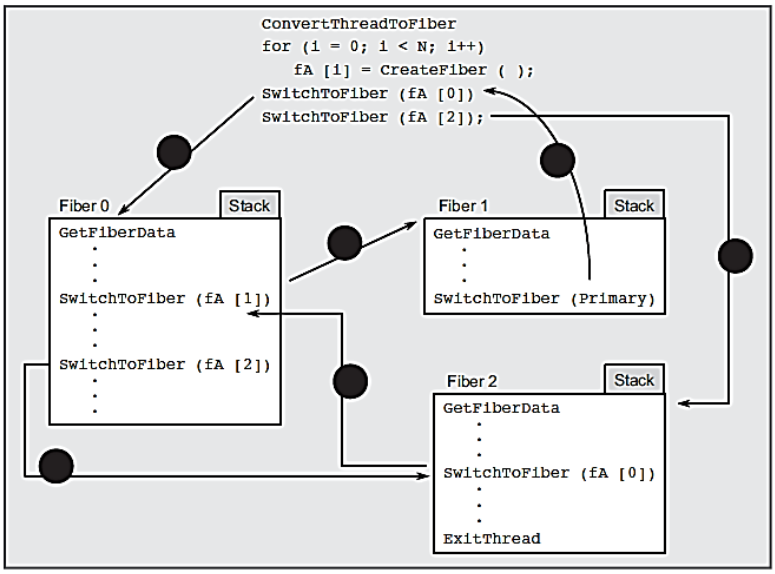
An existing fiber can be deleted using

DeleteFiber()

Topic No – 130

Using Fibers

Previously we discussed several APIs used to manage the fibers, here we will develop a scheme with these APIs to use the fibers. Fiber enables us to control the switching of threads. This scheme firstly converts a thread to fiber and then uses it as the primary fiber. The primary fiber creates other fibers and switching among these fibers is managed by the application.



In the center top, a primary thread is created in which ConvertThreadToFiber is used to convert into fiber and then with the help of a loop number of fibers are created. To convert the execution to a certain fiber SwitchToFiber is used. Primary fiber is switched to Fiber 0 which gets fiber data and then switches to Fiber 1 which also performs the same work i.e. gets data and this fiber switches to primary fiber. The primary fiber starts execution from the point where it was switched before i.e. now it switches to fiber 2 which gets data and switches to fiber 0 and then back to fiber 2 and at the end Thread is closed.

There are two policies

- Master-Slave Scheduling: One fiber decides which fiber to run. Each fiber transfers back the execution to the primary fiber. (Fiber 1)
- Peer to Peer Scheduling: A fiber determine which fiber should be next to execute based on some policy (Fiber 0 and 2)

Topic No – 131

Need for Thread Synchronization

Threads enable concurrent processing and parallelism in multiple processors. However, there are some pros and cons because of this concurrent processing. When many threads may run concurrently. They may need to synchronize for tasks in numerous instances. i.e. in the Boss-worker model, there are one boss thread and many workers threads, the Boss thread waits for all workers to complete execution and compiles all the data. In case if boss executes before the worker completes its execution, this may affect the output we are required to ensure that the Boss will not access the worker's memory unless the worker completes it. Alternately, the workers do not start working unless the boss has created all the workers so that a worker may not try to access the data of another worker which has not been created as yet. When multiple threads are using share data they may require coordination i.e. when a thread is using the data other threads should wait for it. Also, programmers need to ensure that two or more threads are not modifying the same data item simultaneously. In case many threads are modifying elements of the queue, the programmer needs to assure that two or more threads do not attempt to remove an element at the same time. Several programming flaws can leave such vulnerabilities in the code.

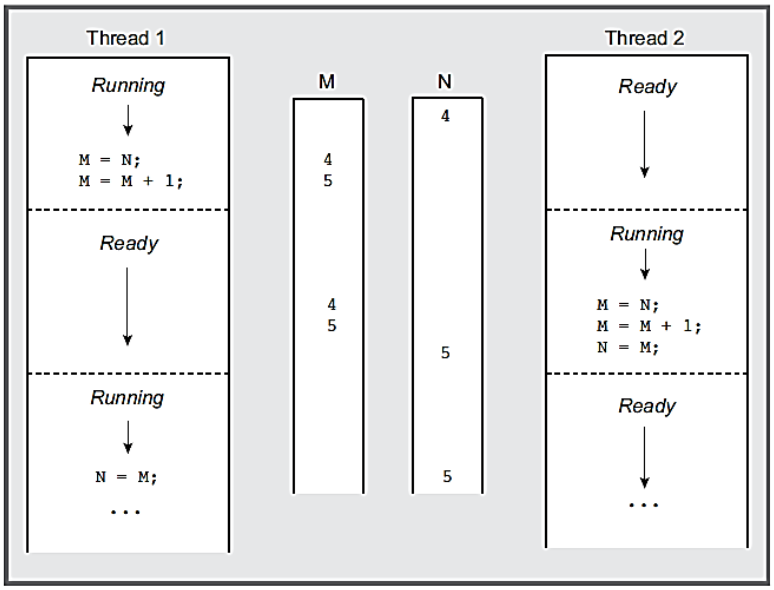
Example

There are two threads i.e. Thread 1 and Thread 2, operation of both of these is the same i.e.

```
Thread 1          Thread 2
{                 {
M = N             M = N
M = M + 1         M = M + 1
N = N             N = N
}                 }
```

Let's suppose that the initial value of N is 4 though the final values after execution of thread 1 will be M=5, N=5. After completion of thread 1 when thread 2 starts executing, the initial value of N is 5 (because of thread 1 execution) thus final values become M=6 and N=6.

When these same threads are run concurrently as shown in the following figure.



When the thread is executing, the value of N is 4 initially. After two instructions thread is switched. Thus the value of M becomes 5 and the value of N is not changed because 3rd instruction is not executed yet. Since the thread is switched therefore thread 2 starts execution and value N is 4 here. It executes all three instructions and values become M=5 and N=5. Again thread is switched and the remaining 1 instruction of thread 1 executes which affects the N variable and because of this final value of N becomes 5.

Final Values in normal processing

M=6 and N=6

Final Values in concurrent processing

M=5 and N=5

Thus we can see in many situations the final output of concurrent processing may not be same as it is in the normal processing

Topic No – 132

A Simple Solution to Critical Section Problem

When problems and errors arise because of parallelism or concurrent processing these types of errors are of critical section problem i.e. A critical resource which is used by number of processes or threads. This critical section problem can be handled by the following ways:

Using Global Variables

This solution uses a global variable named Flag to indicate to all threads that a thread is modifying a variable. A thread turns it TRUE before modifying a variable turns it to FALSE after modifying it. Each thread would check this Flag before modifying the variable. If the Flag is TRUE then if indicates that the variable is being used by some other thread.

```
BOOL Flag = FALSE;
DWORD N;
. . .
DWORD WINAPI ThreadFunc(TH_ARGS pArgs)
{
   . . .
   while (Flag) Sleep (1000);
   Flag = TRUE;
   N++;
   Flag = FALSE;
   . . .
}
```

Even in this case, the thread could be preempted between the time FLAG is tested and the time FLAG is set to TRUE; the first two statements form a critical code region that is not properly protected from concurrent access by two or more threads. Another attempted solution to the critical code region synchronization problem might be to give each thread its own copy of the variable, as follows:

```
DWORD WINAPI ThreadFunc (TH_ARGS pArgs)
{  DWORD N;
   ... N++; ...
}
```

Topic No – 133

Volatile Storage

The volatile stage is a Windows level or Compiler level facility provided when using incrementing operations which change the shared variables to reduce the conflicts that arise because of switching.

Latent Defects

Even if the synchronization problem is somehow resolved there still may remain some latent problems. A thread code may switch to another thread while a variable value has been modified in a register without writing it back. The use of registers for intermediate operations is a compiler optimization technique.

Turning off optimization may adversely affect the performance of the whole program but not always sometimes it may slow the program. ANSI C provides a qualifier volatile for this purpose. A variable with a volatile qualifier will always be accessed from memory for operations and will always be stored in memory after any operation. A volatile qualifier also means that the variable can be accessed at any instance. A volatile qualifier should only be used where necessary as it degrades the performance.

Using volatile

- Use volatile only if a variable is being modified by two or more threads.
- Even if it’s read only for two or more threads but the outcome of the threads depends on its new value.

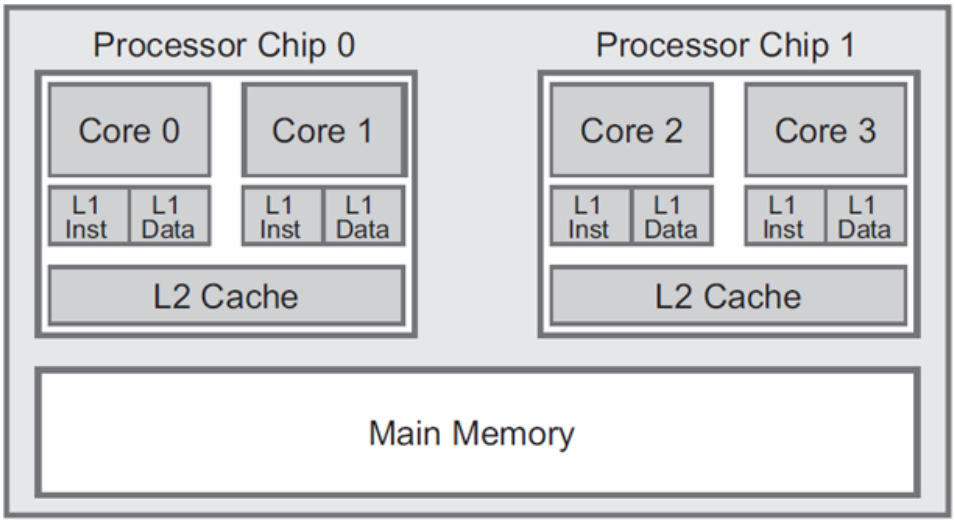
Content Development

Topic No - 134

Memory Architecture and Memory Barriers

Cache Coherency:
The volatile qualifier does not assure that the changes are visible to processors in a desired order. Processor usually hold the values in cache before writing them back to memory. This may alter the sequence in which different processors see the values.

Memory Barriers:
To assure that the memory is accesses in the desired order use memory barrier or memory fences. The interlocked function provide memory barrier. Further, the concept is clarified the diagram showing 4 processors in a system on two dual core chips.



Memory System Architecture

The diagram depicts 4 processor core on two chips. Each core has its own register with intermediate values of variables. Each core has separate Level-1 cache for instruction and data. A common larger L2 cache for cores on each chip. Memory is shared among cores.

Volatile qualifiers only assures that the new data values will be updated in L1. There is no guarantee that the values will be visible to other thread running on different processors. Memory barriers assure that the main memory is updated and cache of all processors is coherent.

For example if core 0 updates a variable N using a memory barrier. If core 3 L1 cache also contains the values of N then the value in its cache is either updated or removed so that core 3 could access the new value coherent value of N. This certainly incurs a huge cost. Moving data within the core registers costs less than a cycle whereas more data from one core to another via main memory can cost 100s of cycles.

Topic No - 135

Interlocked Functions:

Interlocked functions are most suited if variable with volatile scope only need to be incremented, decremented and exchanges. Interlocked functions are simpler and faster and easy to use. However, they do pose the performance drawback as they do generate a memory barrier.

The most basic of these functions are InterlockedIncrement() and InterlockedDecrement().

Long InterlockedIncrement(LONG volatile *Addend)

Long InterlockedDecrement(LONG volatile *Addend)

They both use a 32-bit signed variable as parameter that should be stored at 4-byte boundary in memory. They should be used whenever possible to improve performance.

For 64-bit systems InterlockedIncrement64() and InterlockedDecrement64() can be used with Addend placed at 8-byte boundary.

The function can be used in following way:

InterlockedIncrement(&N);

N should be volatile integer placed on appropriate memory boundary. Function return the new value of N. However some other thread may preempt before the value is returned and change the value of N. Do not call it twice to increment the value by two as the thread may be preempted between both calls. Rather use InterlockedExchangeAdd().

Topic No - 136

Local and Global Storage:

Another criterion for correct thread code is that global storage should not be used for local storage purpose. If a global variable is used to store thread specific data then it will be used by other threads as well. This will result in incorrect behavior no matter how the rest of code is written. The following example is incorrect usage of the global variables.

```
DWORD N;

....

DWORD WINAPI ThreadFunc (TH_ARGS pArgs)

{

        ...

        N = 2 * pArgs->Count; ...

        /* Use N; value is specific to this call to ThreadFunc*/

}
```

N is kept global but is used to store thread specific information from its parameters structure. For example, if many thread functions are running at the same time and each function modifies N with its own parameters, no function will be able to use N properly because N is global and all threads are running at the same time and affecting N's value. Due to this problem, the final results will be incorrect.

It's important to know when to use local and when to use global variables while dealing with these variables. If you know a variable contains thread-specific information, make it local. You can make a variable global if you know the information in it will be used by other threads.

Adhering to such practices can become even more convoluted when single threaded programs are converted to run as multi-threaded programs have threads being run in parallel. For example:

```
 DWORD N=0;

....

for(int i=0; i < MAX; ++i) {

        ...Allocate and initialize pArgs data ...
```

```
        N += ThreadFunc(pArgs);
}
...
DWORD WINAPI ThreadFunc (ARGS pArgs)
{
        DWORD result;
        ...
        result=...;
        return result;
}
```

The code above is for a single-threaded program. The program executes sequentially. The thread function is called repeatedly in a for loop. When a thread is called, the information it returns is stored in N, and the process is repeated for the next thread and so on. Because this is a single-threaded program, only one thread will be executed at a time. If you alter this code to a multi-threaded program, you will face issues.

Topic No - 137

How to write thread Safe Code:

Guide lines for writing thread safe code to ensure that the code runs smoothly in a threaded environment. When more than one thread can run the same code without introducing synchronization issues, it is said to be thread safe.

Variables that are required locally should not be accessible globally. They should be placed on Stack or in Data Structure passed to thread or the thread TLS. If a function is being used by several thread and it contains a variable that is thread specific such as a counter than store it in TLS or thread dedicated DS. Do not store it in global memory.

Avoid race conditions. If some required variables are uninitialized then create suspended threads until variables are initialized. If some condition needs to be met before a code block is executed then make sure the condition is met by waiting on synchronization objects.

Thread should not change the process environment. Changing environment by one thread will effect other threads. Thread should not change standard input or output devices. Also it should not change environment variables. A primary thread may change process environment as an exception. In that case the same environment used by rest of threads.

As a principle primary thread ensures that no other thread changes the process environment. All variable that are meant to be shared among threads are either kept global or static. They are protected by synchronization or interlocked mechanism using memory barriers.

Topic No - 138

Thread Synchronization Objects

Windows support different types of synchronization objects. The objects can be used to enforce synchronization and mutual exclusion.

Synchronization Mechanism

One method that has been used is by WaitForSingleObject() or WaitForMultipleObjects(). Using this method a thread waits for other threads to terminate. Other method used File Locks when multiple processes tries to access a file. Windows provides four different synchronization objects i.e. Mutexes, Semaphores, Events and Critical Section.

Risks in Using Synchronization Objects

There are always inherent risks involved in the use of such objects such as deadlocks. Care is required while using these objects. In higher version of Windows other objects like SRW locks and Condition variables are also available. Other advanced objects are waitable timers and IO completion ports.

Topic No - 139

CRITICAL_SECTION Objects

Critical Section

Critical section is the part of the code that can only be executed by one thread at a time. Windows provides the CRITICAL_SECTION objects as a simple lock mechanism for solving critical section problem. CRITICAL_SECTION Objects are initialized and deleted but do not have handles and are not shared among processes. Only one thread at a time can be in the CS variable, although many threads may enter and leave CS at numerous instances.

The function used to initialize and delete CRITICAL_SECTION are

VOID InitializeCriticalSection(LPCRITICAL_SECTION lpCriticalSection)

VOID DeleteCriticalSection(LPCRITICAL_SECTION lpCriticalSection)

When a thread wants to enter into the critical section it must call EnterCriticalSection() and when it leaves it should call LeaveCriticalSection().

A call to LeaveCriticalSection() must match an EnterCriticalSection(). Multiple threads may call EnterCriticalSection() but only one thread is allowed to enter the critical section while the rest are blocked. A blocked thread can enter the critical section when LeaveCriticalSection() is invoked.

VOID EnterCriticalSection(LPCRITICAL_SECTION lpCriticalSection)

VOID LeaveCriticalSection(LPCRITICAL_SECTION lpCriticalSection)

Recursive Critical Section

If one thread has entered a CS it can enter again. Windows maintain a count. Thread will have to leave as many times as it enters. This is implemented to support recursive functions and to make shared library functions thread safe. There is no timeout to EnterCriticalSection(). A thread will remain blocked forever if a matching Leave call is not received. A thread will remain blocked forever if a matching Leave call is not received.

However as thread can always poll to see whether another thread owns a CS using the function:

BOOL TryEnterCriticalSection(LPCRITICAL_SECTION lpCriticalSection)

If the function returns TRUE then it indicates that the current thread now owns the CS. If it returns false then it indicates that the CS is owned by some other thread and it is not safe to enter CS.

Since CRITICAL_SECTION is a user space object therefore it has apparent advantages over other kernel space objects.

Topic No - 140

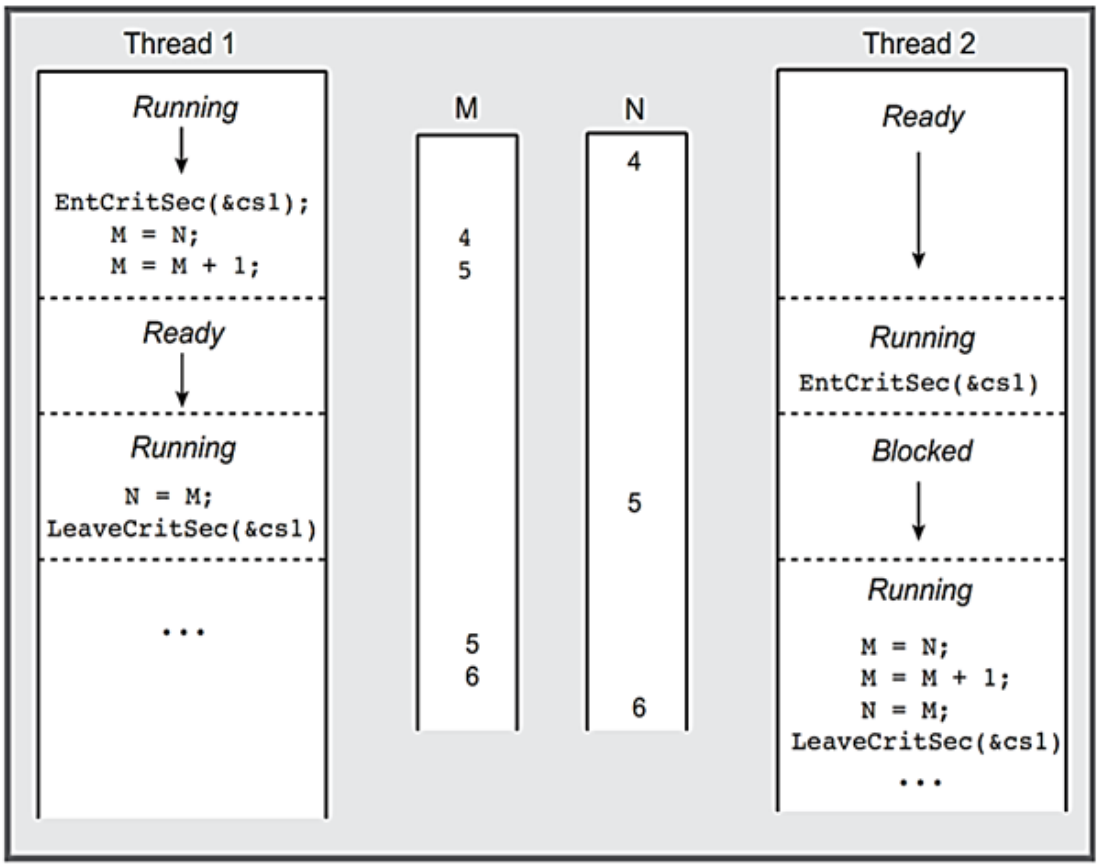
CRITICAL_SECTION for Protecting Shared Variables

In this module, we will study how to use protected shared variables in critical section and how critical section can assist us use protected shared variables.

Using the critical section construct is easy and intuitive. Consider an example of a server that maintains status related information in shared variables, information like number of requests, number of responses and requests currently under process. Since such count variables are shared therefore only one thread can be allowed to modify it at a time.

CRITICAL_SECTION construct can be used easily to ensure this. In this solution also an intermediate variable is used to emphasize the role of CRITICAL_SECTION.

```
 CRITICAL_SECTION cs1;
 volatile DWORD N = 0;
/* N is a global variable, shared by all threads. */
InitializeCriticalSection (&cs1);
    . . .
/* Create one or more threads using ThreadFunc */
    . . .
DWORD WINAPI ThreadFunc (TH_ARGS pArgs);
{
    DWORD M;
    __try {
        EnterCriticalSection (&cs1);
    if (N < N_MAX) { M = N; M += 1; N = M; }
    } __finally {
        LeaveCriticalSection (&cs1)
    }
}
    ...
DeleteCriticalSection (&cs1);
```



Topic No 141:

Protect a Variable with a Single Synchronization Object

In this module, we will discuss a guideline how to protect shared variables, how to use synchronization object, and what kind of mapping is there between synchronization object and variables like one to one or one to many.

Single Synchronization Object

All the variables within the critical section must be guarded by a single object. Using different objects within the same thread or using different objects across numerous threads sharing same data would be incorrect. Shared variables must be protected by a single object across all threads for mutual exclusion to work.

Below is given an example of incorrect use of synchronization object. In this example, two different objects are using the same variable N. Such code may generate incorrect results therefore share all variables using a single object.

```
DWORD WINAPI ThreadFunc (ARGS pArgs)
{
    . . .
    EnterCriticalSection(&cs1);
    N += 5;
    LeaveCriticalSection(&cs1);
    . . .
    InterlockedDecrement(&N);
    . . .
    EnterCriticalSection(&cs2);
    N -= 5;
    LeaveCriticalSection(&cs2);
    . . .
}
```

Topic No 142:

Producer-Consumer Problem

In this module, we will discuss a version of Producer-Consumer problem.

Producer-Consumer Threads

Producer consumer problem is a classical problem in mutual exclusion. It has many versions. Here we describe a simplistic version. This version clearly shows how to build and protect data structures for storing objects. Also we discuss how to establish invariant properties of variables which are always TRUE outside CS. In addition to primary threads there are two more threads: a producer and a consumer thread. The producer periodically creates a message. The message is contained in a table. The consumer on request of the user displays the message. The displayed data must be most recent and no data should be displayed twice. Do not display data while it is being updated by the producer. Do not display old data. Some of the produced messages may never be used and maybe lost. This is also like the pipeline model in which a message moves from one thread to another. The producer also computes a simple checksum of the message. Consumer makes sure by checking the checksum. If the consumer accesses the table while it is being updated the table will be invalid. CS ensures that this does not happen. The invariant is that the checksum is correct for current message contents.

Topic No 143:

Sample Program for Producer-Consumer Problem

Producer-Consumer

Producer consumer problem is a classical problem in mutual exclusion. Based on limitations and function described the following program is developed:

```
/* Chapter 9. simplePC.c */
/* Maintain two threads, a Producer and a Consumer */
/* The Producer periodically creates checksummed mData buffers, */
/* or "message block" which the Consumer displays when prompted */
/* by the user. The consumer reads the most recent complete */
/* set of mData and validates it before display */

#include "Everything.h"
#include <time.h>
#define DATA_SIZE 256
typedef struct MSG_BLOCK_TAG { /* Message block */
        CRITICAL_SECTION mGuard;       /* Guard the message block structure */
        DWORD fReady, fStop;
                /* ready state flag, stop flag   */
        volatile DWORD nCons, mSequence; /* Message block mSequence number         */
        DWORD nLost;
time_t mTimestamp;
        DWORD mChecksum; /* Message contents mChecksum               */
```

```
        DWORD mData[DATA_SIZE]; /* Message Contents                */
} MSG_BLOCK;
/*      One of the following conditions holds for the message block     */
/*        1)    !fReady || fStop                                        */
/*      nothing is assured about the mData              OR              */
/*        2)    fReady && mData is valid                                */
/*                        && mChecksum and mTimestamp are valid         */
/* Also, at all times, 0 <= nLost + nCons <= mSequence                   */
/* Single message block, ready to fill with a new message    */
MSG_BLOCK mBlock = { 0, 0, 0, 0, 0 };
DWORD WINAPI Produce (void *);
DWORD WINAPI Consume (void *);
void MessageFill (MSG_BLOCK *);
void MessageDisplay (MSG_BLOCK *);
int _tmain (int argc, LPTSTR argv[])
{
        DWORD status;
        HANDLE hProduce, hConsume;

        /* Initialize the message block CRITICAL SECTION */
        InitializeCriticalSection (&mBlock.mGuard);
        /* Create the two threads */
        hProduce = (HANDLE)_beginthreadex (NULL, 0, Produce, NULL, 0, NULL);
```

```
        if (hProduce == NULL)
                ReportError (_T("Cannot create Producer thread"), 1, TRUE);
        hConsume = (HANDLE)_beginthreadex (NULL, 0, Consume, NULL, 0, NULL);
        if (hConsume == NULL)
                ReportError (_T("Cannot create Consumer thread"), 2, TRUE);

        /* Wait for the Producer and Consumer to complete */

        status = WaitForSingleObject (hConsume, INFINITE);
        if (status != WAIT_OBJECT_0)
                ReportError (_T("Failed waiting for Consumer thread"), 3, TRUE);
        status = WaitForSingleObject (hProduce, INFINITE);
        if (status != WAIT_OBJECT_0)
                ReportError (__T("Failed waiting for Producer thread"), 4, TRUE);
        DeleteCriticalSection (&mBlock.mGuard);
        _tprintf (_T("Producer and Consumer threads have terminated\n"));
        _tprintf (_T("Messages Produced: %d, Consumed: %d, Lost: %d.\n"),
                mBlock.mSequence, mBlock.nCons, mBlock.mSequence - mBlock.nCons);
        return 0;
}
DWORD WINAPI Produce (void *arg)
/* Producer thread - Create new messages at random intervals */
{
```

```
        srand ((DWORD)time(NULL)); /* Seed the random # generator      */


        while (!mBlock.fStop) {
                /* Random Delay */
                Sleep(rand()/100);


                /* Get the buffer, fill it */


                EnterCriticalSection (&mBlock.mGuard);
                __try {
                        if (!mBlock.fStop) {
                                mBlock.fReady = 0;
                                MessageFill (&mBlock);
                                mBlock.fReady = 1;
                                InterlockedIncrement (&mBlock.mSequence);
                        }
                }
                __finally { LeaveCriticalSection (&mBlock.mGuard); }
        }
        return 0;
}
DWORD WINAPI Consume (void *arg)
{
```

```
CHAR command, extra;
/* Consume the NEXT message when prompted by the user */
while (!mBlock.fStop) { /* This is the only thread accessing stdin, stdout */
        _tprintf (_T("\n**Enter 'c' for Consume; 's' to stop: "));
        _tscanf (_T("%c%c"), &command, &extra);
        if (command == _T('s')) {
                /* ES not needed here. This is not a read/modify/write.
                 * The Producer will see the new value after the Consumer returns */
                mBlock.fStop = 1;
        } else if (command == _T('c')) { /* Get a new buffer to Consume */
                EnterCriticalSection (&mBlock.mGuard);
                __try {
                        if (mBlock.fReady == 0)
                                _tprintf (_T("No new messages. Try again later\n"));
                        else {
                                MessageDisplay (&mBlock);
                                mBlock.nLost = mBlock.mSequence - mBlock.nCons + 1;
                                mBlock.fReady = 0; /* No new messages are ready */
                                InterlockedIncrement(&mBlock.nCons);
                        }
                }
                __finally { LeaveCriticalSection (&mBlock.mGuard); }
        } else {
```

```
                    _tprintf (_T("Illegal command. Try again.\n"));
                }
        }
        return 0;
}
void MessageFill (MSG_BLOCK *msgBlock)
{
        /* Fill the message buffer, and include mChecksum and mTimestamp */
        /* This function is called from the Producer thread while it */
        /* owns the message block mutex*/

        DWORD i;

        msgBlock->mChecksum = 0;
        for (i = 0; i < DATA_SIZE; i++) {
                msgBlock->mData[i] = rand();
                msgBlock->mChecksum ^= msgBlock->mData[i];
        }
        msgBlock->mTimestamp = time(NULL);
        return;
}
void MessageDisplay (MSG_BLOCK *msgBlock)
{
```

```
        /* Display message buffer, mTimestamp, and validate mChecksum  */
        /* This function is called from the Consumer thread while it          */
        /* owns the message block mutex                                       */
        DWORD i, tcheck = 0;


        for (i = 0; i < DATA_SIZE; i++)
                tcheck ^= msgBlock->mData[i];
        _tprintf (_T("\nMessage number %d generated at: %s"),
                msgBlock->mSequence, _tctime (&(msgBlock->mTimestamp)));
        _tprintf (_T("First and last entries: %x %x\n"),
                msgBlock->mData[0], msgBlock->mData[DATA_SIZE-1]);
        if (tcheck == msgBlock->mChecksum)
                _tprintf (_T("GOOD ->mChecksum was validated.\n"));
        else
                _tprintf (_T("BAD  ->mChecksum failed. message was corrupted\n"));


        return;
}
```

Topic No 144:

Mutexes

In this module we will discuss mutexes. Mutex, a short form of Mutual Exclusion. Windows provides an object called a mutex. Using this object we can enforce mutual exclusion.

Mutexes have some advantages beyond CRITICAL_SECTION. Mutexes can be named and have handles. They can also be used for interprocess communication between threads in different processes. For example, if two processes share files through memory maps, then mutexes can be used for protection. Mutexes also allow timeout values. Mutexes can automatically become signaled once abandoned by the terminating thread. A thread gains ownership to mutex by waiting successfully on mutex handle using WaitForSingleObject() or WaitForMultipleObject(). Ownership is released using ReleaseMutex(). A thread should be careful about releasing a thread as soon as possible. A thread can acquire a single mutex several times. A thread will not be blocked if it already has ownership. The recursive property holds for Mutexes also.

Windows functions used to manage mutexes are

- CreateMutex()
- ReleaseMutex()
- OpenMutex()

```
HANDLE CreateMutex (
   LPSECURITY_ATTRIBUTES lpsa,
   BOOL bInitialOwner,
   LPCTSTR lpMutexName)
```

If bInitialOwner is TRUE, handover ownership to the calling thread immediately. However, it is ignored if the mutex already exists as per its name.

lpMutexName is the name of the mutex. The name is assigned as per rules of the Windows namespace.

If NULL is returned it indicates failure.

OpenMutex() is used to open and existing named mutex. The Open operation is followed by Create. The main thread would usually create a mutex while other threads can open it. This construct allows to synchronize threads from different processes.

```
BOOL ReleaseMutex (HANDLE hMutex)
```

Similarly ReleaseMutex() releases the ownership of the Mutex for a thread. It fails if the Mutex is not already owned by the thread.

Topic No 145:

Mutexes, CRITICAL_SECTIONS and Deadlocks

In this module we will discuss deadlocks. Deadlocks may arise when Mutexes or CRITICAL SECTIONS are used.

Using Concurrency Objects

Concurrency objects must be used carefully, otherwise it can lead to a deadlock situation. Deadlocks are a byproduct of concurrency control. It occurs when two or more objects try to lock a resource at the same time.

An Example

For instance, there are two lists (A and B) with the same structure maintained by different worker threads. In one situation, it can be possible that an operation is only allowed if a certain element is either present in both the lists or none. An operation is invalid if an element is present in just one.

In another situation, an element in one list can not be in the other. Based on these situations, concurrency objects are required for both lists. Using a single mutex for both lists will degrade performance by restricting concurrent updates.

Next slide shows an implementation for this situation.

```
static struct {
   /* Invariant: List is a valid list. */
   HANDLE guard; /* Mutex handle. */
   struct ListStuff;
}  ListA, ListB;
...
DWORD WINAPI AddSharedElement (void *arg)
/* Add a shared element to lists A and B. */
{ /* Invariant: New element is in both or neither list. */
   WaitForSingleObject (ListA.guard, INFINITE);
   WaitForSingleObject (ListB.guard, INFINITE);
   /* Add the element to both lists ... */
   ReleaseMutex (ListB.guard);
   ReleaseMutex (ListA.guard);
   return 0;
}
DWORD WINAPI DeleteSharedElement (void *arg)
/* Delete a shared element to lists A and B. */
{
   WaitForSingleObject (ListB.guard, INFINITE);
   WaitForSingleObject (ListA.guard, INFINITE);
   /* Delete the element from both lists ... */
   ReleaseMutex (ListB.guard);
   ReleaseMutex (ListA.guard);
   return 0;
}
```

Avoiding deadlock situation

One method to avoid this situation is to maintain a hierarchy. All threads should follow the same hierarchy. They should acquire and release mutexes in the same order. In the example, the situation can be easily avoided if both the threads acquire mutexes in the same order.

Another technique that can be used is an array of mutexes and using WaitForMultipleObjects() with the fWaitAll flag. In this case, the thread will either acquire both the mutexes or none. This technique is not possible with CRITICAL_SECTION.

Topic No 146:

Mutexes and CS

This module is about Mutexes VS CRITICAL_SECTIONs. Mutexes and CS are similar in the sense that they are used to achieve the same goal. Both can be owned by a single thread, and other threads trying to gain access to a resource shall be denied access until the resource is released. However, there are few advantages of using mutexes against some performance drawbacks.

Advantages of Mutexes

Mutexes that are abandoned due to the abrupt termination of a thread are automatically signaled so that waiting threads are not blocked permanently. However, abrupt termination of a thread indicates a serious programming flaw. Mutex waits can time out, whereas CS wait does not. As they are named, mutexes are shareable over numerous threads in different processes. Threads that create the mutexes can acquire immediate ownership (slight convenience). However, in most cases, CSs will work considerably faster.

Topic No 147:

Semaphores

This module is about semaphores. Semaphore is simply a data structure that is used for concurrency control.

Semaphore Count

Semaphore maintains a count. A semaphore is in a signaled state when the count is greater than 0. Semaphore is unsignaled when the count is 0. Threads use the wait function on semaphore. The count is decremented when a waiting thread is released. The following functions are used: CreateSemaphore(), CreateSemaphoreEx(), OpenSemaphore(), and ReleaseSemaphore().

```
HANDLE CreateSemaphore (
    LPSECURITY_ATTRIBUTES lpsa,
    LONG lSemInitial,
    LONG lSemMax,
    LPCTSTR lpSemName)
```

lSemMax must be 1 or greater, which is the maximum value of semaphore count.

lSemInitial is the initial value.

0<= lSemInitial<= lSemMax.

A NULL returns in case of failure.

```
BOOL ReleaseSemaphore (
    HANDLE hSemaphore,
    LONG cReleaseCount,
    LPLONG lpPreviousCount)
```

lpPreviousCount gives the count preceding the release.

cReleaseCount gives the count after the release and must be greater than 0. The call will fail and return FALSE if it would cause the count to exceed the maximum, and the count will remain unchanged. Also, in this case, the release count will not be valid.

Any thread can release the semaphore, not just the one that acquired its ownership. Hence, there is no concept of abandonment.

Topic No 148:

Using Semaphores

This module is about using semaphores. We have already discussed semaphores. Now we will discuss how to use the structure and functions in a program to enforce mutual exclusion.

Semaphore Concepts

Classically, a semaphore count represents the number of available resources. Sem Maximum represents the number of resources. In a producer-consumer scenario, the producer would place an element in the queue and call ReleaseSemaphore(). The consumer will wait on the semaphore and decrease the count after consuming the item.

Thread Creation

Previously, in many examples, numerous threads were created in a suspended state and were not activated until all the threads were created. This problem can be handled by semaphores without suspending threads. A newly created thread will wait on a semaphore with the semaphore initialized by 0. The boss thread will call ReleaseSemaphore with the count set to the number of threads.

Topic No 149:

Semaphore Limitation

In this module we will discuss limitations of semaphore.

Limitations

Some limitations are encountered while using windows semaphore. How can a thread request to decrease the count by 2 or more? The thread will have to wait twice on the semaphore. Calling wait twice will not be atomic and execution may switch in between.

Below is given a code of two threads, thread1 on the left side and thread2 on the right side. The problem with the below code is that the wait functions are not atomic; they are separate wait functions. Because they are not single functions, switching between these two wait functions (after completion of first wait function) in thread1 may occur, and execution control may reach the thread2 wait function, resulting in a deadlock-like situation. So this is one of the limitations of semaphores.

```
/* hSem is a semaphore handle.
The maximum semaphore count is 2. */
   ...
/* Decrement the semaphore by 2. */
WaitForSingleObject (hSem, INFINITE);
WaitForSingleObject (hSem, INFINITE);
   ...
/* Release two semaphore counts. */
ReleaseSemaphore (hSem, 2, &PrevCount);
```

```
/* hSem is a semaphore handle.
The maximum semaphore count is 2. */
   ...
/* Decrement the semaphore by 2. */
WaitForSingleObject (hSem, INFINITE);
WaitForSingleObject (hSem, INFINITE);
   ...
/* Release two semaphore counts. */
ReleaseSemaphore (hSem, 2, &PrevCount);
```

How can we deal with this limitation? Make the two wait functions atomic by using a CriticalSection to avoid switching during the wait, as shown in the below code.

```
/* Decrement the semaphore by 2. */
EnterCriticalSection (&csSem);
WaitForSingleObject (hSem, INFINITE);
WaitForSingleObject (hSem, INFINITE);
LeaveCriticalSection (&csSem);
   ...
ReleaseSemaphore (hSem, 2, &PrevCount);
```

Other Solutions

Another solution one can suggest is the use of WaitForMultipleObjects() with an array holding multiple references to the same semaphore. However, this solution will fail readily as the call to WaitForMultipleObjects() fails when it detects duplicate objects in the array. Secondly, all the handles may get signals even if the count is 1.

Topic No 150:

Events

In this module, we will discuss Events. Events are synchronization objects like Mutexes, Semaphores and Critical Section that can be used for concurrency control.

Events can indicate to other threads that a specific condition now holds, such as some message being available. Multiple threads can be released from wait simultaneously when an event is triggered. Events are classified as either manual-reset or auto-reset. Event property is set using CreateEvent(). A manual-reset event can signal several waiting threads simultaneously and can be reset. An auto-reset event signals a single waiting thread, and the event is reset automatically. The functions used to handle events are CreateEvent(), CreateEventEx(), OpenEvent(), SetEvent(), ResetEvent() and PulseEvent().

```
HANDLE CreateEvent (
    LPSECURITY_ATTRIBUTES lpsa,
    BOOL bManualReset,
    BOOL bInitialState,
    LPTCSTR lpEventName)
```

The function creates an event. bManualReset is set to TRUE to set a manual reset event. If bInitialState is TRUE, the event is set to a signaled state. A named event can be opened using OpenEvent() from any process.

```
BOOL SetEvent (HANDLE hEvent)
BOOL ResetEvent (HANDLE hEvent)
BOOL PulseEvent (HANDLE hEvent)
```

The above three functions are used to control events. A thread can signal the event by using SetEvent(). In the case of auto-reset, a single thread is released out of many. Event automatically returns to a non-signaled state. If no threads are waiting, the event remains in a signaled state until a thread waits on it. In this case, the thread will be immediately released.

If the event is manual-reset, it remains signaled until a thread explicitly calls ResetEvent(). Meanwhile, all the waiting threads are released. Consequently, other threads may also wait to be released immediately before the reset.

PulseEvent() releases all threads currently waiting for a manual reset. The event is automatically reset. In the case of an auto-reset event, PulseEvent() releases a single waiting thread, if any.

Topic No 151:

Event Usage Models

In this module, we will discuss Event Usage Models. Event can be implemented using different types of models. These models are designed according to the usage.

Combinations

There can be four distinct ways to use events. These four models culminate from combinations of SetEvent(), PulseEvent() and use of auto and manual events. Each combination proves useful depending on the situation. The combinations are highlighted in the table.

	Auto-Reset Event	Manual-Reset Event
`SetEvent`	Exactly one thread is released. If none is currently waiting on the event, the first thread to wait on it in the future will be released immediately. The event is automatically reset.	All currently waiting threads are released. The event remains signaled until reset by some thread.
`PulseEvent`	Exactly one thread is released, but only if a thread is currently waiting on the event. The event is then reset to nonsignaled.	All currently waiting threads, if any, are released, and the event is then reset to nonsignaled.

Understanding Events

An auto-reset event can be conceptualized as a door that automatically shuts when opened. Whereas the manual reset does not shuts automatically when opened. Hence, PulseEvent() can be considered as a door that is open which shuts when a single thread passes through in the case of auto-reset. While in the case of a manual reset, multiple waiting threads can pass through. SetEvent(), on the other hand, simply opens the door and releases a thread.

Topic No 152:

Producer-Consumer Solution Using Events

In this module, we will try to solve Producer-Consumer problem using Events.

Producer Consumer

Here is another example that provides a solution to our Producer-Consumer problem using Events. The solution uses Mutexes rather than CSs. A combination of auto-reset and SetEvent() is used in the producer to ensure only one thread is released. Mutexes in the program make access to the message data structure mutually exclusive while an event is used to signal the availability of new messages.

```
#include "Everything.h"

#include <time.h>

#define DATA_SIZE 256

typedef struct MSG_BLOCK_TAG { /* Message block */

        HANDLE      mGuard;             /* Mutex to uard the message block structure*/

        HANDLE      mReady;             /* "Message ready" auto-reset event   */

        DWORD       fReady, fStop;      /* ready state flag, stop flag   */

        volatile DWORD mSequence;       /* Message block mSequence number*/

volatile DWORD nCons, nLost;

        time_t mTimestamp;

        DWORD       mChecksum;          /* Message contents mChecksum*/

        DWORD       mData[DATA_SIZE]; /* Message Contents*/

} MSG_BLOCK;

/*One of the following conditions holds for the message block */

/*1)    !fReady || fStop*/

/*nothing is assured about the mData          OR */

/*2)    fReady && mData is valid*/

/* && mChecksum and mTimestamp are valid*/

/*  Also, at all times, 0 <= nLost + nCons <= mSequence*/

/* Single message block, ready to fill with a new message    */
```

```
MSG_BLOCK mBlock = { 0, 0, 0, 0, 0 };
DWORD WINAPI Produce (void *);
DWORD WINAPI Consume (void *);
void MessageFill (MSG_BLOCK *);
void MessageDisplay (MSG_BLOCK *);
int _tmain (int argc, LPTSTR argv[])
{
        DWORD status;
        HANDLE hProduce, hConsume;

        /* Initialize the message block mutex and event (auto-reset) */
        mBlock.mGuard = CreateMutex (NULL, FALSE, NULL);
        mBlock.mReady = CreateEvent (NULL, FALSE, FALSE, NULL);
        /* Create the two threads */
        hProduce = (HANDLE)_beginthreadex (NULL, 0, Produce, NULL, 0, NULL);
if (hProduce == NULL)
                ReportError (_T("Cannot create producer thread"), 1, TRUE);
        hConsume = (HANDLE)_beginthreadex (NULL, 0, Consume, NULL, 0, NULL);
        if (hConsume == NULL)
                ReportError (_T("Cannot create consumer thread"), 2, TRUE);
        /* Wait for the producer and consumer to complete */
        status = WaitForSingleObject (hConsume, INFINITE);
if (status != WAIT_OBJECT_0)
```

```
            ReportError (_T("Failed waiting for consumer thread"), 3, TRUE);
        status = WaitForSingleObject (hProduce, INFINITE);
        if (status != WAIT_OBJECT_0)
            ReportError (__T("Failed waiting for producer thread"), 4, TRUE);
        CloseHandle (mBlock.mGuard);
        CloseHandle (mBlock.mReady);
        _tprintf (_T("Producer and consumer threads have terminated\n"));
_tprintf (_T("Messages produced: %d, Consumed: %d, Known Lost: %d\n"),
            mBlock.mSequence, mBlock.nCons, mBlock.mSequence - mBlock.nCons);
        return 0;
}
DWORD WINAPI Produce (void *arg)
/* Producer thread - Create new messages at random intervals */
{
        srand ((DWORD)time(NULL));        /* Seed the random # generator        */
while (!mBlock.fStop) {
            /* Random Delay */
            Sleep(rand()/5); /* wait a long period for the next message */
                /* Adjust the divisor to change message generation rate */
            /* Get the buffer, fill it */

            WaitForSingleObject (mBlock.mGuard, INFINITE);
            __try {
```

```
                        if (!mBlock.fStop) {
                                mBlock.fReady = 0;
                                MessageFill (&mBlock);
                                mBlock.fReady = 1;
                        InterlockedIncrement(&mBlock.mSequence);
                                SetEvent(mBlock.mReady); /* Signal that a message is ready. */
                        }
                }
                __finally { ReleaseMutex (mBlock.mGuard); }
        }
        return 0;
}
DWORD WINAPI Consume (void *arg)
{
DWORD ShutDown = 0;
        CHAR command[10];
        /* Consume the NEXT message when prompted by the user */
        while (!ShutDown) { /* This is the only thread accessing stdin, stdout */
                _tprintf (_T ("\n**Enter 'c' for Consume; 's' to stop: "));
                _tscanf_s (_T("%9s"), command, sizeof(command)-1);
                if (command[0] == _T('s')) {
                        WaitForSingleObject (mBlock.mGuard, INFINITE);
                        ShutDown = mBlock.fStop = 1;
```

```
                        ReleaseMutex (mBlock.mGuard);
} else if (command[0] == _T('c')) { /* Get a new buffer to consume */
                        WaitForSingleObject (mBlock.mReady, INFINITE);
                        WaitForSingleObject (mBlock.mGuard, INFINITE);
                        __try {
                                if (!mBlock.fReady) _leave; /* Don't process a message twice */
                                /* Wait for the event indicating a message is ready */
                                MessageDisplay (&mBlock);
                                InterlockedIncrement(&mBlock.nCons);
                                mBlock.nLost = mBlock.mSequence - mBlock.nCons;
                        mBlock.fReady = 0; /* No new messages are ready */
                        }
                        __finally { ReleaseMutex (mBlock.mGuard); }
                } else {
                        _tprintf (_T("Illegal command. Try again.\n"));
                }
        }
        return 0;
}
void MessageFill (MSG_BLOCK *msgBlock)
{
/* Fill the message buffer, and include mChecksum and mTimestamp        */
        /* This function is called from the producer thread while it   */
```

```
        /* owns the message block mutex*/

        DWORD i;

        msgBlock->mChecksum = 0;
        for (i = 0; i < DATA_SIZE; i++) {
                msgBlock->mData[i] = rand();
                msgBlock->mChecksum ^= msgBlock->mData[i];
}
        msgBlock->mTimestamp = time(NULL);
        return;
}
void MessageDisplay (MSG_BLOCK *msgBlock)
{
        /* Display message buffer, mTimestamp, and validate mChecksum  */
        /* This function is called from the consumer thread while it */
        /* owns the message block mutex                                */
        DWORD i, tcheck = 0;
        TCHAR timeValue[26];
for (i = 0; i < DATA_SIZE; i++)
                tcheck ^= msgBlock->mData[i];
        _tctime_s (timeValue, 26, &(msgBlock->mTimestamp));
        _tprintf (_T("\nMessage number %d generated at: %s"),
```

```
            msgBlock->mSequence, timeValue);
        _tprintf (_T("First and last entries: %x %x\n"),
            msgBlock->mData[0], msgBlock->mData[DATA_SIZE-1]);
        if (tcheck == msgBlock->mChecksum)
            _tprintf (_T("GOOD ->mChecksum was validated.\n"));
        else
            _tprintf (_T("BAD  ->mChecksum failed. message was corrupted\n"));


        return;
}
```

Content Development

Topic No -154

Content:

Windows Synchronization Objects and their properties

Lecture 153-Reading Material

Windows operating system is a multithreaded kernel that provides support for real time applications and multiprocessors.

Program design and performance can be simplified and improved by threads, but it requires much care to use threads in case of shared resources at the same time.

Process Synchronization is a way to coordinate processes that use shared data

Topic No -154

Content:

Lecture 154-Reading Material

To minimize the concurrency related issues and errors following are the main guidelines

1)-The first issue is timeout when you call a wait on any object then you must specify the timeout otherwise the thread will be blocked for an indefinite period, so a programmer should take care of it.

2)- When a thread is in the critical section (CS) and it terminates without releasing the critical section then it will be impossible for the other thread in the queue to complete its execution and be blocked.

3)-Mutexes have a way out for the above problem in critical section, if a thread or process owns a mutex and it terminates, then mutex has an abandonment property, abandoned mutex handles are signaled, it is a useful feature not available with **CSs**. It abandons the process and other threads are not blocked and fail as in **CS** case. Mutex abandonment of a process shows a flaw in the programming.

4)- If you used the construct of **WaitForSingleObject()** and also specified the timeout then you must program in such a way that it checks if the wait ends due to timeout then it must release the critical resource.

5)-Exactly one of the waiting threads at a time should be given the ownership of mutex. Only the OS Scheduler decides which thread has the priority according to its scheduling policy. Program should not assume the priority of any particular thread over the other.

6)- A single mutex can be used to define several critical regions for several threads which are competing for the resource.

7)-To ensure optimization critical Region size should be minimum. Programmers must ensure that critical regions cannot be used concurrently.

8)-Performance decrease because of locks so minimize the use of locks inside the critical region

9)-The same data structure that stores the resource should also be used to store the mutexes because mutexes correspond to the resources.

10)- Invariant is the property that assures whether you have enforced the concurrency correctly or not.

12)- Complex conditions and decision structures should be avoided for entering into the critical region. Each critical region must have one entry and one exit.

13)- Must ensure that mutex is locked on entry and unlocked on exit.

14)- Avoid premature exits from the critical region such as **break**, **return** or **goto** statements, termination handlers are useful for protecting against such problems.

15)- If the critical code region becomes too lengthy (longer than one page, perhaps), but all the logic is required, consider putting the code in a function so that the synchronization logic will be easy to read and comprehend.

Topic No -155

Content:

More on interlocked functions

Lecture-155-Reading Material

The interlocked functions provide a simple mechanism for synchronizing access to a variable that is shared by multiple threads. They also perform operations on variables in an atomic manner. The threads of different processes can use these functions if the variable is in shared memory.

The **InterlockedIncrement** and **InterlockedDecrement** functions combine the steps involved in incrementing or decrementing a variable into an atomic operation.

Interlocked functions are as useful as they are efficient; they are implemented using atomic machine instructions (for this reason, they are sometimes called "**compiler intrinsic statements**").

Some Interlocked Functions and their details:

InterlockedExchange stores one variable into another.

==========

Long InterlockedExchange(

LONG volatile *Target

LONG Value

)

It returns the previous value of *Target and sets its value to Value.

==========

Long InterlockedExchangeAdd(

LONG volatile *Addend

LONG Increment

)

Increment is added to the ***addend** and the original value of ***addend** is returned.

==========

More on interlocked functions

Lecture-156-Reading Material

When numbers of threads are running concurrently, there is a competition for resources, memory is also a resource and threads compete for the memory. So, we have to devise a way to reduce or optimize this competition to improve the performance. When we allocate a memory dynamically it is allocated from the heap. Threads allocate memory and free memory using **malloc** () and **free** () functions respectively.

Managing Access to Heap

---

a)-Each thread that performs memory management can create a Handle to its own heap using **HeapCreate().** Memory allocation is then performed using using **HeapAlloc()** and **HeapFree()** rather than using **malloc()** and **free()**

b)- Consider third **party APIs** for concurrent memory management.

Synchronization Performance Impact

Lecture-157-Synchronization Performance Impact

Synchronization can impact on the performance following are the main reasons

a)- Locking, waiting and even interlocked operations are time consuming

b)- Locking requires kernel operation and waiting is expensive

c)- Only one thread at a time can execute a critical code region, reducing concurrency and produces almost serialization execution

d)- Many processes when competing for memory and cache can produce unexpected effects.

Gauging Performance Impact of Synchronization

Lecture-158-159-Synchronization Performance Impact

Reading Material

To enforce synchronization, we have many constructs like Critical Section, Mutexes, Semaphore. When we implement synchronization with these constructs there are some trade-offs performance wise.

We arrange data in the memory in such a way to avoid cache conflicts and allocation are aligned over optimal cache boundaries

Following are the main modifiers used for minimal memory contention

__declpec(align), with this modifier memory is aligned in such a way to minimize contention

Similarly __align_malloc() and __align_free() functions are used for allocation/deallocation.

Example 9-1 (see in the book)

We developed different programs in different seniors

1)-Without using synchronization – NS

2)-Using Critical Section -CS

3)-Using Mutexes-MX

4)-Using Interlocked Function-IN

Following are the main inferences we drew by running the programs using these scenarios.

We basically compare the performance on the basis of following three aspects

a)-Real Time

b)-User Time

c)-System Time

Inferences

1)- NS (no synchronization) and IN (interlocked functions) exhibit all most the same time for this example.

2)-CS (Critical Section) version costs 2 times more than IN

3)-MX(Mutexes) version costs more than 2 to 30 times than IN

4)-Any type of locking (even IN) is more expensive than no locking but we cannot discard it because of the significance of synchronization.

#Processes Clock Rate	StatsMX	statsCS
1, 1.4GHz	55.03	14.15
2, 2.0GHz	93.11	5.30

4, 2.4GHz	118.3	4.34
8, 1.7GHz	262.2	20.2

5- Mutexes are very slow, and unlike the behavior with CSs, performance degrades rapidly as the processor count increases. For instance, Table 9–1 shows the elapsed and times (seconds) for 64 threads and 256,000 work units on 1-, 2-, 4-, and 8-processor systems. CS performance, however, improves with processor count and clock rate.

Lecture-160-False Sharing Contention

Reading Material

While working with synchronization we face the problem of false sharing contention

Cache has cache lines having different capacities, if you have a large array with different elements on a single line and if we access any element on that line then the whole line will be locked.

If on one line we store more than one element then we have to face the consequences of false sharing contention.If any one element is locked on a single line then other elements will also be locked and can only be accessed serially.

False sharing contention occurs when two or more threads on different processes place variables on the same address line and a change in one variable will lock the whole cache line.

Solution

—----------

It will be better to align each variable in such a way that each variable is placed on a different cache line. We have to use this type of alignment in order to avoid false sharing.Of course when we use this type of alignment it will be expensive memory wise.

Tuning Performance with CS Spin Counts

Lecture 161-Reading Material

Different synchronization objects have different overheads. Critical Section overheads are moderate but we can future optimize these overheads by fine tuning the spin counts.

Critical Section works in user space; it does not work in kernel space and does not require complicated and/or convoluted system calls while mutexes invoke kernel calls for locking and unlocking like **ReleaseMutex()** requires system calls.

Lock bit is on

==========

If any thread invoke CS with the help of **EnterCriticalSection()** it test its lock bit, if lock bit is off ,it mean no other thread is entered in the CS yet, so lock bit atomically set and operation proceed without waiting hence locking and unlocking of CS is very efficient and it just require a couple of machine level instructions

The ID of the thread is stored in the CS data structure and it also stores the status of the recursive calls, so recursion overheads are also performed in the critical section.

Lock bit is off

=========

If CS is already locked then after invoking **EnterCriticalSection(),** the thread will enter into the tight loop, thread calls tight loop because it involves all the processes of the multiprocessor system and tight loop will repetitively check the **lock bit** whether it is off or on.

Spin count will tell how much time the tight loop repetitively checks the lock bit. If the lock bit repetitively checked without yielding the processor then it gives up calling **WaitForSingleObject()**

-Spin count determines the number of times the loop repeats and it is used in multiprocessor system

LeaveCriticalSection() turns off the lock bit and also informs the kernel by **ReleaseSemphore**() call in case there are any waiting threads.

Setting the Spin Counts

Lecture 162-Reading Material

Spin count is important due to the fact that you can optimize the Critical Section with the help of spin count. How to set the spin count it will depend on many factors

We can specify spin count when we initialize the Critical Section using the function

InitializeCriticalSectionAndSpinCount(). The main advantage of optimizing the spin count is to avoid the system calls, when there are less system calls then overheads will be reduced.

We can set spin count initially and as well as dynamically on runtime using the function **SetCriticalSectionSpinCount()**

According to MSDN 4000 is the good spin count for heap related functions but it depends, like if you have a short critical section then small spincount will work optimally.

Spin count should be set as per the dynamics of the application and the number of processes.

—----------

Slim Reader Writer Locks

Lecture 163-Reading Material

Slim reader writer locks are used to optimize the Synchronization and minimizing the overheads due to synchronization and mutual exclusion.

Lightweight Reader Writer Locks has two modes

a)- Exclusive Mode

b)- Shared Mode

-In Exclusive mode they can be locked and writable

-In shared mode reading is possible and different threads can read the data at a time in shared mode

SRWs can be used either in Exclusive Mode or in Shared mode but you can not upgrade or downgrade the Exclusive Mode or Shared Mode so Exclusive mode cannot be converted into Shared mode and Shared mode cannot be converted to Exclusive mode once acquired. Threads have to decide the mode before using SRWs whether it is exclusive or shared.

SRWs are light weight and slim and the size of the associated pointers are 32 bit or 64 bits only. No Kernel objects are associated with SRWs and hence SRWs locks require minimal resources.

SRWs do not support recursion unlike Critical Section, so SRW is simpler and faster.

The spin count value of SRW is preset optimally and cannot be set manually

It is not imperative to delete the SRW

Nonblocking calls are not associated with SRW.

==========

APIs for SRWs

Lecture 164-Reading Material

We use APIs to manage SRWs, Slim Read Write Constructs are light weight.

Following are the SRW APIs

In order to Initialize the SRWs Lock we use

InitializeSRWLock()

Similarly, there are two APIs which can access SRWs in shared mode

AcquiredSRWLockShared() and **ReleaseSRWLockShared()**.

There are also two APIs which are used to access SRWs in Exclusive mode

-AcquireSRWLockExclusive

-ReleaseSRWLockExclusive

Using SRWs

If any thread needs to read the shared data then we get the SRW in shared mode but the thread to write that data will use SRW in exclusive mode.

SRWs are used just like CS OR MUTEX in exclusive mode and are used in shared mode if the guarded variable is not changed by thread.

Improved Locking through SRWs

Lecture 165-Reading Material

-The main advantage of the SRWs is the shared mode, in shared mode you can read the locked variable. The thread which reads the locked variable need not to wait for the other threads to release it.

- Similarly, SRWs are light weight in implementation and resource requirement.

- The don’t allow recursion because in recursion you have to maintain the stack and it will require time and more execution

Evaluation SRWs performance by using threads shows that these are almost twice fast than critical Section

Improved Locking through SRWs

Lecture 166-Reducing Thread Contention

Thread Limitations:

Just like memory contention systems also have to face the thread contention

As the number of threads increases, it causes serious performance issues for following reasons

1)- When a thread is created almost 1MB space is reserved for that thread and with increasing threads memory area is piled up.

2)- With large number of threads, context-switching can become time consuming

3)-Thread context-switching can cause page faults.

4)- If you minimize the threads the you can lose the benefits of parallelism and synchronization

Optimizations:

Following are the main optimization techniques

1)- Use of semaphore throttles

2)-Using I/O completion ports

3)- Asynchronous I/O

4)-Using Thread Pools through callback functions

5)-Asynchronous Procedure Calls (APC)

6)- Third party frameworks like **OpenMP** and **cilk++**

Semaphore Throttles

The Solution:

- The solution is simple and can be incorporated into existing programs.
- The boss thread creates a semaphore with a small count say 4, depending upon the number of processors and other conditions.
- Each worker thread that wants to enter the critical section will wait on the semaphore before entering its critical section. The thread may use CS or Mutex to acquire the resource.
- The worker thread will release the semaphore after exiting its critical section.
- Overall the CS or mutex contention will be minimized. Thread execution will be serialized among few threads waiting on mutex or CS.
- The boss thread can control the count dynamically depending upon the overall progress.
- However, a maximum count for semaphore is set initially.

```
WaitForSingleObject (hThrottleSem, INFINITE);
while (TRUE) { /* Worker loop */
   WaitForSingleObject (hMutex, INFINITE);
      ... Critical code section ...
   ReleaseMutex (hMutex);
} /* End of worker loop */
ReleaseSemaphore (hThrottleSem, 1, NULL);
```

More variations:

- If a specific worker thread is considered to be expensive then it can be made to wait longer using some programming technique.

- This technique works well with older version of windows. NT6 onwards have optimization techniques of their own. This technique should be used with care in case of NT6.

- This technique can also be applied to reduce contention in other resources like files, memory etc.

Topic 168

Thread Pools

- Thread pools are easy to use. Also already developed programs that uses threads can be easily modified to use thread pools.
- Application creates a work object rather than a thread. Each work object is submitted to the thread pool.
- Each work object has a call back function and is identified by a handle like structure
- Thread pool manages a small number of “worker threads”

- Windows assigns worker threads to each worker object.
- Worker thread executes a worker object by invoking the callback function.
- When a worker object completes it returns to windows worker thread which in turn invokes another worker object.
- A work object should never invoke _endthreadex() within the callback function.

- Windows dynamically adjusts the number of worker threads based on process behaviour.
- The application can set upper and lower bounds for worker threads.
- There is a default thread pool. Also, the application can create additional pools

169

Thread Pools APIs:

- Thread pools are created and assigned thread objects
- The thread objects execute via a call back function
- Now, we discuss all the APIs for this purpose.

```
PTP_WORK CreateThreadpoolWork (
   PTP_WORK_CALLBACK pfnwk,
   PVOID pv,
   PTP_CALLBACK_ENVIRON pcbe)
```

- pfnwk is the pointer to the callback function
- pv is a value that is passed to every call to the callback function
- pcbe is the TP_CALLBACK_ENVIRON structure. Usually default will suffice.
- Return value is the thread pool work object. It is NULL in case of failure.

```
VOID SubmitThreadpoolWork (
   PTP_WORK pwk)
```

- This API is used to register a work object (i.e. A call back function and a parameter)
- The Windows Thread pool decides which and when to invoke the callback function of a work object
- pwk is the reference returned by the CreateThreadpoolWork() call
- The callback function associated with pwk will be executed once. The thread used to run the callback function is determined by the kernel scheduler.
- Programmer do not need to manage threads but synchronization still needs to be enforced.

```
VOID WaitForThreadpoolWorkCallbacks (
   PTP_WORK pwk,
   BOOL fCancelPendingCallbacks)
```

- The wait function does not have a timeout. It returns when all the callback functions have returned.

- Optionally it can also cancel the work objects whose call back function has not started as yet using the second parameter. Callbacks that have started will run to completion.
- pwk is the work thread pool work object.

```
VOID CALLBACK WorkCallback (
    PTP_CALLBACK_INSTANCE Instance,
    PVOID Context,
    PTP_WORK Work)
```

- Work is the work object and Context is pv value obtained from the work object creation call.
- The Instance is the callback instance that provide important information regarding the instance to enable kernel to schedule callbacks.
- CallbackMayRunLong informs the kernel that the callback instance may run for a longer period. Normally callback instances are expected to run for a short period.

Topic 70: Binary Search Using Heaps

The example is formulated using two heaps. One is a node heap and the other is a data heap. Node heap is used to build a tree, while data heap is used to store keys. Recursive functions are used for allocating nodes and scanning nodes as tree is a recursive structure. Data in file is read in a record and the key is used to lexically build a tree. The tree only contains the key entries. The file is ultimately sorted by an In-order traversal of the tree.

```
/* Chapter 5, sortBT command.   Binary Tree version. */
/* sort files
                Sort one or more files.
                This limited implementation sorts on the first field only.
                The key field length is assumed to be fixed length (8-characters).
                The data fields are varying length character strings. */
/* This program illustrates:
                1.      Multiple independent heaps; one for the sort tree nodes,
                   the other for the records.
                2.      Using HeapDestroy to free an entire data structure in a single operation.
                3.      Structured exception handling to catch memory allocation errors. */
/* Technique:
                1.      Scan the input file, placing each key (which is fixed size)
                   into the binary search tree and each record onto the data heap.
                2.      Traverse the search tree and output the records in order.
                3.      Destroy the heap and repeat for the next file. */

#include "Everything.h"
#define KEY_SIZE 8
                /* Structure definition for a tree node. */
typedef struct _TREENODE {
                struct _TREENODE *Left, *Right;
                TCHAR key[KEY_SIZE];
                LPTSTR pData;
} TREENODE, *LPTNODE, **LPPTNODE;
#define NODE_SIZE sizeof (TREENODE)
#define NODE_HEAP_ISIZE 0x8000
#define DATA_HEAP_ISIZE 0x8000
#define MAX_DATA_LEN 0x1000
#define TKEY_SIZE KEY_SIZE * sizeof (TCHAR)
#define STATUS_FILE_ERROR 0xE0000001        // Customer exception
LPTNODE FillTree (HANDLE, HANDLE, HANDLE);
BOOL Scan (LPTNODE);
int KeyCompare (LPCTSTR, LPCTSTR), iFile; /* for access in exception handler */
BOOL InsertTree (LPPTNODE, LPTNODE);
int _tmain (int argc, LPTSTR argv[])
{
```

```
            HANDLE hIn = INVALID_HANDLE_VALUE, hNode = NULL, hData =
NULL;
            LPTNODE pRoot;
            BOOL noPrint;
            CHAR errorMessage[256];
            int iFirstFile = Options (argc, argv, _T ("n"), &noPrint, NULL);
            if (argc <= iFirstFile)
                  ReportError (_T ("Usage: sortBT [options] files"), 1, FALSE);
                                    /* Process all files on the command line. */
for (iFile = iFirstFile; iFile < argc; iFile++) __try {
                                                      /* Open the input file. */
                  hIn = CreateFile (argv[iFile], GENERIC_READ, 0, NULL,
OPEN_EXISTING, 0, NULL);
                  if (hIn == INVALID_HANDLE_VALUE)
                        RaiseException (STATUS_FILE_ERROR, 0, 0, NULL);

                  __try {
                        /* Allocate the two growable heaps. */
                        hNode = HeapCreate (
HEAP_GENERATE_EXCEPTIONS | HEAP_NO_SERIALIZE, NODE_HEAP_ISIZE, 0);
                        hData = HeapCreate (
HEAP_GENERATE_EXCEPTIONS | HEAP_NO_SERIALIZE, DATA_HEAP_ISIZE, 0);
/* Process the input file, creating the tree. */
                        pRoot = FillTree (hIn, hNode, hData);
                        /* Display the tree in key order. */
                        if (!noPrint) {
                                    _tprintf (_T ("Sorted file: %s\n"), argv[iFile]);
                                    Scan (pRoot);
                        }
                  } __finally { /* Heaps and file handle are always closed */
                        /* Destroy the two heaps and data structures. */
                        if (hNode != NULL) HeapDestroy (hNode);
                        if (hData != NULL) HeapDestroy (hData);
                        hNode = NULL; hData = NULL;
                        if (hIn != INVALID_HANDLE_VALUE) CloseHandle (hIn);
                        hIn = INVALID_HANDLE_VALUE;
                  }
            } /* End of main file processing loop and try block. */
            /* Handle the exceptions that we can expect - Namely, file open error or out of
memory. */
            __except ( (GetExceptionCode() == STATUS_FILE_ERROR ||
GetExceptionCode() == STATUS_NO_MEMORY)
                                    ? EXCEPTION_EXECUTE_HANDLER :
EXCEPTION_CONTINUE_SEARCH)
            {
```

```
                    _stprintf (errorMessage, _T("\n%s %s"), _T("sortBT error on file:"),
argv[iFile]);
                    ReportError (errorMessage, 0, TRUE);
                }
                return 0;
}
LPTNODE FillTree (HANDLE hIn, HANDLE hNode, HANDLE hData)
/* Scan the input file, creating a binary search tree in the
                hNode heap with data pointers to the hData heap. */
/* Use the calling program's exception handler. */
{
                LPTNODE pRoot = NULL, pNode;
                DWORD nRead, i;
                BOOL atCR;
                TCHAR dataHold[MAX_DATA_LEN];
                LPTSTR pString;
                                                /* Open the input file. */
while (TRUE) {
                    pNode = HeapAlloc (hNode, HEAP_ZERO_MEMORY, NODE_SIZE);
                    pNode->pData = NULL;
                    (pNode->Left) = pNode->Right = NULL;
                                                /* Read the key. Return if done. */
                    if (!ReadFile (hIn, pNode->key, TKEY_SIZE,
                                        &nRead, NULL) || nRead != TKEY_SIZE)
                                                /* Assume end of file on error. All records
must be just the right size */
                        return pRoot;   /* Read the data until the end of line. */
                    atCR = FALSE;               /* Last character was not a CR. */
                    for (i = 0; i < MAX_DATA_LEN; i++) {
                        ReadFile (hIn, &dataHold[i], TSIZE, &nRead, NULL);
                        if (atCR && dataHold[i] == LF) break;
                        atCR = (dataHold[i] == CR);
}
                    dataHold[i - 1] = _T('\0');
                    /* dataHold contains the data without the key.
                        Combine the key and the Data. */
                    pString = HeapAlloc (hData, HEAP_ZERO_MEMORY,
                                        (SIZE_T)(KEY_SIZE + _tcslen (dataHold) + 1) *
TSIZE);
                    memcpy (pString, pNode->key, TKEY_SIZE);
                    pString[KEY_SIZE] = _T('\0');
                    _tcscat (pString, dataHold);
                    pNode->pData = pString;
                                                /* Insert the new node into the search tree. */
                    InsertTree (&pRoot, pNode);
} /* End of while (TRUE) loop */
```

```
                return NULL; /* Failure */
}
BOOL InsertTree (LPPTNODE ppRoot, LPTNODE pNode)
/* Insert the new node, pNode, into the binary search tree, pRoot. */
{
                if (*ppRoot == NULL) {
                        *ppRoot = pNode;
                        return TRUE;
                }
                if (KeyCompare (pNode->key, (*ppRoot)->key) < 0)
                        InsertTree (&((*ppRoot)->Left), pNode);
                Else
InsertTree (&((*ppRoot)->Right), pNode);
                return TRUE;
}
int KeyCompare (LPCTSTR pKey1, LPCTSTR pKey2)
/* Compare two records of generic characters.
                The key position and length are global variables. */
{
                return _tcsncmp (pKey1, pKey2, KEY_SIZE);
}
static BOOL Scan (LPTNODE pNode)
/* Scan and print the contents of a binary tree. */
{
                if (pNode == NULL)
                        return TRUE;
                Scan (pNode->Left);
                _tprintf (_T ("%s\n"), pNode->pData);
                Scan (pNode->Right);
                return TRUE;
}
```

Topic 71: Memory-Mapped Files

Memory Mapping

Dynamic memory is allocated from the paging file. The paging file is controlled by the Operating System's (OS) virtual memory management system. Also the OS controls the mapping of virtual address onto physical memory. Memory mapped files help to directly map virtual memory space onto a normal file.

Advantages of Memory Mapped IO

There is no need to invoke direct file Input Output (IO) operations. Any data structure placed in file will be available for later use as well. It is convenient and efficient to use in-memory algorithms for sorting, searching etc. Large files could be processed as if they are placed in

memory. File processing is faster than ReadFile() and WriteFile(). There is no need to manage buffers for repetitive operation on a file. This is more optimally done by OS. Multiple processes can share memory space by mapping their virtual memory space onto a file. For file operations, page file space is not needed.

Other considerations

Windows also use memory mapping while implementing Dynamic Link Libraries (DLLs) and loading & executing executable (EXE) files. It is strongly recommended to use SHE exception handling while dealing with memory mapped file to look for EXCEPTION_IN_PAGE_ERROR exceptions.

Topic 72: File Mapping Objects

File Mapping Objects

In order to perform memory mapped file IO operations, file mapping objects need to be created. This object uses the file handle of an open file. The open file or part of the file is mapped onto the address space of the process. File mapping objects are assigned names so that they are also available to other processes. Moreover, these mapping objects also require protection and security attributes and a size. The API used for this purpose is CreateFileMapping().

HANDLE CreateFileMapping(HANDLE hFile,

LPSECURITY_ATTRIBUTES lpFileMappingAttributes,

DWORD flProtect,

DWORD dwMaximumSizeHigh,

DWORD dwMaximumSizeLow, LPCTSTR lpName);

hFile is the open handle to file compatible with protection flag dwProtect

INVALID_HANDLE_VALUE refers to the paging file

LPSECURITY_ATTRIBUTES allow the mapping object to be secured

dwProtect specifies the mapping file access with following attributes

PAGE_READONLY - It means that page can only be read within the mapped region. It can neither be written nor executed. hFile must have GENERIC_READ access.

PAGE_READWRITE - Provides full access to object if the hFile has GENERIC_READ and GENERIC_WRITE access.

PAGE_WRITECOPY - Means that when a mapped region changes, a private copy is written to the paging file and not to the original file.

dwMaximumSizeHigh and dwMaximumSizeLow specify the size of the mapping object. If set to 0, then the current file size is used. Carefully specify this size in the following cases:

· If the file size is expected to grow, then use the expected file size.

· Do not map a region beyond this limit. Once the size is assigned the mapping region cannot grow.

· An attempt to create a mapping on a zero length file will fail.

· The mapping size needs to be specified in the form of two 32-bit values rather than one 64-bit value.

· lpMapName is the name of the map that can also be used by other processes. Set this to NULL if you do not mean to share the map.

Topic 73: Open Existing File Mapping Objects

Open Existing File Mapping Objects

Previously, we discussed that a file mapping can be assigned a shared name by using CreateFileMapping(). This shared name can be used to open existing file maps using OpenFileMapping(). A file map created by a certain process can be subsequently used by other processes by referring to the object by name. The operation may fail if the name does not exist.

HANDLE OpenFileMapping(DWORD dwDesiredAccess,

BOOL bInheritHandle,

LPCSTR lpMapName);

dwDesiredAccess is checked against the accessed assigned by CreateFileMapping();

lpMapName is the name of the mapping assigned by CreateFileMapping().

Topic 74: Mapping Objects to Process Address Space

Mapping into Process Space

Previously, file mapping was created or a handle to already created mapping handle was obtained. The next step is to assign a process address space to file mapping. In case of heaps, first heap was created and then HeapAlloc() was used to allocate space within heap. Similarly once the file mapping is created, MapViewOfFile() is used to define file view block.

LPVOID MapViewOfFile(HANDLE hFileMappingObject,

DWORD dwDesiredAccess,

DWORD dwFileOffsetHigh,

DWORD dwFileOffsetLow,

SIZE_T dwNumberOfBytesToMap);

hFileMappingObject is file mapping object previously obtained by CreateFileMapping().

dwDesiredAccess should be compatible with access rights of file mapping object. The three flag commonly used are:

FILE_MAP_WRITE

FILE_MAP_READ

FILE_MAP_ALL_ACCESS

dwFileOffsetHigh and dwFileOffsetLow give the starting address of the file from where the mapping starts. To start the mapping from the start of a file, set both as zero. This value should be specified in multiples of 64K.

dwNumberOfBytesToMap shows the number of bytes of file to map. Set as zero to map the whole file.

If the function is successful it returns the base address of the mapped region. If the function fails, the return value is NULL.

Closing Mapping Handle

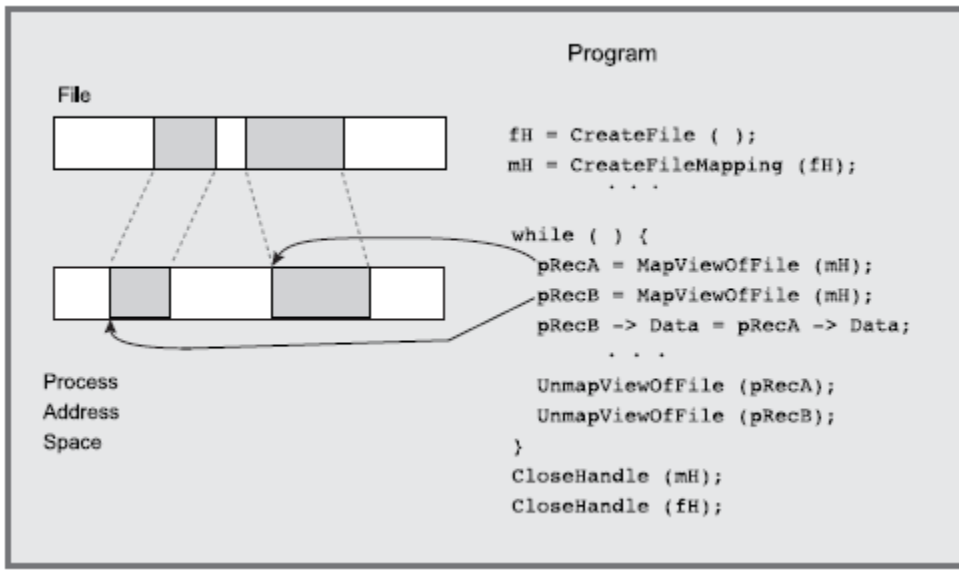
As it is necessary to release Heap blocks with HeapFree(), it is also necessary to unmap file views. File views are unmapped using UnmapViewOfFile().

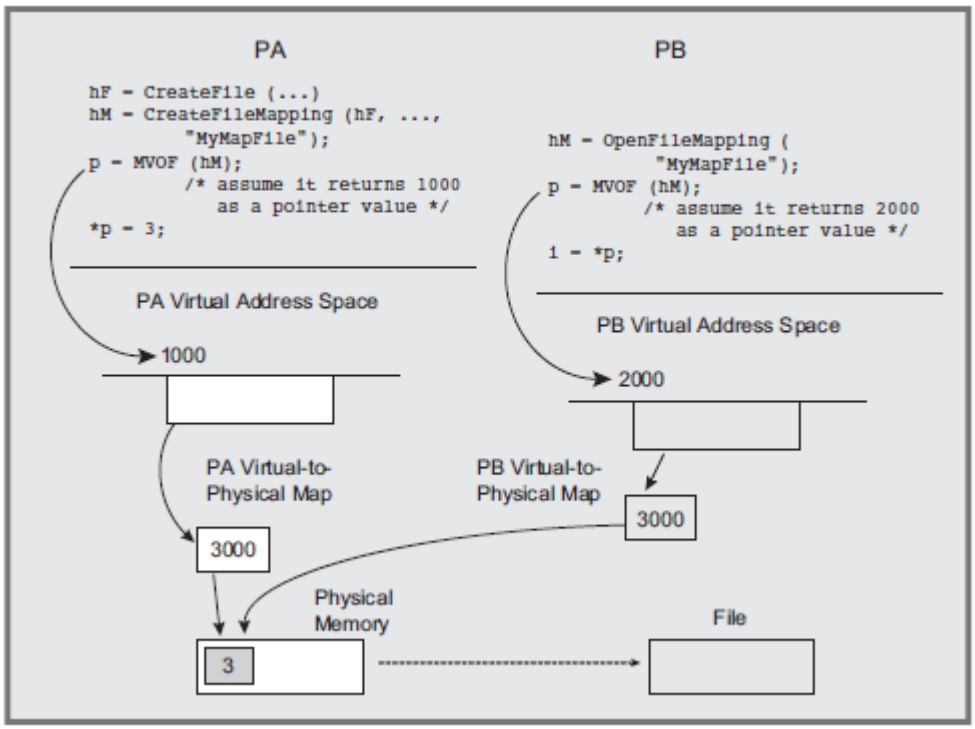
BOOL UnmapViewOfFile(LPCVOID lpBaseAddress);

lpBaseAddress is the pointer to the base address of the mapped view. If the function fails, the return value is zero.

Flushing File View

The file view can be flushed using the FlushViewOfFile() API. This will force the OS to writeback the dirty pages of the file on to disk. In case two processes try to access a file at a time such that one uses file mapping and other uses ReadFile() and WriteFile(). Then both processes may not receive the same view. Changes made through file maps might still be in memory and may not be accessible through ReadFile or WriteFile unless they are flushed. To get a uniform view, it is necessary that all the processes use file maps.





Topic 75: More About File Making

File Mapping Limitations

In Win32, it is not possible to map files bigger than 2-3 GB. Also the entire 3GB might not be available for merely file space. The above limitation is alleviated in Win64. File mapping cannot be extended. You need to know the size of a map in advance. Customized functions would be required to allocate memory within the mapped region.

File Mapping Procedure

The following minimum steps need to be taken while working with mapped files:

· Open File with at least GENERIC_READ access.

· If the file is new then set its length as some non-zero value using SetFilePointerEx() followed by SetEndOfFile().

· Map the file using CreateFileMapping() or OpenFileMapping ().

· Create one or more map views using MapViewOfFile().

· Access file through memory references.

· Unmap and then again Map file view to change regions.

· Use SEH to catch EXCEPTIO_IN_PAGE_ERROR exceptions.

· In the end, unmap file view with UnmapViewOfFile() and use CloseHandle() to close map and file handles.

Topic 76: Sequential File Access Using file mapping

Sequential File Access

Accessing a file through file mapping presents visible advantages. Although the setting up of fileviews might be programmatically complex, the advantages are far bigger. The processing time may reduce 3 folds as compared to conventional file operations while dealing with sequential files. These advantages may only seem to disappear if the size of input and output files is too large. The example is a simple Ceasar cipher application. It sequentially processes all the characters with the file. It simply substitutes each character by shifting it a few places in the ASCII set.

```
/* Chapter 5.
cci_fMM.c function: Memory Mapped implementation of the
simple Caeser cipher function. */
#include "Everything.h"
BOOL cci_f (LPCTSTR fIn, LPCTSTR fOut, DWORD shift)
/* Caesar cipher function.
*       fIn:                    Source file pathname.
*       fOut:                   Destination file pathname.
*       shift:                  Numeric shift value */
{
BOOL complete = FALSE;
                HANDLE hIn = INVALID_HANDLE_VALUE, hOut =
INVALID_HANDLE_VALUE;
                HANDLE hInMap = NULL, hOutMap = NULL;
                LPTSTR pIn = NULL, pInFile = NULL, pOut = NULL, pOutFile = NULL;
                __try {
                        LARGE_INTEGER fileSize;
                        /* Open the input file. */
                        hIn = CreateFile (fIn, GENERIC_READ, 0, NULL,
                                OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL, NULL);
                        if (hIn == INVALID_HANDLE_VALUE)
                                ReportException (_T ("Failure opening input file."), 1);
/* Get the input file size. */
                        if (!GetFileSizeEx (hIn, &fileSize))
                                ReportException (_T ("Failure getting file size."), 4);
                        /* This is a necessar, but NOT sufficient, test for mappability on 32-bit
systems S
                         * Also see the long comment a few lines below */
                        if (fileSize.HighPart > 0 && sizeof(SIZE_T) == 4)
                                ReportException (_T ("This file is too large to map on a Win32
system."), 4);
                        /* Create a file mapping object on the input file. Use the file size. */
                        hInMap = CreateFileMapping (hIn, NULL, PAGE_READONLY, 0, 0,
NULL);
                        if (hInMap == NULL)
                                ReportException (_T ("Failure Creating input map."), 2);
/* Map the input file */
                        /* Comment: This may fail for large files, especially on 32-bit systems
                         * where you have, at most, 3 GB to work with (of course, you have much
less
                         * in reality, and you need to map two files.
                         * This program works by mapping the input and output files in their
entirety.
                         * You could enhance this program by mapping one block at a time for
each file,
```

```
                        * much as blocks are used in the ReadFile/WriteFile implementations.
This would
                        * allow you to deal with very large files on 32-bit systems. I have not
taken
                        * this step and leave it as an exercise.
                        */
                        pInFile = MapViewOfFile (hInMap, FILE_MAP_READ, 0, 0, 0);
                        if (pInFile == NULL)
                                ReportException (_T ("Failure Mapping input file."), 3);
/*  Create/Open the output file. */
                        /* The output file MUST have Read/Write access for the mapping to
succeed. */
                        hOut = CreateFile (fOut, GENERIC_READ | GENERIC_WRITE,
                                0, NULL, CREATE_ALWAYS, FILE_ATTRIBUTE_NORMAL,
NULL);
                        if (hOut == INVALID_HANDLE_VALUE) {
                                complete = TRUE; /* Do not delete an existing file. */
                                ReportException (_T ("Failure Opening output file."), 5);
                        }
                        /* Map the output file. CreateFileMapping will expand
                        the file if it is smaller than the mapping. */
                        hOutMap = CreateFileMapping (hOut, NULL, PAGE_READWRITE,
fileSize.HighPart, fileSize.LowPart, NULL);
                        if (hOutMap == NULL)
ReportException (_T ("Failure creating output map."), 7);
                        pOutFile = MapViewOfFile (hOutMap, FILE_MAP_WRITE, 0, 0,
(SIZE_T)fileSize.QuadPart);
                        if (pOutFile == NULL)
                                ReportException (_T ("Failure mapping output file."), 8);
                        /* Now move the input file to the output file, doing all the work in
memory. */
                        __try
                        {
                                CHAR cShift = (CHAR)shift;
                                pIn = pInFile;
                                pOut = pOutFile;
                                while (pIn < pInFile + fileSize.QuadPart) {
                                        *pOut = (*pIn + cShift);
                                        pIn++; pOut++;
}
                                complete = TRUE;
                        }
                        __except(GetExceptionCode() == EXCEPTION_IN_PAGE_ERROR ?
EXCEPTION_EXECUTE_HANDLER : EXCEPTION_CONTINUE_SEARCH)
                        {
                                complete = FALSE;
```

```
                        ReportException(_T("Fatal Error accessing mapped file."), 9);
                    }
                    /* Close all views and handles. */
                    UnmapViewOfFile (pOutFile); UnmapViewOfFile (pInFile);
                    CloseHandle (hOutMap); CloseHandle (hInMap);
                    CloseHandle (hIn); CloseHandle (hOut);
                    return complete;

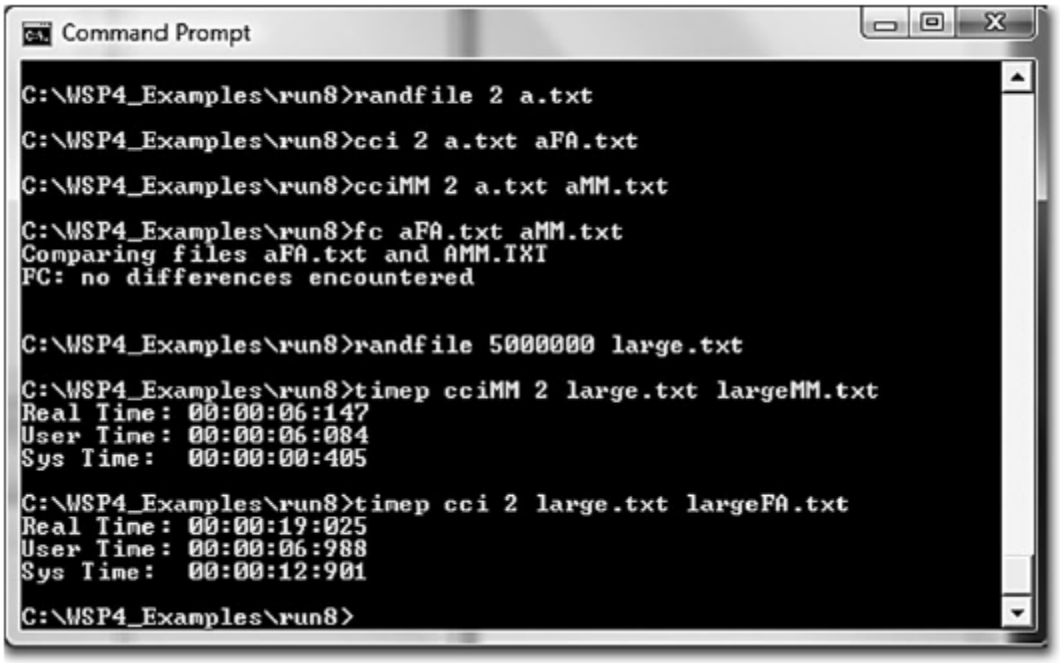
}
            __except (EXCEPTION_EXECUTE_HANDLER) {
                    if (pOutFile != NULL) UnmapViewOfFile (pOutFile); if (pInFile !=
NULL) UnmapViewOfFile (pInFile);
                    if (hOutMap != NULL) CloseHandle (hOutMap); if (hInMap != NULL)
CloseHandle (hInMap);
                    if (hIn != INVALID_HANDLE_VALUE) CloseHandle (hIn); if (hOut !=
INVALID_HANDLE_VALUE) CloseHandle (hOut);
                    /* Delete the output file if the operation did not complete successfully. */
                    if (!complete)
                        DeleteFile (fOut);
                    return FALSE;               }
}
```

Topic 77: Sorting a Memory-Mapped File

Content:

Another advantage of memory mapping is the ability to use convenient memory based algorithms to process files. Sorting data in memory, for instance, is much easier than sorting records in a file.

Program explained in topic 77 sorts a file with fixed-length records. This program, called sortFL, is similar to Program explaining example of sorting with binary search tree that it assumes an 8-byte sort key at the start of the record, but this example is restricted to fixed records.



The sorting is performed by the ***<stdlib.h>*** C library function ***qsort***. Notice that ***qsort*** requires a programmer-defined record comparison function.

This program structure is straightforward. Simply create the file mapping on a temporary copy of the input file, create a single view of the file, and invoke qsort. There is no file I/O. Then the sorted file is sent to standard output using ***_tprintf***, although a null character is appended to the file map.

Exception and error handling are omitted in the listing but are in the Examples solution on the recommended book's Website.

Program: Sorting a File with Memory Mapping

```
#include<everything.h>
typedef struct  _RECORD
{
        TCHAR key[KEY_SIZE];
        TCHAR data[DATALEN]
} RECORD;
```

```
#define RECSIZE sizeof (RECORD)

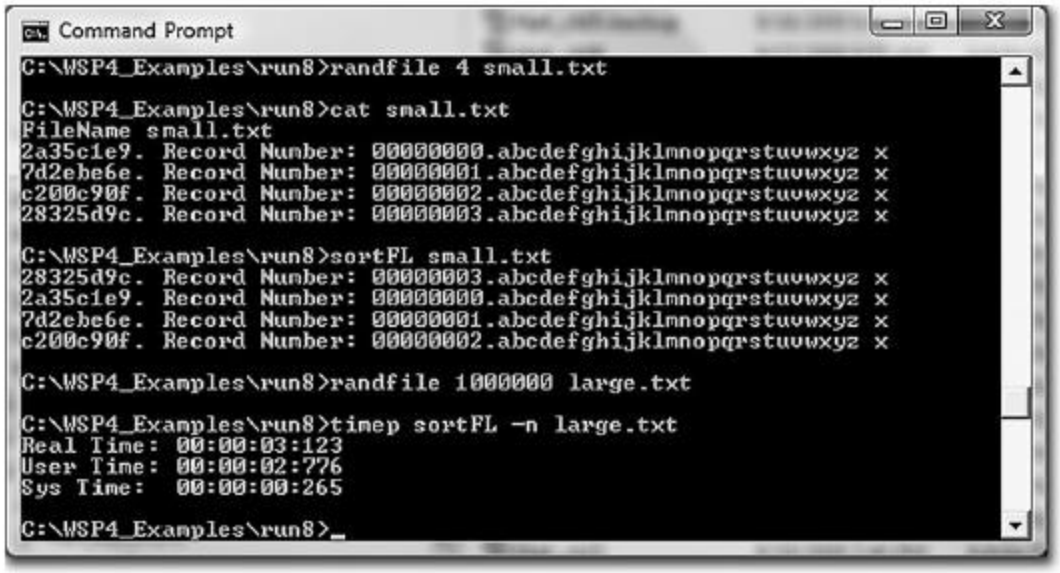
int _tmain (int argc, LPTSTR argv[])
{
   HANDLE hFile = INVALID_HANDLE_VALUE, hMap = NULL;
   LPVOID pFile = NULL;
   LARGE_INTEGER fileSize;
   TCHAR tempFile[MAX_PATH];
   LPTSTR pTFile;

   /* Create the name for a temporary file to hold a copy of
      the file to be sorted. Sorting is done in the temp file.
   _stprintf (tempFile, _T ("%s.tmp"), argv[1]);
   CopyFile (argv[1], tempFile, TRUE);

   /* Map the temporary file and sort it in memory. */
   hFile = CreateFile (tempFile, GENERIC_READ | GENERIC_WRITE,
         0, NULL, OPEN_EXISTING, 0, NULL);
   GetFileSizeEx (hFile, &fileSize);
   fileSize.QuadPart += 2;
   hMap = CreateFileMapping (hFile, NULL, PAGE_READWRITE,
         fileSize.HighPart, fileSize.LowPart, NULL);
   pFile = MapViewOfFile (hMap, FILE_MAP_ALL_ACCESS, 0, 0, 0);

   qsort (pFile, FsLow / RECSIZE, RECSIZE, KeyCompare);
                          /* KeyCompare is as in Program 5-2. '

   /* Print the sorted file. */
   pTFile = (LPTSTR) pFile;
   pTFile[fileSize.QuadPart/TSIZE] = '\0';
   _tprintf (_T ("%s"), pFile);
   UnmapViewOfFile (pFile);
   CloseHandle (hMap);
   CloseHandle (hFile);
   DeleteFile (tempFile);
   return 0;
}
```



Topic 78: Using Base Pointer

Content:

File maps are convenient, as the preceding examples demonstrate. Suppose, however, that the program creates a data structure with pointers in a mapped file and expects to access that file in the future. Pointers will all be relative to the virtual address returned from MapViewOfFile, and they will be meaningless when mapping the file the next time. The solution is to use based pointers, which are actually offsets relative to another pointer. The Microsoft C syntax, available in Visual C++ and some other systems, is:

type _based (base) declarator

Here are two examples.

LPTSTR pInFile = NULL;

DWORD _based (pInFile) *pSize;

TCHAR _based (pInFile) *pIn;

Notice that the syntax forces use of the *, a practice that is contrary to Windows convention but which the programmer could easily fix with a typedef.

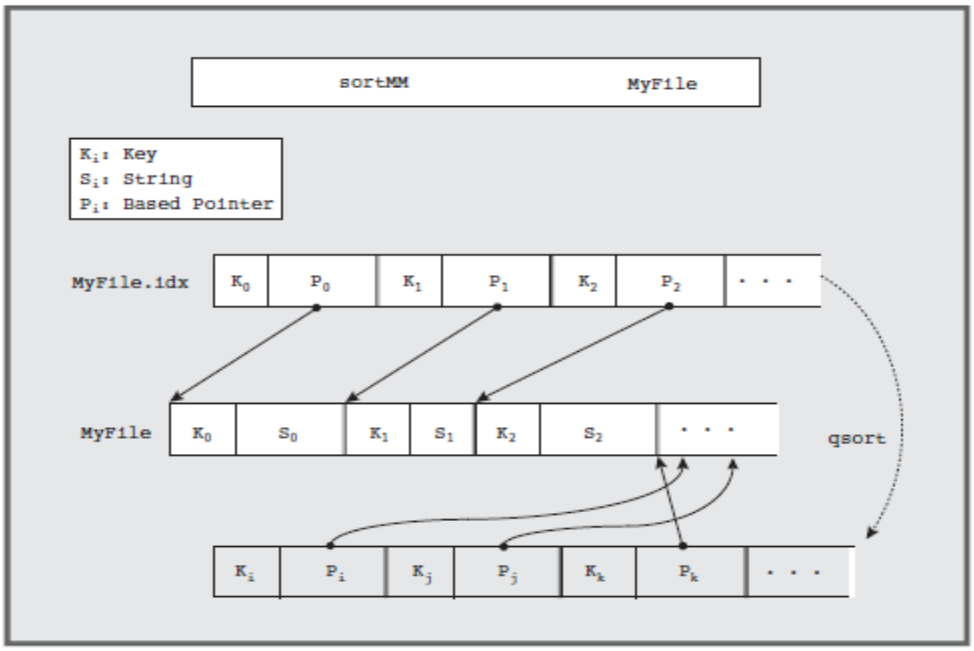
Topic 79: Base Pointer Example

Previously we developed a program that sorted record stored in memory mapped file. Sorting huge files each time they are accessed is not feasible, so permanent indexes are maintained on required key.

Using the address returned by MapViewOfFile() for maintaining indexes is meaningless as the address is liable to change in each call to API. A simple methodology is to maintain an array of record and then build an index for the records. And subsequently use the index to access records directly.

Following example program implements this concept in the given way:

The program uses record of varying sizes in a file. It uses the first field of each record as the key of 8 characters. There are two file mapping. One mapping maps the original file and the other maps the index file. Each record in index file contains a key and the pointer location into the original file for the record containing that key. Once index file is created it can be easily used later. Subsequently, index file records can be sorted for faster searching. The input file remains unchanged. Pictorial Representation of the example is attached below;



Topic 80: Dynamic Link Library

We have previously seen the example use of memory mapped files in windows. This a fundamental feature of windows. Windows itself uses this feature while working with Dynamic Link Libraries (DLLs). DLLs are one of the most important components of windows on which many high-level technologies depend like COM.

Static Linking

The conventional approach is to gather all the source code and library functions attach them and encapsulate them into a single executable file. This approach is simple but has few disadvantages.

- The executable image will be large as it contains all library functions. Hence it will consume more disk space and will require large physical memory to run.
- If a library function updates the whole program will require recompilation.
- There can be many programs that require a library function. Each program will have static copy of its own. Hence, resources requirement will increase.
- It will reduced portability as a program compiled with certain environment setting will run same functions in different environment where some other version might be.

How DLLs Solve this Problem

Using DLLs the library functions are not linked at compile time. They are linked at program load time (implicit linking) or at run time (explicit linking).

As a result the size of executable package is smaller. DLLs can be easily used to create shared libraries which can be used by multiple programs concurrently. Only a single copy of shared DLLs is placed in memory. All the processes sharing the DLL map the DLL space onto their program space. Each program will have its own copy of DLL global variables.

New versions or updates can be simply supported by just providing a new DLL without the need of recompiling main code. The library runs in the same processes as the calling the program.

Importance of DLLs

DLLs are used in almost all modern operating systems. DLLs are most important in case of windows as they are used to implement OS interfaces. The entire Windows API is supported by a set of DLLs which are invoked to call kernel services. The DLL code can be shared by multiple processes. DLL function when invoked by a process runs in process space. Therefore, it can use resources of the calling process such as file handles and thread stack. DLLs should be written in Thread-safe manner. A DLL exports variables as well as function entry points.

Topic 81: Implicit Linking

Implicit linking is the easier of the two techniques. Functions defined in a DLL are collected and build as DLL. The build process builds a .LIB file which is a stub for actual code. The stub is linked to the calling program at build time. It provides a place holder for each function in the DLL.

The place holder/stub will call the original function in the DLL. This file should be placed in common user library directory for the project. The build process also constructs the DLL that contain the original binary image for the functions. This File is usually placed in the same directory as the application.

Function interfaces defined in DLLs should be exported carefully.

Content Development

Topic No - 96 (Waiting for a Process)

Content:

The synchronization can be attained easily by the wait process. Windows provides a general-purpose wait function. Which can be used to wait for a single object and also multiple objects in a group. Windows send a signal to the waiting process when the process terminates

We have two API's

i)DWORD waitForSingleObject(HANDLE hHandle, DWORD dwMilliseconds);

ii)DWORD waitForMultipleObjects(DWORD nCount, const HANDLE*lpHandles, BOOL bWaitAll,DWORD dwMilliseconds);

hHandle is the single object for which the process waits.

lpHandle is the array of handles for which the process waits.

nCount is the number of objects in an array. Should not exceed MAXIMUM WAIT OBJECTS

dwMilliseconds is the timeout period for wait . 0 for no wait and INFINITE FOR indefinite wait.

bWaitAll describes if it's necessary to wait for all the objects to get free.

The possible return values are

WAIT_OBJECT_0

WAIT_OBJECT_0+n

WAIT_TIMEDOUT

WAIT_FAILED

WAIT_ABANDONED_0

Topic No - 97 (Environment Block)

Content:

Environment Block string

-An Environment Block is associated with each process.

-The EB contains string regarding the environment of the process of the form Name = Value

-Each string is NULL-terminated

-PATH is an example of an environment string.

-Environment variables can get and set using these API's

DWORD GetEnvironmentVatiable (LPCTSTR lpNAME,LpTSTR lpBuffer, DWORD nSize);

And

BOOL SetEnviromentVariable(LPCTSTR ipName, LPCTSTR lpValue);

To share the parent environment with the child process set lpEnvironment to NULL in the call to CreateProcess()

Any process can modify the environment variables and make new ones

lpName is the variable name. On setting the string the value is modified if the variable already exists. If it does not exist

then a new variable is created and assigned the value. IF the value is NULL then the variable is deleted.

"=" si reserved and cannot be used in the variable name.

GetEnvironmentVariable() return the length of variable string or 0 in case of failure.

lf LpValue is not as long as the value specified by the count then the actual length of the string is returned.

The access rights given in Create Process () are usually

Process_All_Access. But there are several options

PROCESS_QUERY_INFORMATION

CREATE_PROCESS

PROCESS_TERMINATE

PROCESS_SET_INFORMATION

DUPLICATE_HANDLE

CREATE_HANDLE

it can be useful to limit some processes right like giving PROCESS_TERMINATE right to parent process only.

Topic No - 98 (A Pattern Searching Example)

Content:

A pattern searching example

This example uses the power of windows multitasking to search a specific pattern among numerous files.

The process take the specific pattern along with filenames through command line.

The standard output file is specified as inheritable in new process start-up info structure.

Wait functions are used for synchronizations

As soon as the search end the results are displayed one at a time.

Wait functions are limited for 64 handles so program works accordingly.

The program uses the exit code to identify whether the process has detected a match or not.

A Pattern Searching Example

The Example

```
#include "Everything.h"

int _tmain (int argc, LPTSTR argv[])

/*      Create a separate process to search each file on the

        command line. Each process is given a temporary file,

        in the current directory, to receive the results. */

{

        HANDLE hTempFile;

        SECURITY_ATTRIBUTES stdOutSA = /* SA for inheritable handle. */
```

```
                        {sizeof (SECURITY_ATTRIBUTES), NULL, TRUE};
TCHAR commandLine[MAX_PATH + 100];
        STARTUPINFO startUpSearch, startUp;
        PROCESS_INFORMATION processInfo;
        DWORD exitCode, dwCreationFlags = 0;
        int iProc;
        HANDLE *hProc;  /* Pointer to an array of proc handles. */
        typedef struct {TCHAR tempFile[MAX_PATH];} PROCFILE;
        PROCFILE *procFile; /* Pointer to array of temp file names. */
#ifdef UNICODE
   dwCreationFlags = CREATE_UNICODE_ENVIRONMENT;
#endif
if (argc < 3)
                ReportError (_T ("Usage: grepMP pattern files."), 1, FALSE);
        /* Startup info for each child search process as well as
                the child process that will display the results. */
        GetStartupInfo (&startUpSearch);
        GetStartupInfo (&startUp);
        /* Allocate storage for an array of process data structures,
                each containing a process handle and a temporary file name. */
procFile = malloc ((argc - 2) * sizeof (PROCFILE));
        hProc = malloc ((argc - 2) * sizeof (HANDLE));
        /* Create a separate "grep" process for each file on the
                command line. Each process also gets a temporary file
```

```
                name for the results; the handle is communicated through
                the STARTUPINFO structure. argv[1] is the search pattern. */
        for (iProc = 0; iProc < argc - 2; iProc++) {
                /* Create a command line of the form: grep argv[1] argv[iProc + 2] */
/* Allow spaces in the file names. */
                _stprintf (commandLine, _T ("grep \"%s\" \"%s\""), argv[1], argv[iProc + 2]);
                /* Create the temp file name for std output. */
                if (GetTempFileName (_T ("."), _T ("gtm"), 0, procFile[iProc].tempFile) == 0)
                        ReportError (_T ("Temp file failure."), 2, TRUE);
                /* Set the std output for the search process. */
hTempFile = /* This handle is inheritable */
                        CreateFile (procFile[iProc].tempFile,
                                /** GENERIC_READ | Read access not required **/ GENERIC_WRITE,
                                FILE_SHARE_READ | FILE_SHARE_WRITE, &stdOutSA,
                                CREATE_ALWAYS, FILE_ATTRIBUTE_NORMAL, NULL);
                if (hTempFile == INVALID_HANDLE_VALUE)
                        ReportError (_T ("Failure opening temp file."), 3, TRUE);
                /* Specify that the new process takes its std output
                        from the temporary file's handles. */
startUpSearch.dwFlags = STARTF_USESTDHANDLES;
                startUpSearch.hStdOutput = hTempFile;
                startUpSearch.hStdError = hTempFile;
                startUpSearch.hStdInput = GetStdHandle (STD_INPUT_HANDLE);
                /* Create a process to execute the command line. */
```

```
                if (!CreateProcess (NULL, commandLine, NULL, NULL,
                                TRUE, dwCreationFlags, NULL, NULL, &startUpSearch, &processInfo))
                        ReportError (_T ("ProcCreate failed."), 4, TRUE);
                /* Close unwanted handles */
                CloseHandle (hTempFile); CloseHandle (processInfo.hThread);
/* Save the process handle. */
                hProc[iProc] = processInfo.hProcess;
        }
        /* Processes are all running. Wait for them to complete, then output
                the results - in the order of the command line file names. */
        for (iProc = 0; iProc < argc-2; iProc += MAXIMUM_WAIT_OBJECTS)
                WaitForMultipleObjects (min(MAXIMUM_WAIT_OBJECTS, argc - 2 - iProc),
                                &hProc[iProc], TRUE, INFINITE);
/* Result files sent to std output using "cat".
                Delete each temporary file upon completion. */
        for (iProc = 0; iProc < argc - 2; iProc++) {
                if (GetExitCodeProcess (hProc[iProc], &exitCode) && exitCode == 0) {
                        /* Pattern was detected - List results. */
                        /* List the file name if there is more than one file to search */
                        if (argc > 3) _tprintf (_T("%s:\n"), argv[iProc+2]);
                        _stprintf (commandLine, _T ("cat \"%s\""), procFile[iProc].tempFile);
if (!CreateProcess (NULL, commandLine, NULL, NULL,
                                TRUE, dwCreationFlags, NULL, NULL, &startUp, &processInfo))
                                ReportError (_T ("Failure executing cat."), 0, TRUE);
```

```
                else {
                        WaitForSingleObject (processInfo.hProcess, INFINITE);
                        CloseHandle (processInfo.hProcess);
                        CloseHandle (processInfo.hThread);
                }
        }
CloseHandle (hProc[iProc]);
        if (!DeleteFile (procFile[iProc].tempFile))
                ReportError (_T ("Cannot delete temp file."), 6, TRUE);
    }
    free (procFile); free (hProc);
    return 0;
}
```

Topic No - 99 (Working in Multiprocessor Environment)

Content:

Working in Multiprocessor Environment

Multiprocessor Environment

All the processes run concurrently in windows environment.

If the system is uniprocessor the processor time is multiplexed among multiple processes in an interleaved manner

If the system is multiprocessor then windows scheduler can run process threads on separate processors.

There would be substantial performance gain in this case

The performance gain will not be linear because of dependencies among processes (wait and signal)

From programming point of view it is essential to understand this potential of windows so that programs can be designed optimally.

Alternately, constraining the processing to a specific processor is also possible.

This is accomplished by processor affinity mask

Subsequently, it is possible to create independent threads within a process which can be scheduled on separate processor.

Topic No - 100 (Process Execution Times)

Process Execution Times

Process Times

Windows API provides a very simple mechanism for determining the amount of time a process has consumed.

The function used for this purpose is GetProcessTime()

BOOL GetProcessTimes(HANDLE hProcess,

LPFILETIME lpCreationTime,

LPFILETIME lpExitTime,

LPFILETIME lpKernelTime,

LPFILETIME lpUserTime);

Process handle can be of the current process or a terminated one.

Elapsed time can be computed by subtracting creation time from exit time.

FILETIME is a 64-bit value

GetThreadTime() can be used similarly and requires a hread handle.

Topic No - 101 (Example of Process Execution Times)

Content:

Example of Process Execution Times

Example

This example prints the elapsed, real, and user times.

It uses the API GetCommandLine()to get the command line as a single string

It then uses the SkipArg() function the skip past the executable name.

Process Execution Times example

#include "Everything.h"

```
int _tmain (int argc, LPTSTR argv[])

{

        STARTUPINFO startUp;

        PROCESS_INFORMATION procInfo;

        union {             /* Structure required for file time arithmetic. */

                LONGLONG li;

                FILETIME ft;

        } createTime, exitTime, elapsedTime;



FILETIME kernelTime, userTime;

        SYSTEMTIME elTiSys, keTiSys, usTiSys;

        LPTSTR targv, cLine = GetCommandLine ();

        OSVERSIONINFO windowsVersion;

        HANDLE hProc;



        targv = SkipArg(cLine, 1, argc, argv);

        /*  Skip past the first blank-space delimited token on the command line */

        if (argc <= 1 || NULL == targv)

                ReportError (_T("Usage: timep command ..."), 1, FALSE);

/* Determine is this is Windows 2000 or NT.  */
```

```
        windowsVersion.dwOSVersionInfoSize = sizeof(OSVERSIONINFO);
        if (!GetVersionEx (&windowsVersion))
                ReportError (_T("Can not get OS Version info. %d"), 2, TRUE);

        if (windowsVersion.dwPlatformId != VER_PLATFORM_WIN32_NT)
                ReportError (_T("This program only runs on Windows NT kernels"), 2, FALSE);

/* Determine is this is Windows 2000 or NT.  */
        windowsVersion.dwOSVersionInfoSize = sizeof(OSVERSIONINFO);
        if (!GetVersionEx (&windowsVersion))
                ReportError (_T("Can not get OS Version info. %d"), 2, TRUE);

        if (windowsVersion.dwPlatformId != VER_PLATFORM_WIN32_NT)
                ReportError (_T("This program only runs on Windows NT kernels"), 2, FALSE);
GetStartupInfo (&startUp);

        /* Execute the command line and wait for the process to complete. */
        if (!CreateProcess (NULL, targv, NULL, NULL, TRUE,
                NORMAL_PRIORITY_CLASS, NULL, NULL, &startUp, &procInfo))
                ReportError (_T ("\nError starting process. %d"), 3, TRUE);
        hProc = procInfo.hProcess;
```

```
        if (WaitForSingleObject (hProc, INFINITE) != WAIT_OBJECT_0)

ReportError (_T("Failed waiting for process termination. %d"), 5, TRUE);;

        if (!GetProcessTimes (hProc, &createTime.ft,
                &exitTime.ft, &kernelTime, &userTime))
                        ReportError (_T("Can not get process times. %d"), 6, TRUE);

        elapsedTime.li = exitTime.li - createTime.li;
        FileTimeToSystemTime (&elapsedTime.ft, &elTiSys);
        FileTimeToSystemTime (&kernelTime, &keTiSys);
        FileTimeToSystemTime (&userTime, &usTiSys);
        _tprintf (_T ("Real Time: %02d:%02d:%02d.%03d\n"),

elTiSys.wHour, elTiSys.wMinute, elTiSys.wSecond,
                elTiSys.wMilliseconds);
        _tprintf (_T ("User Time: %02d:%02d:%02d.%03d\n"),
                usTiSys.wHour, usTiSys.wMinute, usTiSys.wSecond,
                usTiSys.wMilliseconds);
        _tprintf (_T ("Sys Time:  %02d:%02d:%02d.%03d\n"),
                keTiSys.wHour, keTiSys.wMinute, keTiSys.wSecond,
```

```
                keTiSys.wMilliseconds);


        CloseHandle (procInfo.hThread); CloseHandle (procInfo.hProcess);

        return 0;

}
```

Topic No - 102 (Generating Console events)

Content:

Example of Process Execution Times

Example

This example prints the elapsed, real and user times.

It uses the API GetCommandLine()to get the command line as a single string

It than uses the SkipArg() function the skip past the executable name.

Process Execution Times example

```
#include "Everything.h"


int _tmain (int argc, LPTSTR argv[])

{
```

```
        STARTUPINFO startUp;
        PROCESS_INFORMATION procInfo;
        union {             /* Structure required for file time arithmetic. */
                LONGLONG li;
                FILETIME ft;
        } createTime, exitTime, elapsedTime;


FILETIME kernelTime, userTime;
        SYSTEMTIME elTiSys, keTiSys, usTiSys;
        LPTSTR targv, cLine = GetCommandLine ();
        OSVERSIONINFO windowsVersion;
        HANDLE hProc;


        targv = SkipArg(cLine, 1, argc, argv);
        /*  Skip past the first blank-space delimited token on the command line */
        if (argc <= 1 || NULL == targv)
                ReportError (_T("Usage: timep command ..."), 1, FALSE);
/* Determine is this is Windows 2000 or NT.  */
        windowsVersion.dwOSVersionInfoSize = sizeof(OSVERSIONINFO);
        if (!GetVersionEx (&windowsVersion))
                ReportError (_T("Can not get OS Version info. %d"), 2, TRUE);


        if (windowsVersion.dwPlatformId != VER_PLATFORM_WIN32_NT)
                ReportError (_T("This program only runs on Windows NT kernels"), 2, FALSE);
```

```
/* Determine is this is Windows 2000 or NT.  */
        windowsVersion.dwOSVersionInfoSize = sizeof(OSVERSIONINFO);
        if (!GetVersionEx (&windowsVersion))
                ReportError (_T("Can not get OS Version info. %d"), 2, TRUE);

        if (windowsVersion.dwPlatformId != VER_PLATFORM_WIN32_NT)
                ReportError (_T("This program only runs on Windows NT kernels"), 2, FALSE);
GetStartupInfo (&startUp);

        /* Execute the command line and wait for the process to complete. */
        if (!CreateProcess (NULL, targv, NULL, NULL, TRUE,
                NORMAL_PRIORITY_CLASS, NULL, NULL, &startUp, &procInfo))
                ReportError (_T ("\nError starting process. %d"), 3, TRUE);
        hProc = procInfo.hProcess;
        if (WaitForSingleObject (hProc, INFINITE) != WAIT_OBJECT_0)

ReportError (_T("Failed waiting for process termination. %d"), 5, TRUE);;

        if (!GetProcessTimes (hProc, &createTime.ft,
                &exitTime.ft, &kernelTime, &userTime))
                        ReportError (_T("Can not get process times. %d"), 6, TRUE);

        elapsedTime.li = exitTime.li - createTime.li;
```

```
        FileTimeToSystemTime (&elapsedTime.ft, &elTiSys);
        FileTimeToSystemTime (&kernelTime, &keTiSys);
        FileTimeToSystemTime (&userTime, &usTiSys);
        _tprintf (_T ("Real Time: %02d:%02d:%02d.%03d\n"),

elTiSys.wHour, elTiSys.wMinute, elTiSys.wSecond,
                elTiSys.wMilliseconds);
        _tprintf (_T ("User Time: %02d:%02d:%02d.%03d\n"),
                usTiSys.wHour, usTiSys.wMinute, usTiSys.wSecond,
                usTiSys.wMilliseconds);
        _tprintf (_T ("Sys Time:  %02d:%02d:%02d.%03d\n"),
                keTiSys.wHour, keTiSys.wMinute, keTiSys.wSecond,
                keTiSys.wMilliseconds);

        CloseHandle (procInfo.hThread); CloseHandle (procInfo.hProcess);
        return 0;
}
```

Topic No - 103 (Simple job Management Shell)

Content:

Generating Console Events

Usage and Significance

Terminating another process can be problematic as the terminating process does not get a chance to release resources.

SEH does not help either as there is no mechanism that can be used to raise an exception in another process.

Console control event allows sending a message from one process to another.

Usually, a handler is set up in a process to catch such signals. Subsequently, the handler generates an exception.

It is hence possible for a process to generate an event in another process.

CreateProcess() can be used with the flag CREATE_NEW_PROCESS_GROUP.

This creates a process group in which the created process is the root and all the subsequently created processes by the parent are in the new group.

One process in a group can generate a CTRL_C_EVENT or CTRL_BREAK_EVENT in a group identified by the root process ID.

The target process should have the same console as the process generating the event.

More specifically, the calling process cannot have its own console using CREATE_NEW_CONSOLE or DETACHED_PROCESS flags.

BOOL GenerateConsoleCtrlEvent(DWORD dwCtrlEvent,

DWORD dwProcessGroupId);

dwCtrlEvent can be CTRL_C_EVENT or CTRL_BREAK_EVENT

dwProcessGroupId is the id of the process group

Topic No - 104 (Get a job Number)

Content:

Simple Job Management Shell

Job Management Shell

This job shell will allow three commands to run

jobbg

jobs

kill

The job shell parses the command line and then calls the respective function for the given command.

The shell uses a user-specific file keeping track of process ID and other related information

Several shells can run concurrently and use this shared file.

Also, concurrency issues can be encountered as several shells can try to use the same file.

File locking is used to make the file access mutually exclusive.

```
/*      Chapter 6 */
/* JobObjectShell.c One program combining three
job management commands:
        Jobbg   - Run a job in the background
        jobs    - List all background jobs
        kill    - Terminate a specified job of the job family
                        There is an option to generate a console
                        control signal.
        This implementation enhances JobShell with a time limit on each process.
                There is a time limit on each process, in seconds, in argv[1] (if present)
                0 or omitted means no process time limit
*/
#include "Everything.h"
#include "JobManagement.h"
```

```
#define MILLION 1000000
static int Jobbg (int, LPTSTR *, LPTSTR);
static int Jobs  (int, LPTSTR *, LPTSTR);
static int Kill  (int, LPTSTR *, LPTSTR);
HANDLE hJobObject = NULL;

JOBOBJECT_BASIC_LIMIT_INFORMATION basicLimits = {0, 0, JOB_OBJECT_LIMIT_PROCESS_TIME};

int _tmain (int argc, LPTSTR argv[])
{
LARGE_INTEGER processTimeLimit;
        BOOL exitFlag = FALSE;
        TCHAR command[MAX_COMMAND_LINE], *pc;
        DWORD i, localArgc;
        TCHAR argstr[MAX_ARG][MAX_COMMAND_LINE];
        LPTSTR pArgs[MAX_ARG];

        /* NT Only - due to file locking */
        if (!WindowsVersionOK (3, 1))
                ReportError (_T("This program requires Windows NT 3.1 or greater"), 1, FALSE);
        hJobObject = NULL;
        processTimeLimit.QuadPart = 0;
        if (argc >= 2) processTimeLimit.QuadPart = _ttoi(argv[1]);
        basicLimits.PerProcessUserTimeLimit.QuadPart = processTimeLimit.QuadPart *
MILLION;
```

```
	hJobObject = CreateJobObject(NULL, NULL);
	if (NULL == hJobObject)
		ReportError(_T("Error creating job object."), 1, TRUE);
	if (!SetInformationJobObject(hJobObject, JobObjectBasicLimitInformation, &basicLimits,
sizeof(JOBOBJECT_BASIC_LIMIT_INFORMATION)))
		ReportError(_T("Error setting job object information."), 2, TRUE);

	for (i = 0; i < MAX_ARG; i++)
		pArgs[i] = argstr[i];

	_tprintf (_T("Windows Job Mangement with Windows Job Object.\n"));
	while (!exitFlag) {
_tprintf (_T("%s"), _T("JM$"));
		_fgetts (command, MAX_COMMAND_LINE, stdin);
		pc = _tcschr (command, _T('\n'));
		*pc = _T('\0');

		GetArgs (command, &localArgc, pArgs);
		CharLower (argstr[0]);

		if (_tcscmp (argstr[0], _T("jobbg")) == 0) {
			Jobbg (localArgc, pArgs, command);
		}
		else if (_tcscmp (argstr[0], _T("jobs")) == 0) {
			Jobs (localArgc, pArgs, command);
		}
```

```
                else if (_tcscmp (argstr[0], _T("kill")) == 0) {
                        Kill (localArgc, pArgs, command);
}
                else if (_tcscmp (argstr[0], _T("quit")) == 0) {
                        exitFlag = TRUE;
                }
                else _tprintf (_T("Illegal command. Try again.\n"));
        }

        CloseHandle (hJobObject);

        return 0;
}

/* Jobbg: Execute a command line in the background, but
        the job identity in the user's job file, and exit.
        Related commands (jobs, fg, kill, and suspend) can be used to manage the jobs. */
/* jobbg [options] command-line
                -c: Give the new process a console.
                -d: The new process is detached, with no console.
        These two options are mutually exclusive.
        If neither is set, the background process shares the console with jobbg. */

/* These new features this program illustrates:
                1. Creating detached and with separate consoles.
                2. Maintaining a Jobs/Process list in a shared file.
```

```
                3. Determining a process status from a process ID.*/


/* Standard include files. */
/* - - - - - - - - - - - - - - - - - - - - - - - - */


int Jobbg (int argc, LPTSTR argv[], LPTSTR command)
{
/*      Similar to timep.c to process command line. */
        /*      - - - - - - - - - - - - - - - - - - - - - */
        /*      Execute the command line (targv) and store the job id,
                the process id, and the handle in the jobs file. */

        DWORD fCreate;
        LONG jobNumber;
        BOOL flags[2];

        STARTUPINFO startUp;
        PROCESS_INFORMATION processInfo;
        LPTSTR targv = SkipArg (command, 1, argc, argv);

        GetStartupInfo (&startUp);
/* Determine the options. */
        Options (argc, argv, _T ("cd"), &flags[0], &flags[1], NULL);

                /* Skip over the option field as well, if it exists. */
                /* Simplifying assumptions: There's only one of -d, -c (they are mutually exclusive.
```

```
                Also, commands can't start with -. etc. You may want to fix this. */
        if (argv[1][0] == _T('-'))
                targv = SkipArg (command, 2, argc, argv);

        fCreate = flags[0] ? CREATE_NEW_CONSOLE : flags[1] ? DETACHED_PROCESS : 0;

                /* Create the job/thread suspended.
                        Resume it once the job is entered properly. */
if (!CreateProcess (NULL, targv, NULL, NULL, TRUE,
                        fCreate | CREATE_SUSPENDED | CREATE_NEW_PROCESS_GROUP,
                        NULL, NULL, &startUp, &processInfo)) {
                ReportError (_T ("Error starting process."), 0, TRUE);
        }
        if (hJobObject != NULL)
        {
                if (!AssignProcessToJobObject(hJobObject, processInfo.hProcess)) {
                        ReportError(_T("Could not add process to job object. The process will be
terminated."), 0, TRUE);
                        TerminateProcess (processInfo.hProcess, 4);
                        CloseHandle (processInfo.hThread);
                        CloseHandle (processInfo.hProcess);
                        return 4;
}
        }

                /* Create a job number and enter the process Id and handle
```

```
                    into the Job "database" maintained by the
                    GetJobNumber function (part of the job management library). */

        jobNumber = GetJobNumber (&processInfo, targv);
        if (jobNumber >= 0)
                ResumeThread (processInfo.hThread);
        else {
                TerminateProcess (processInfo.hProcess, 3);
                CloseHandle (processInfo.hThread);
                CloseHandle (processInfo.hProcess);
                ReportError (_T ("Error: No room in job control list."), 0, FALSE);
                return 5;
}

        CloseHandle (processInfo.hThread);
        CloseHandle (processInfo.hProcess);
        _tprintf (_T (" [%d] %d\n"), jobNumber, processInfo.dwProcessId);
        return 0;
}

/* Jobs: List all running or stopped jobs that have
        been created by this user under job management;
        that is, have been started with the jobbg command.
        Related commands (jobbg and kill) can be used to manage the jobs. */
/* These new features this program illustrates:
        1. Determining process status.
```

```
	2. Maintaining a Jobs/Process list in a shared file.

	3. Obtaining job object information

*/



int Jobs (int argc, LPTSTR argv[], LPTSTR command)

{

	JOBOBJECT_BASIC_ACCOUNTING_INFORMATION basic info;



	if (!DisplayJobs ()) return 1;

	/* Dispaly the job information */

	if (!QueryInformationJobObject(hJobObject, JobObjectBasicAccountingInformation, &basicInfo,
sizeof(JOBOBJECT_BASIC_ACCOUNTING_INFORMATION), NULL)) {

		ReportError(_T("Failed QueryInformationJobObject"), 0, TRUE);

		return 0;

	}

	_tprintf (_T("Total Processes: %d, Active: %d, Terminated: %d.\n"),

		basicInfo.TotalProcesses, basicInfo.ActiveProcesses,

basicInfo.TotalTerminatedProcesses);

	_tprintf (_T("User time all processes: %d.%03d\n"),

		basicInfo.TotalUserTime.QuadPart / MILLION, (basicInfo.TotalUserTime.QuadPart %
MILLION) / 10000);



	return 0;

}
```

```
/* kill [options] jobNumber
        Terminate the process associated with the specified job number. */
/* This new features this program illustrates:
        1. Using TerminateProcess
        2. Console control events */
/* Options:
        -b  Generate a Ctrl-Break
        -c  Generate a Ctrl-C
Otherwise, terminate the process. */

/* The Job Management data structures, error codes,
        constants, and so on are in the following file. */

int Kill (int argc, LPTSTR argv[], LPTSTR command)
{
        DWORD processId, jobNumber, iJobNo;
        HANDLE hProcess;
        BOOL cntrlC, cntrlB, killed;

        iJobNo = Options (argc, argv, _T ("bc"), &cntrlB, &cntrlC, NULL);

        /* Find the process ID associated with this job. */

jobNumber = _ttoi (argv[iJobNo]);
        processId = FindProcessId (jobNumber);
        if (processId == 0)         {
```

```
                ReportError (_T ("Job number not found.\n"), 0, FALSE);
                return 1;
        }
        hProcess = OpenProcess (PROCESS_TERMINATE, FALSE, processId);
        if (hProcess == NULL) {
                ReportError (_T ("Process already terminated.\n"), 0, FALSE);
                return 2;
        }



if (cntrlB)
                killed = GenerateConsoleCtrlEvent (CTRL_BREAK_EVENT, 0);
        else if (cntrlC)
                killed = GenerateConsoleCtrlEvent (CTRL_C_EVENT, 0);
        else
                killed = TerminateProcess (hProcess, JM_EXIT_CODE);
        if (!killed) {
                ReportError (_T ("Process termination failed."), 0, TRUE);
                return 3;
        }
                WaitForSingleObject (hProcess, 5000);
        CloseHandle (hProcess);

        _tprintf (_T ("Job [%d] terminated or timed out\n"), jobNumber);
        return 0;
}
```

Topic No - 105 (Listing Background jobs)

Content: Content: Content: Job Listing

The job management function used for the purpose is DisplayJobs()

The function opens the file.

Looks up into the file and acquires the status of the processes listed in the file.

Also displays the status of the processes listed and other information.

BOOL DisplayJobs (void)

```
/* Scan the job database file, reporting on the status of all jobs.
        In the process remove all jobs that no longer exist in the system. */
{
        HANDLE hJobData, hProcess;
        JM_JOB jobRecord;
        DWORD jobNumber = 0, nXfer, exitCode;
        TCHAR jobMgtFileName[MAX_PATH];
        OVERLAPPED regionStart;

        if ( !GetJobMgtFileName (jobMgtFileName) )
                return FALSE;
        hJobData = CreateFile (jobMgtFileName, GENERIC_READ | GENERIC_WRITE,
                FILE_SHARE_READ | FILE_SHARE_WRITE,
                NULL, OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL, NULL);
```

```
if (hJobData == INVALID_HANDLE_VALUE)
                return FALSE;

        /* Read records and report on each job. */
        /*  First, create an exclusive lock on the entire
                file as entries will be changed. */
        regionStart.Offset = 0;
        regionStart.OffsetHigh = 0;
        regionStart.hEvent = (HANDLE)0;
        LockFileEx (hJobData, LOCKFILE_EXCLUSIVE_LOCK, 0, 0, 0, ®ionStart);

        __try {
        while (ReadFile (hJobData, &jobRecord, SJM_JOB, &nXfer, NULL) && (nXfer > 0)) {
                jobNumber++; /* jobNumber won't exceed MAX_JOBS_ALLOWED as the file is

not allowed to grow that long */
                hProcess = NULL;
                if (jobRecord.ProcessId == 0) continue;
                                        /* Obtain process status. */
                hProcess = OpenProcess (PROCESS_ALL_ACCESS, FALSE, jobRecord.ProcessId);
                if (hProcess != NULL) {
                        GetExitCodeProcess (hProcess, &exitCode);
                        CloseHandle (hProcess);
                }
                                        /* Show the job status. */
```

```
                _tprintf (_T (" [%d] "), jobNumber);
                if (NULL == hProcess)
                        _tprintf (_T (" Done"));
                else if (exitCode != STILL_ACTIVE)
                        _tprintf (_T ("+ Done"));
else _tprintf (_T ("      "));
                                        /* Show the command. */
                _tprintf (_T (" %s\n"), jobRecord.CommandLine);

                /* Remove processes that are no longer in the system. */

                if (NULL == hProcess) { /* Back up one record. */
                        /* SetFilePoiner is more convenient here than SetFilePointerEx since the move is
a short shift from the current position */
                        SetFilePointer (hJobData, -(LONG)nXfer, NULL, FILE_CURRENT);
                                /* Set the job number to 0. */
                        jobRecord.ProcessId = 0;
                        if (!WriteFile (hJobData, &jobRecord, SJM_JOB, &nXfer, NULL))
                                ReportError (_T ("Rewrite error."), 1, TRUE);

}
        } /* End of while. */
        } /* End of try.   */

        __finally {             /* Release the lock on the file. */
                UnlockFileEx (hJobData, 0, 0, 0, ®ionStart);
```

```
                if (NULL != hProcess) CloseHandle (hProcess);
        }

        CloseHandle (hJobData);
        return TRUE;
}
```

Topic No - 106 (Finding a Process Id)

Content:

Finding a Process Id

The job management function used for the purpose is FindProcessId()

It obtains the process id of the given job number

It simply looks up into the file based on job number and reads the record at the specific location.

```
DWORD FindProcessId (DWORD jobNumber)

/* Obtain the process ID of the specified job number. */
{
        HANDLE hJobData;
```

```
        JM_JOB jobRecord;
        DWORD nXfer, fileSizeLow;
        TCHAR jobMgtFileName[MAX_PATH+1];
        OVERLAPPED regionStart;
        LARGE_INTEGER fileSize;

        if ( !GetJobMgtFileName (jobMgtFileName) ) return 0;
        hJobData = CreateFile (jobMgtFileName, GENERIC_READ,
                        FILE_SHARE_READ | FILE_SHARE_WRITE,
                        NULL, OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL, NULL);
if (hJobData == INVALID_HANDLE_VALUE) return 0;
        /* Position to the correct record, but not past the end of file */
        /* As a variation, use GetFileSize to demonstrate its operation. */
        if (!GetFileSizeEx (hJobData, &fileSize) ||
                (fileSize.HighPart != 0 || SJM_JOB * (jobNumber - 1) > fileSize.LowPart
                 || fileSize.LowPart > SJM_JOB * MAX_JOBS_ALLOWED))
                        return 0;
        fileSizeLow = fileSize.LowPart;
        /* SetFilePoiner is more convenient here than SetFilePointerEx since the the file is known to be
"short" ( < 4 GB). */
        SetFilePointer (hJobData, SJM_JOB * (jobNumber - 1), NULL, FILE_BEGIN);

        /* Get a shared lock on the record. */
        regionStart.Offset = SJM_JOB * (jobNumber - 1);
        regionStart.OffsetHigh = 0; /* Assume a "short" file. */
        regionStart.hEvent = (HANDLE)0;
```

```
        LockFileEx (hJobData, 0, 0, SJM_JOB, 0, &regionStart);


        if (!ReadFile (hJobData, &jobRecord, SJM_JOB, &nXfer, NULL))
                ReportError (_T ("JobData file error"), 0, TRUE);


        UnlockFileEx (hJobData, 0, SJM_JOB, 0, &regionStart);
        CloseHandle (hJobData);


        return jobRecord.ProcessId;
}
```

Topic No - 107 (Job Objects)

Content: Job Objects
Windows provides the provision of grouping a number of processes into a job object.
Resource limits can be specified for each job object.
This also helps maintain accounting information
Creating Job Objects
Firstly a job object is created using CreateJobObject(). It uses a name and security attributes.
Also, OpenJobObject() is used with named object and CloseHandle() is used to close the object.
Once the job object is created it can be assigned processes using AssignProcesstoJobObject()
Creating Job Objects
Once a process is assigned to a job object it cannot be assigned to another job object.
Child processes are automatically assigned the same job object as the parent unless the CREATE_BREAKAWAY_FROM_JOB flag is used at creation.
Control limits are specified to the processes in job object using:
BOOL SetInformationJobObject(HANDLE hJob,
JOBOBJECTINFOCLASS JobObjectInformationClass,
LPVOID lpJobObjectInformation, DWORD cbJobObjectInformationLength);

hJob is handled to the existing job object

JobObjectInformationClass is the class of the limits you wish to set regarding per process time, working set limit, the limit on active processes, priority, processor affinity, etc.

lpJobObjectInformation is the structure congaing the actual information as it uses a different structure for each class.

Topic No - 108 (Using Job Objects)
Content: Using Job Objects
In this example job objects are used to limit process execution time and obtain user time statistics.
The simple JobShell program discussed previously is modified.
A command line time limit argument is added.

```
/* Chapter 6 */
/* JobObjectShell.c One program combining three
job management commands:
Jobbg - Run a job in the background
jobs - List all background jobs
kill - Terminate a specified job of job family
  There is an option to generate a console
  control signal.
This implemenation enhances JobShell with a time limit on each process.
There is a time limit on each process, in seconds, in argv[1] (if present)
0 or omitted means no process time limit
*/

#include "Everything.h"
#include "JobManagement.h"

#define MILLION 1000000
static int Jobbg (int, LPTSTR *, LPTSTR);
static int Jobs  (int, LPTSTR *, LPTSTR);
static int Kill  (int, LPTSTR *, LPTSTR);
HANDLE hJobObject = NULL;

JOBOBJECT_BASIC_LIMIT_INFORMATION basicLimits = {0, 0, JOB_OBJECT_LIMIT_PROCESS_TIME};

int _tmain (int argc, LPTSTR argv[])
{
LARGE_INTEGER processTimeLimit;
BOOL exitFlag = FALSE;
```

```
TCHAR command[MAX_COMMAND_LINE], *pc;
DWORD i, localArgc;
TCHAR argstr[MAX_ARG][MAX_COMMAND_LINE];
LPTSTR pArgs[MAX_ARG];
/* NT Only - due to file locking */
if (!WindowsVersionOK (3, 1))
ReportError (_T("This program requires Windows NT 3.1 or greater"), 1,
FALSE);
hJobObject = NULL;
processTimeLimit.QuadPart = 0;
if (argc >= 2) processTimeLimit.QuadPart = _ttoi(argv[1]);
basicLimits.PerProcessUserTimeLimit.QuadPart = processTimeLimit.QuadPart * MILLION;

hJobObject = CreateJobObject(NULL, NULL);
if (NULL == hJobObject)
ReportError(_T("Error creating job object."), 1, TRUE);
if (!SetInformationJobObject(hJobObject, JobObjectBasicLimitInformation, &basicLimits,
sizeof(JOBOBJECT_BASIC_LIMIT_INFORMATION)))
ReportError(_T("Error setting job object information."), 2, TRUE);

for (i = 0; i < MAX_ARG; i++)
pArgs[i] = argstr[i];

_tprintf (_T("Windows Job Mangement with Windows Job Object.\n"));
while (!exitFlag) {
_tprintf (_T("%s"), _T("JM$"));
_fgetts (command, MAX_COMMAND_LINE, stdin);
pc = _tcschr (command, _T('\n'));
*pc = _T('\0');

GetArgs (command, &localArgc, pArgs);
CharLower (argstr[0]);

if (_tcscmp (argstr[0], _T("jobbg")) == 0) {
Jobbg (localArgc, pArgs, command);
}
else if (_tcscmp (argstr[0], _T("jobs")) == 0) {
Jobs (localArgc, pArgs, command);
}
else if (_tcscmp (argstr[0], _T("kill")) == 0) {
Kill (localArgc, pArgs, command);
}
```

```
else if (_tcscmp (argstr[0], _T("quit")) == 0) {
exitFlag = TRUE;
}
else _tprintf (_T("Illegal command. Try again.\n"));
}
CloseHandle (hJobObject);
return 0;
}

/* Jobbg: Execute a command line in the background, put
the job identity in the user's job file, and exit.
Related commands (jobs, fg, kill, and suspend) can be used to manage the jobs. */

/* jobbg [options] command-line
-c: Give the new process a console.
-d: The new process is detached, with no console.
These two options are mutually exclusive.
If neither is set, the background process shares the console with jobbg. */

/* These new features this program illustrates:
1. Creating detached and with separate consoles.
2. Maintaining a Jobs/Process list in a shared file.
3. Determining a process status from a process ID.*/
/* Standard includes files. */
/* - - - - - - - - - - - - - - - - - - - - - - - - */

int Jobbg (int argc, LPTSTR argv[], LPTSTR command)
{
/* Similar to timep.c to process command line. */
/* - - - - - - - - - - - - - - - - - - - - - - */
/* Execute the command line (targv) and store the job id,
the process id, and the handle in the jobs file. */
DWORD fCreate;
LONG jobNumber;
BOOL flags[2];
STARTUPINFO startUp;
PROCESS_INFORMATION processInfo;
LPTSTR targv = SkipArg (command, 1, argc, argv);
GetStartupInfo (&startUp);

/* Determine the options. */
Options (argc, argv, _T ("cd"), &flags[0], &flags[1], NULL);
```

```
/* Skip over the option field as well, if it exists. */
/* Simplifying assumptions: There's only one of -d, -c (they are mutually exclusive.
   Also, commands can't start with -. etc. You may want to fix this. */
if (argv[1][0] == _T('-'))
targv = SkipArg (command, 2, argc, argv);

fCreate = flags[0] ? CREATE_NEW_CONSOLE : flags[1] ? DETACHED_PROCESS : 0;
/* Create the job/thread suspended.
Resume it once the job is entered properly. */
if (!CreateProcess (NULL, targv, NULL, NULL, TRUE,
fCreate | CREATE_SUSPENDED | CREATE_NEW_PROCESS_GROUP,
NULL, NULL, &startUp, &processInfo)) {
ReportError (_T ("Error starting process."), 0, TRUE);
}
if (hJobObject != NULL)
{
if (!AssignProcessToJobObject(hJobObject, processInfo.hProcess)) {
ReportError(_T("Could not add process to job object. The process will be terminated."), 0, TRUE);
TerminateProcess (processInfo.hProcess, 4);
CloseHandle (processInfo.hThread);
CloseHandle (processInfo.hProcess);
return 4;
}
}

/* Create a job number and enter the process Id and handle
into the Job "database" maintained by the
GetJobNumber function (part of the job management library). */
jobNumber = GetJobNumber (&processInfo, targv);
if (jobNumber >= 0)
ResumeThread (processInfo.hThread);
else {
TerminateProcess (processInfo.hProcess, 3);
CloseHandle (processInfo.hThread);
CloseHandle (processInfo.hProcess);
ReportError (_T ("Error: No room in job control list."), 0, FALSE);
return 5;
}
CloseHandle (processInfo.hThread);
CloseHandle (processInfo.hProcess);
_tprintf (_T (" [%d] %d\n"), jobNumber, processInfo.dwProcessId);
```

```
return 0;
}
/* Jobs: List all running or stopped jobs that have
been created by this user under job management;
that is, have been started with the jobbg command.
Related commands (jobbg and kill) can be used to manage the jobs. */
/* These new features this program illustrates:
1. Determining process status.
2. Maintaining a Jobs/Process list in a shared file.
3. Obtaining job object information
 */
int Jobs (int argc, LPTSTR argv[], LPTSTR command)
{
JOBOBJECT_BASIC_ACCOUNTING_INFORMATION basicInfo;
if (!DisplayJobs ()) return 1;
/* Dispaly the job information */
if (!QueryInformationJobObject(hJobObject, JobObjectBasicAccountingInformation, &basicInfo,
sizeof(JOBOBJECT_BASIC_ACCOUNTING_INFORMATION), NULL)) {
ReportError(_T("Failed QueryInformationJobObject"), 0, TRUE);
return 0;
}
_tprintf (_T("Total Processes: %d, Active: %d, Terminated: %d.\n"),
basicInfo.TotalProcesses, basicInfo.ActiveProcesses, basicInfo.TotalTerminatedProcesses);
_tprintf (_T("User time all processes: %d.%03d\n"),
basicInfo.TotalUserTime.QuadPart / MILLION, (basicInfo.TotalUserTime.QuadPart % MILLION) / 10000);

return 0;
}
/* kill [options] jobNumber
Terminate the process associated with the specified job number. */
/* This new features this program illustrates:
1. Using TerminateProcess
2. Console control events */
/* Options:
-b  Generate a Ctrl-Break
-c  Generate a Ctrl-C
Otherwise, terminate the process. */

/* The Job Management data structures, error codes,
constants, and so on are in the following file. */

int Kill (int argc, LPTSTR argv[], LPTSTR command)
```

```
{
DWORD processId, jobNumber, iJobNo;
HANDLE hProcess;
BOOL cntrlC, cntrlB, killed;

iJobNo = Options (argc, argv, _T ("bc"), &cntrlB, &cntrlC, NULL);
/* Find the process ID associated with this job. */

jobNumber = _ttoi (argv[iJobNo]);
processId = FindProcessId (jobNumber);
if (processId == 0) {
ReportError (_T ("Job number not found.\n"), 0, FALSE);
return 1;
}
hProcess = OpenProcess (PROCESS_TERMINATE, FALSE, processId);
if (hProcess == NULL) {
ReportError (_T ("Process already terminated.\n"), 0, FALSE);
return 2;
}
if (cntrlB)
killed = GenerateConsoleCtrlEvent (CTRL_BREAK_EVENT, 0);
else if (cntrlC)
killed = GenerateConsoleCtrlEvent (CTRL_C_EVENT, 0);
else
killed = TerminateProcess (hProcess, JM_EXIT_CODE);
if (!killed) {
ReportError (_T ("Process termination failed."), 0, TRUE);
return 3;
}
WaitForSingleObject (hProcess, 5000);
CloseHandle (hProcess);
_tprintf (_T ("Job [%d] terminated or timed out\n"), jobNumber);
return 0;
}
```

Topic No - 109 (Thread Overview)

Content: Threads overview

Thread is an independent unit of execution within a process.

In a multi-threaded system, multiple threads may exist within a process.

Organizing and coordinating these threads is a challenge.

Some programming problems are greatly simplified by the use of threads.

The conventional scenario uses single-threaded processes being run concurrently

This scenario has varying disadvantages

Switching among processes is time-consuming and expensive. Threads can easily allow concurrent processing of the same function hence reducing overheads

Usually, independent processes are not tightly coupled, and hence sharing resources is difficult.

Synchronization and mutual exclusion in single-threaded processes halt computations.

Threads can perform the asynchronously overlapped functions with less programming effort.

The use of multithreading can benefit more with a multiprocessor environment using efficient scheduling of threads.

Topic No - 110 (Thread Issues)

Content: Issues with Threads

Threads share resources within a process. One thread may inadvertently another thread's data.

In some specific cases, concurrency can greatly degrade performance rather than improve.

In some simple single-threaded solutions using multithreading greatly complicates matters and even results in poor performance.

Topic No - 111 (Thread Basics)

Content: Threads Basics

Threads use the space assigned to a process.

In a multi threaded system, multiple threads might be associated with a process

Thread within a process share data and code.

However, individual threads may have their unique data storage in addition to the shared data.

This data is not altogether exclusive.

Programmer must assure that a thread only uses its own data and does not interfere with shared data.

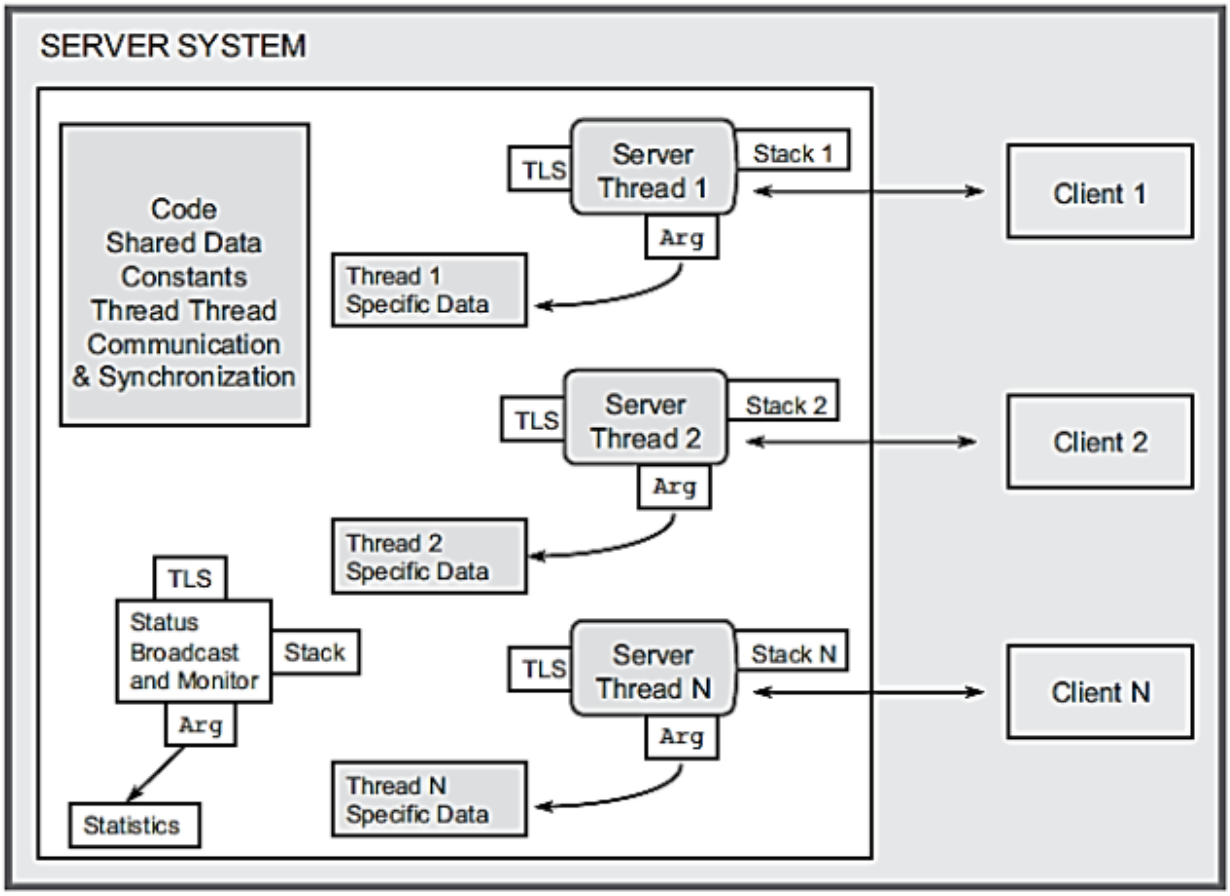
Moreover, each thread has its own stack for function calls.

The calling process usually passes an argument to the thread. These arguments are stored in the thread's stack

Each thread can allocate its own Thread Local Storage(TLS) and set clear values.

TLS is a small pointer array that is only accessible to the thread.

This also assures that a thread will not modify data of any other thread's TLS.



Topic No - 112 (Thread Management)

Content: Thread Management

Like every other resource threads are also treated as objects

The API used to create a thread is CreateThread()

Threads can also be treated as parent and child. Although the OS is unaware of that.

CreateThread() has many unique requirements

The CreateThread() requires the starting address of the thread within the calling process.

It also requires stack space size for the thread which comes from process address space.

Default memory space IMB

One page is committed in start, this grows with requirement.

Requires a pointer to thread argument. This can be any structure.

HANDLE CreateThread(LPSECURITY_ATTRIBUTES lpsa, SIZE_T dwStackSize, LPTHREAD_START_ROUTINE lpStartAddress, LPVOID lpParameter, DWORD dwCreationFlags, LPDWORD lpThreadId);

lpsa is security attributes structure used previously.

dwStackSize is the stack size in bytes.

lpStartAddress is the starting address of the thread function within the calling process of the form:

DWORD WINAPI ThreadFunc(LPVOID)

The function returns a DWORD value which is usually an exit code and accepts a single pointer argument.

lpThreadParam is the pointer passed to the thread and is usually interpreted as a pointer to a structure containing arguments

dwCreationFlags if 0 would been that thread would start readily. If its CREATE_SUSPENDED then the thread will be suspended requiring the use of ResumeThread() to start execution.

lpThreadId is a pointer to a DWORD that will receive the thread identifier. If its kept NULL then no thread identifier will be returned,

Topic No - 113 (Exiting Thread)

Content: Exiting Thread

All threads in a process can exit using a ExitThread() function.

An alternate is that the thread function returns with the exit code.

When a thread exits the thread stack is deallocated and the handle reffering to the thread are invalidated.

If the thread is linked to some DLL then the DllMain() function is invoked with te reason DLL_THREAD_DETACH.

VOID ExitThread(DWORD dwExitCode);

When all the threads exit the process terminates.

One thread can terminate another thread using TerminateThread().

In this case Thread resources will not be deallocated, completion handlers do not execute, no notification is sent to attached DLLs.

Because of these reasons use of TerminateThread() is strongly discouraged.

A thread object will continue to exist even its execution has ended until the last reference to the thread handle is destroyed with CloseHandle().

Any other running thread can retrieve the exit code.

BOOL GetExitCodeThread(HANDLE hThread, LPDWORD lpdwExitCode);

lpdwExitCode contains a pointer to exit code.

The thread is still running the exit code will be STILL_ACTIVE

Topic No - 114 (Thread Identity)

Content: Thread Identity

Thread Ids and handles can be obtained using functions quite similar to one used with processes.

GetCurrentThread()

Non-inheritable pseudohandle to the calling thread.

GetCurrentThreadId()

Obtains thread Id rather than the handle.

GetThreadId()

Obtains thread Id from thread handle

OpenThread()

Creates Thread handle from the thread Id