Mid Term from Feb-2005 to Nov 2012
CS604- Operating System
Latest Solved Subjective from Mid term Papers
Resource Person Hina

Question No.1
List and define the different metrics by which might evaluate a scheduler (List at least 4). 5 marks
Answer: handouts Page No. 80
When evaluating a scheduler's performance, utilization and Throughput are traded off for better Response Time. Response time is important for OS's that aim to be user-friendly.
List of Schedulers include:
1. First-Come, First-Served (FCFS)
2. Round-Robin (RR)
3. Shortest-Job-First (SJF)
4. Priority Scheduling (PS)

Question No.2
Write brief the multilevel feedback queue scheduling. 5 marks
Answer: handouts Page No. 89
Multilevel feedback queue scheduling allows a process to move between queues. The idea is to separate processes with different CPU burst characteristics. If a process uses too much CPU time, it will be moved to a lower-priority queue. This scheme leaves I/O bound and interactive processes in the higher-priority queues. Similarly a process that waits too long in a lower-priority queue may be moved o a higher priority queue. This form of aging prevents starvation.
In general, a multi-level feedback queue scheduler is defined by the following parameters:
- Number of queues
- Scheduling algorithm for each queue
- Method used to determine when to upgrade a process to higher priority queue
- Method used to determine when to demote a process
- Method used to determine which queue a process enters when it needs service

Question No.3
Assumption made while formulating a solution to the critical section problem. 2 marks
Answer: handouts Page No. 98
A solution to the critical section problem must satisfy the following three requirements:
Assumptions made While formulating a solution, we must keep the following assumptions in mind:
- Assume that each process executes at a nonzero speed
- No assumption can be made regarding the relative speeds of the N processes.

Question No.4
There are many commands. Write the method through which these commands can communicate with each other. 3 marks
Answer: handouts Page No. 27
There are many commands some of these are given below:-
Command Name $ pw
Details
You can display the absolute pathname of your working directory with the pwd command
Syntax
/home/students/nadeem/courses/cs604/programs

Command Name cp
Details

You can use the cp command for copying files. You can use the cp file1 file2 command to copy file1 to file2.
Syntax
cp ~/file1 ~/memos/file2

Command Name mv
Details
You can use the mv command for moving files. You can use the mv file1 file2 command to move file1 to file2.
Syntax
$ mv ~/file1 ~/memos/file2

Command Name rm
Details
You can use the rm command to remove files. You can use the rm file1 command to remove file1
Syntax
$ rm ~/courses/cs604/programs/test.c $ For single file
rm *.o For all files

Question No.5
Write Difference between SJF and Shortest Remaining Time First Scheduling algorithm. 3 marks
Answer: handouts Page No. 82

Shorted Job First (SJF) Scheduling	Shortest Remaining Time First Scheduling
For long term scheduling in a batch system, we can use as the length the process time limit that a user specifies when he submits the job	Whenever a new job comes in, check the remaining service time on the current job.
Preemptive SJF scheduling is sometimes called shortest-remaining-time-first scheduling.	For all but the longest jobs, SRT better than SJF
The response ratio is good (Fast)	The response ratio is good (low)
Waiting time is fast	Waiting time is also quite low for most processes

Question No 6.
Write the formula/ procedure for calculating the waiting time in preemptive Shortest Job First scheduling
Answer: handouts Page No. 83
Preemptive SJF scheduling is sometimes called shortest-remaining-time-first scheduling. We illustrate the working of the SJF algorithm by using the following system state.

Process	Arrival Time	Burst Time
P1	0.0	7
P2	2.0	4
P3	4.0	1
P4	5.0	4

Qno7.
Define race condition and how prevent this condition. 2 marks
Answer: handouts Page No. 97
A situation like this, where several processes access and manipulate the same data concurrently and the outcome of the manipulation depends on the particular order in which the access takes place, is called a race condition.

Question No.8
What is Convoy Effect?
Answer: handouts Page No. 82
When FCFS scheduling algorithm is used, the convoy effect occurs when short Processes wait behind a long process to use the CPU and enter the ready queue in a Convoy after completing their I/O.

Question No.9

What are the common data structures in Bakery Algorithm?

Answer: handouts Page No. 103

The bakery algorithm is due to Leslie Lamport and is based on a scheduling algorithm commonly used in bakeries, ice-cream stores.

Question No.10 How a pipe can be created?

Answer: handouts Page No. 103

The pipe() system call creates a pipe and returns two file descriptors, one for reading and second for writing,

Question No.11

Difference between "progress" and " bounded time: in critical section

Answer: handouts Page No. 99

Process:

If no process is executing in its critical section and some processes wish to enter their critical sections, then only those processes that are not executing in their remainder section can participate in the decision on which will enter its critical section next, and this selection cannot be postponed indefinitely.

Bounded Waiting:

There exists a bound on the number of times that other processes are allowed to enter their critical sections after a process has made a request to enter its critical section and before that request is granted.

Question No.12 What is Starvation and how it occures

Answer: handouts Page No. 129

Starvation:

A process that is ready to run but lacking the CPU can be considered blocked-waiting for the CPU.

How it occur:-

If a deadlock occurs, it can be resolved if one car backs up (preempt resources and rollback). Several cars may have to be backed up if a deadlock occurs. Starvation is possible.

Question No.13 What are the advantages of Round Robin Scheduling Algorithm?

Answer: handouts Page No. 86

- If time slice is too short then you will have freq switching btw processes.
- If time slice is too long then you will have less switching btw processes and high priority may have to wait for low priority tasks leading to starvation.
- The average waiting time under the RR policy however is often quite long.
- The CPU scheduler goes around the ready queue, allocating the CPU to each process for a time interval of up to 1 time quantum.
- The process may have a CPU burst of less than 1 time quantum, in which case the process itself will release the CPU voluntarily.
- It is the simplest scheduling algorithm.
- It is easy to implement in software.
- If the processes are of varied length then it becomes slow.

Question No.14

Analyze the following algorithm to solve the critical section problem and explain whether it satisfies the Mutual Exclusion Characteristic

```
Flag[i] = True;
Turn = j;
do{
while(Flag[j] = True && turn==j);
critical section
```

Flag[i] = False;
remainder section
} While(1)

Answer: handouts Page No. 145
To enter its critical section, Pi sets flag[i] to true, and sets 'turn' to j, asserting that if the other process wishes to enter its critical section, it may do so. If both try to enter at the same time, they will attempt to set 'turn' to i and j. However, only one of these assignments will last, the other will occur but be overwritten instantly. Hence, the eventual value of 'turn' will decide which process gets to enter its critical section.
To prove mutual exclusion, note that Pi enters its critical section only if either flag[j]=false or turn=i. Also, if both processes were executing in their critical sections at the same time, then flag[0]= = flag[1]= = true. These two observations suggest that P0 and P1 could not have found both conditions in the while statement true at the same time, since the value of 'turn' can either be 0 or 1. Hence only one process say P0 must have successfully exited the while statement. Hence mutual exclusion is preserved.
To prove bounded wait and progress requirements, we note that a process Pi can be prevented the critical section only if it is stuck in the while loop with the condition flag[j]= =true and turn=j. If Pj is not ready to enter the critical section, then flag[j]=flase and Pi can enter its critical section. If Pj has set flag[j]=true and is also executing its while statement then either turn=i or turn=j. If turn=i then Pi enters its critical section, otherwise Pj. However, whenever a process finishes executing in its critical section, lets assume Pj, it resets flag[j] to false allowing Pi to enter its critical section. If Pj resets flag[j]=true, then it must also set 'turn' to i, and since Pi does not change the value of 'turn' while executing in its while statement, Pi will enter its critical section (progress) after at most one entry by Pj (bounded waiting).

Question No.15 which command is used in Linux environment to get process information? (2)
Answer: handouts Page No. 63
ps gives a snapshot of the current processes. Without options, ps prints information about processes owned by the user. Some of the commonly used options are -u, -e, and -l.

- -e selects all processes
- -l formats the output in the long format
- -u displays the information in user-oriented format

Question No.16 resource sharing is disadvantage of multithreading....explain?(5)
Answer: handouts Page No. 70
1. Resource sharing: Whereas resource sharing is one of the major advantages of threads, it is also a disadvantage because proper synchronization is needed between threads for accessing the shared resources (e.g., data and files).
2. Difficult programming model: It is difficult to write, debug, and maintain multi-threaded programs for an average user. This is particularly true when it comes to writing code for synchronized access to shared resources.

Question No.17 Synchronization process is needed in resource sharing and data sharing (3)
Answer: handouts Page No. 70
Often access to shared data and shared resources, if there is no controlled access to shared data, it is often possible to obtain an inconsistent state of this data. Maintaining data consistency requires mechanisms to ensure the orderly execution of cooperating processes, and hence various process synchronization methods are used.

Question No.17 how threads overcome these issues (5)
Answer: handouts Page No. 68
A thread, sometimes called a lightweight process (LWP), is a basic unit of CPU utilization and executes within the address space of the process that creates it. It comprises a thread ID, a program counter, a register set, errno, and a stack. It shares with other threads belonging to the same process its code sections, data section, current working directory, user and group IDs, signal setup and handlers, PCB and other operating system resources, such as open files and system.
Question No.18 If process in background then we want its move in foreground then what unix/inux command is use to moving. 3 marks

Answer: **handouts Page No. 65**

Moving a process into background

You can use the bg command to put the current or a suspended process into the background. Here is the syntax of the command. bg [%job_id]

If %job_id is omitted the current job is assumed.

Question No.19 How the open source help us to test the algorithm?

Answer: **handouts Page No. 95**

The Open Source software licensing has made it possible for us to test various algorithms by implementing them in the Linux kernel and measuring their true performance. The major difficulty is the cost of this approach. The expense is incurred in coding the algorithm and modifying the operating system to support it, as well as its required data structures. The other difficulty with any algorithm evaluation is that the environment in which the algorithm will change.

Question No.20 Using C for compiling means?

Answer: **handouts Page No. 28**

To compile a source file titled program.c, type:

$ gcc program.c

You can run the executable program generated by this command by typing./a.out and hitting the <Enter> key, as hown in the following session.

$ ./a.out [... program output ...]

You can store the executable program in a specific file by using the –o option. For example, in the following session, the executable program is stored in the assignment file.

$ gcc program.c -o assignment

You can link a library explicitly by using the –l option. In the following session, we are asking the compiler to link the math library with our object file as it creates the executable file.

$ gcc program.c –o assignment -lm

$ assignment

[... program output ...]

Question No.21 what is difference b/w preemptive and non-preemptive (2)

Answer: **handouts Page No. 80**

Preemptive	**non-preemptive**
in preemptive scheduling we preempt the currently executing process, in non preemptive scheduling we allow the current process to finish its CPU burst time	in non preemptive scheduling the process at running state can not be forced to leave the CPU until it completes On a preemptive kernel, a process running in kernel mode can be replaced by another process while
in preemptive scheduling the process is forcibly sent to waiting state when a process with higher priority comes to CPU,	**When scheduling takes place only under :-** • When a process switches from the running state to the waiting state (for example, an I/O request is being completed) • When a process switches from the running state to the ready state (for example when an interrupt occurs) • When a process switches from the waiting state to the ready state (for example, completion of I/O) • When a process terminates

Question No.22 Virtual machine V scheduling algorithm different steps

Answer: handouts Page No. 91

UNIX System V scheduling algorithm is essentially a multilevel feedback priority queues algorithm with round robin within each queue, the quantum being equal to1 second. The priorities are divided into two groups/bands:

- Kernel Group
- User Group

Priorities in the Kernel Group are assigned in a manner to minimize bottlenecks, i.e, processes waiting in a lower-level routine get higher priorities than those waiting at relatively higher-level routines. We discuss this issue in detail in the lecture with an example. In decreasing order of priority, the kernel bands are:

- Swapper
- Block I/O device control processes
- File manipulation
- Character I/O device control processes
- User processes

Question No.23 If a process exits and there are still threads of that process running, will they continue to run?

Answer: Click here for detail

if the thread in the process is running and receives a signal(say Ctrl-C) and the default action of the signal is to terminate a process, does the running thread terminates or the parent process will also terminate. That happens to the threads if the running process terminates because of some signal

Question No.24 What are the important characteristics of TestAndSet? What will be its advantage.

Answer: handouts Page No. 106

The important characteristic is that this instruction is executed atomically. Thus if two TestAndSet instructions are executed simultaneously, they will be executed sequentially in some arbitrary order.

Advantages :-

- Simple to implement and understand
- One memory location for arbitrarily many CPUs

Question No.25

Considering the Resource sharing feature of thread, what do you think is 'resource sharing' an advantage of a thread or disadvantage of a thread. Explain yours answe*r briefly*.

Answer: handouts Page No. 70

The Advantages and Disadvantages of Threads

Four main advantages of threads are:

1. **Responsiveness**: Multithreading an interactive application may allow a program to continue running even if part of it is blocked or is performing a lengthy operation, thereby increasing responsiveness to the user.
2. **Resource sharing**: By default, threads share the memory and the resources of the process to which they belong. Code sharing allows an application to have several different threads of activity all within the same address space.
3. **Economy:** Allocating memory and resources for process creation is costly. Alternatively, because threads share resources of the process to which they belong, it is more economical to create and context switch threads.
4. **Utilization of multiprocessor architectures:** The benefits of multithreading of multithreading can be greatly increased in a multiprocessor environment, where each thread may be running in parallel on a different processor. A single threaded process can only run on one CPU no matter how many are available.

Multithreading on multi-CPU machines increases concurrency.

Some of the main disadvantages of threads are:

1. **Resource sharing:** Whereas resource sharing is one of the major advantages of threads, it is also a disadvantage because proper synchronization is needed between threads for accessing the shared resources (e.g., data and files).
2. **Difficult programming model:** It is difficult to write, debug, and maintain multi-threaded programs for an average user. This is particularly true when it comes to writing code for synchronized access to shared resources.

Question No.26 Write data structures for bakery algorithm?

Answer: handouts Page No. 103

The bakery algorithm is due to Leslie Lamport and is based on a scheduling algorithm commonly used in bakeries, ice-cream stores.

Question No.27 What is a batch system?
Answer: **Click here for detail**
A batch system is used to monitor and control the jobs running on a system. It enforces limits on runtime (walltime) as well as the number of jobs running at one time (both total and per user). To run a job, the batch system allocates the sources requested in the batch script, sets up an environment to run the job in (thus running the users .cshrc and .login files), and then runs the job in that environment. In this environment, standard out and standard error are redirected into files in the current working directory at the time the executable is actually run. The normal and recommended mode is to use all four cores on each node (XT4 node queues) or all eight cores on each node (XT5 node queues), that is 4 PEs is default for XT4 nodes and 8 PEs for XT5 nodes. The following example is valid both for XT4 and XT5 nodes:
Example :

```
#!/bin/sh
#PBS -N my_job
#PBS -j oe
#PBS -l walltime=1:00:00
#PBS -l mppwidth=256
#PBS -m e
#PBS -M user1@univ2.fi
#PBS -r n
cd $PBS_O_WORKDIR
aprun -n 256 ./program
```

Question No.28 What is the main disadvantage of semaphore?5marks
Answer: **Click here for detail**
Disadvantage of semaphore:-

- Simple algorithms require more than one semaphore.
- This increases the complexity of semaphore solutions to such algorithms.
- Semaphores are too low level.
- It is easy to make programming mistakes (e.g. P(s) followed by P(s)).
- The programmer must keep track of all calls to wait and to signal the semaphore.
- If this is not done in the correct order, programmer error can cause deadlock.
- Semaphores are used for both condition synchronization and mutual exclusion.
- These are distinct and different events, and it is difficult to know which meaning any given semaphore may have.

Question No.29 about spinlock?
Answer: **handouts Page No. 110**
Spinlocks are useful in Multiprocessor systems, and busy waiting wastes CPU cycles that some other process may be able to use productively. This type of semaphore is also called a spinlock (because the process spins while waiting for the lock)

Question No.30 which command is used for suspending a foreground process?
Answer: **handouts Page No. 66**
You can suspend a foreground process by pressing <Ctrl-Z>, which sends a STOP/SUSPEND signal to the process.

Question No.31 which command is used to resume the execution of a suspended job in the background?
Answer: **handouts Page No. 65**
You can use the fg command to resume the execution of a suspended job in the foreground or move a background job into the foreground. Here is the syntax of the command.
fg [%job_id]

Question No.32 How to implement RR scheduling?

Answer: handouts Page No. 86, 87

To implement RR scheduling, we keep ready queue as a FIFO queue of processes. New processes are added to the tail of the ready queue. The CPU scheduler picks the first process from the ready queue, sets a timer to interrupt after 1 time quantum, and then dispatches the process. One of the two things will then happen. The process may have a CPU burst of less than 1 time quantum, in which case the process itself will release the CPU voluntarily. The scheduler will then proceed to the next process in the ready queue. Otherwise, if the CPU burst of currently running process is longer than one time quantum, the timer will go off and will cause an interrupt to the operating system. A context switch will happen, the current process will be put at the tail of the ready queue, and the newly scheduled process will be given the CPU.

Question No.33Highlight critical sections of this code? (5 marks)

```
do
{
while (TestAndSet(lock)) ;
Critical section
lock=false;
Remainder section
} while(1);
```

Answer: handouts Page No. 107

The above TSL-based solution is no good because even though mutual exclusion and progress are satisfied, bounded waiting is not. The semantics of the Swap instruction, another atomic instruction, are, as expected, as follows:

```
boolean Swap(boolean &a, boolean &b)
{
boolean temp=a;
a=b;
b=temp;
}
```

If the machine supports the Swap instruction, mutual exclusion can be implemented as follows. A global Boolean variable lock is declared and is initialized to false. In addition each process also has a local Boolean variable key. The structure of process Pi is:

```
do
{
key=true;
while(key == true)
Swap(lock,key);
Critical section
lock=false;
Remainder section
} while(1);
```

Just like the TSL-based solution shown in this section, the above Swap-based solution is not good because even though mutual exclusion and progress are satisfied, bounded waiting is not. In the next lecture, we will discuss a good solution for the critical section problem by using the hardware instructions.

Question No.34

Analyze the given algorithm proposed to solve the critical section problem. Identify the shortcomings of this algorithm.

```
do{
while(turn!=j);
critical section
turn=j;
remainder section
```

Answer: handouts Page No. 100

This solution ensures mutual exclusion, that is only one process at a time can be in its critical section. However it does not satisfy the progress requirement, since it requires strict alternation of processes in the execution of the critical section. For example, if turn= =0 and P1 is ready to enter its critical section, P1 cannot do so even though P0 may be in its remainder section. The bounded wait condition is satisfied though, because there is an alternation between the turns of the two processes.

Question No.35
what assumptions should be kept in mind while formulating a solution to critical section problem?
Answer: Click here for detail
Following in mind before solving the formulating a solution to critical section problem:-

- Outline the general context of the problem area.
- Highlight key theories, concepts and ideas current in this area.
- What appear to be some of the underlying assumptions of this area?
- Why are these issues identified important?
- What needs to be solved?
- Read round the area (subject) to get to know the background and to identify unanswered questions or controversies, and/or to identify the most significant issues for further exploration.

Question No.36 Differentiate between independent processes and cooperative processes?
Answer: handouts Page No. 41

- a process which does not need any other external factor to trigger it is an **independent process.**
- a process which works on occurrence of any event and the outcome effects any part of the rest of the system is called a **cooperating process.**

Question No.37 Differentiate between fifo and pipe?
Answer: Click here for detail
The only difference between pipes and FIFOs is the manner in which they are created and opened. Once these tasks have been accomplished, I/O on pipes and FIFOs has exactly the same semantics.

Question No.38 What are the three requirements for the solution of critical section problems?
Answer: handouts Page No. 99
A solution to the critical section problem must satisfy the following three requirements:

1. Mutual Exclusion
2. Progress
3. Bounded Waiting

Question No.39
mkfifo() call may fail due to many reason. Describe the situations in which mkfifo() command will terminate and also describe the situations in which the mkfifo() command will not terminate
Answer: handouts Page No. 58
The mknod() call is normally used for creating special (i.e., device) files but it can be used to create FIFOs too. The 'mode' argument should be permission mode OR-ed with S_IFIFO and 'dev' is set to 0 for creating a FIFO. As is the case with all system calls in UNIX/Linux, mknod() returns –1 on failure and errno is set accordingly. Some of the reasons for this call to fail are:

- File with the given name exists
- Pathname too long
- A component in the pathname not searchable, non-existent, or non-directory
- Destination directory is read-only
- Not enough memory space available
- Signal caught during the execution of mknod()

Here is the synopsis of the mkfifo() library call.
#include <sys/types.h>

#include <sys/stat.h>
int mkfifo (const char *path, mode_t mode)

The argument path is for the name and path of the FIFO created, where was the argument
mode is for specifying the file permissions for the FIFO. The specification of the mode argument for this function is the same as for the open(). Once we have created a FIFO using mkfifo(), we open it using open(). In fact, the normal file I/O system calls (close(), read(), write(), unlink(), etc.) all works with FIFOs. Since mkfifo() invokes the mknod() system call, the reasons for its failure are pretty much the same as for the mknod() call given above.

Question No.40

What are the advantages of Round Ribon scheduling algorithm and what are the disadvantages of this scheduling algorithm?

Answer: **Click here for detail**

- The advantage of RR is that it is starvation-free since all process will always get a time share. The disadvantage is there is no prioritization.
- if short processes get higher CPU priority all the time, long processes will never finish. They'll just sit at the back of the queue.
- The advantage is that multiple processes will get done quickly, if they are all short. The disadvantage is that long processes will get done quickly, if there are no other processes running.

Question No.40 Highlight the shortcoming in the following hardware solution

```
do
{
while(testandset(lock));
critical section
lock=false;
remainder section
}while(1);
```

Answer: handouts Page No. 107

The above TSL-based solution is no good because even though mutual exclusion and progress are satisfied, bounded waiting is not. The semantics of the Swap instruction, another atomic instruction, are, as expected, as follows:

```
boolean Swap(boolean &a, boolean &b)
{
boolean temp=a;
a=b;
b=temp;
}
```

If the machine supports the Swap instruction, mutual exclusion can be implemented as follows. A global Boolean variable lock is declared and is initialized to false. In addition each process also has a local Boolean variable key. The structure of process Pi is:

```
do
{
key=true;
while(key == true)
Swap(lock,key);
Critical section
lock=false;
Remainder section
} while(1);
```

Just like the TSL-based solution shown in this section, the above Swap-based solution is not good because even though mutual exclusion and progress are satisfied, bounded waiting is not. In the next lecture, we will discuss a good solution for the critical section problem by using the hardware instructions.

Question No.41 How can you differentiate between a library call and a system call? Explain briefly giving at least one example.
Answer: Click here for detail
library Call:-

- Library calls are handled by a dynamic library. The program making the library call must first import that library, before the call will work. The library calls themselves may use system calls.
- Library calls (functions within program libraries)

System Call:-

- System calls (functions provided by the kernel)
- System calls are handled directly by the kernel.

Example
there is getmntinfo(3) and getfsstat(2), both look like they do the same thing.

Question No.42 Define waiting time with respect to CPU Scheduling?
Answer: handouts Page No. 30
This information includes a process priority, pointers to scheduling queues, and any other scheduling parameters.

Question No.43 what does the term 'input redirection' mean? Write down the basic syntax for input, output and error redirection of UNIX/LINUX system. 5 marks
Answer: handouts Page No. 55
Linux redirection features can be used to detach the default files from stdin, stdout, and stderr and attach other files with them for a single execution of a command. The act of detaching defaults files from stdin, stdout, and stderr and attaching other files with them is known as input, output, and error redirection. In this section, we show the syntax, semantics, and examples of I/O and error redirection.
Input Redirection: Here is the syntax for input redirection:
command < input-file
or
command 0< input-file

Question No.44 what are pros and cons of multithreading
Answer: Click here for detail
The purpose of building multi-threading applications when multi-threading should be used, the pros and cons for multi-threaded applications. The various multi-threading implementations, the pros and cons for each.

Question No.45 What is waiting time in context of CPU burst?2marks
Answer: Click here for detail
Burst time is an assumption of how long a process requires the CPU between I/O waits. It can not be predicted exactly, before a process starts. It means the amount of time a process uses the CPU for a single time. (A process can use the CPU several times before complete the job)
Question No.47 define data race detection is static and dynamic if it is define in detail
Answer: Click here for detail
Dynamic data race detectors are less prone to false warnings than static techniques because they monitor an actual execution of the program. However, they may miss races because successful detection might require an error-inducing input and/or an appropriate thread Schedule. Also, many dynamic detectors employ several heuristics and approximations that can lead to false alarms.

Question No.48 priority scheduling algorithm is indefinite blocking or starvation define it why? marks
Answer: handouts Page No. 84

Priority- scheduling algorithms is **indefinite blocking** (or **starvation**). A process that is ready to run but lacking the CPU can be considered blocked-waiting for the CPU. A priority-scheduling algorithm can leave some low priority processes waiting indefinitely for the CPU.

Question No.49 Define any three queues that are used in Multilevel Feedback scheduling
Answer: handouts Page No. 90
Multilevel Feedback Queue Scheduling
Multilevel feedback queue scheduling allows a process to move between queues. The idea is to separate processes with different CPU burst characteristics. If a process uses too much CPU time, it will be moved to a lower-priority queue. This scheme leaves I/O bound and interactive processes in the higher-priority queues. Similarly a process that waits too long in a lower-priority queue may be moved o a higher priority queue. This form of aging prevents starvation. In general, a multi-level feedback queue scheduler is defined by the following parameters:

- Number of queues
- Scheduling algorithm for each queue
- Method used to determine when to upgrade a process to higher priority queue
- Method used to determine when to demote a process
- Method used to determine which queue a process enters when it needs service

Question No.50 similarities between process and thread execution
Answer: Click here for detail

- Unlike processes, threads are not independent of one another.
- Unlike processes, all threads can access every address in the task.
- Unlike processes, thread are design to assist one other. Note that processes might or might not assist one another because processes may originate from different users.