City University

Faculty of Science & Engineering

Department of Computer Science and Engineering

Program: B.Sc. in CSE

Midterm Examination Semester: Spring 2022

Course Code: CSE 315 Course Title: Operating System

Total Marks: 30 Duration: 1 hour 30 Minutes

Answer any 3(three) questions

3 × 10 =30

1 (a) Write down the main purposes of an operating system. *3*

(b) Describe the main differences between operating system for mainframe computers and personal computers. *3*

(c) Briefly explain the main difficulty that a programmer must overcome in writing an operating system for a real-time environment. *4*

2 (a) Define the term process. *3*

(b) Discuss the following process state diagram with examples. *4*

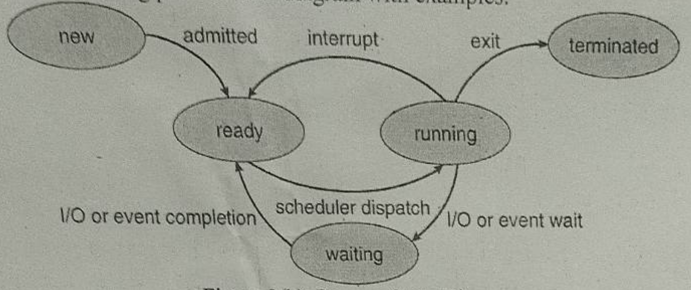

Figure 2(b): Process State Diagram

(c) Briefly describe the six transitions among the different states of the process state diagram. 3

3 (a) What is the difference between preemptive and nonpreemptive scheduling? 2

(b) Consider the following set of processes, with the length of CPU burst time given in milliseconds: 8

Process	Burst Time	Priority
P1	10	3
P2	1	1
P3	2	3
P4	1	4
P5	~~5~~ 4	~~2~~ 1

Time size (2)

The processes have arrived in the order P1, P2, P3, P4, P5, all at time 0. Draw Four Gantt charts illustrating the execution of these processes using FCFS, SJF, RR and priority scheduling. Calculate Average Waiting Time and Average Turn Around Time for them.

4 (a) Why multithreading is more beneficiary to the real time system? 2

(b) Write down the advantages of thread over process. 4

(c) Write down the differences between User Level Thread and Kernel Level Thread. 4

1. (**a)**Almost many purpose of operating system but there are some **main purpose of an operating system** such as:
•Handle Memory Management
•Perform all Loading and Execution task
•Peripheral Controlling

(b)What are the main differences between the operating system for mainframe computers and personal computers?
1) A mainframe will support multiple users. A personal computer originally supported only one user at a time (Windows can now have simultaneous login sessions).
2) Multitasking. A mainframe must be able to multitask between different threads. Today, PC OSs do as well but earlier versions didn't have true preemptive multitasking.
3) Remote access. Most mainframes can be remotely managed. No such requirement on a PC.
4) A mainframe should easily support a large number of processes. A PC might not be well suited to this.

(**c)**The main difficulty is keeping the operating system within the fixed time constraints of a real-time system. If the system does not complete a task in a certain time frame, it may cause a breakdown of the entire system it is running. Therefore when writing an operating system for a real-time system, the writer must be sure that his scheduling schemes don't allow response time to exceed the time constraint.

2.
(a)
Informally, a process is a program in execution. A process is more than the program code, which is sometimes known as the text section.

(b)
As a process executes, it changes state. The state of a process is defined in part by the current activity of that process. Each process may be in one of the following states:

- New: The process is being created.
- Running: Instructions are being executed.
- Waiting: The process is waiting for some event to occur (such as an I/O completion or reception of a signal).
- Ready: The process is waiting to be assigned to a processor.
- Terminated: The process has finished execution.

3.
(a)Key Differences Between Preemptive and Non-Preemptive Scheduling:

Parameter	**PREEMPTIVE SCHEDULING**	**NON-PREEMPTIVE SCHEDULING**
Basic	In this resources(CPU Cycle) are allocated to a process for a limited time.	Once resources(CPU Cycle) are allocated to a process, the process holds it till it completes its burst time or switches to waiting state.
Interrupt	Process can be interrupted in between.	Process can not be interrupted until it terminates itself or its time is up.
Starvation	If a process having high priority frequently arrives in the ready queue, a low priority process may starve.	If a process with a long burst time is running CPU, then later coming process with less CPU burst time may starve.
Overhead	It has overheads of scheduling the processes.	It does not have overheads.
Flexibility	flexible	rigid
Cost	cost associated	no cost associated
CPU Utilization	In preemptive scheduling, CPU utilization is high.	It is low in non preemptive scheduling.
Waiting Time	Preemptive scheduling waiting time is less.	Non-preemptive scheduling waiting time is high.
Response Time	Preemptive scheduling response time is less.	Non-preemptive scheduling response time is high.
Examples	Examples of preemptive scheduling are Round Robin and Shortest Remaining	Examples of non-preemptive scheduling are First Come First Serve and Shortest Job First.

Parameter	PREEMPTIVE SCHEDULING	NON-PREEMPTIVE SCHEDULING
	Time First.	

4.(**a)why multithreading is more beneficiary to the real time system?**
The benefits of multi threaded programming can be broken down into four major categories:

1.Responsiveness –

Multithreading in an interactive application may allow a program to continue running even if a part of it is blocked or is performing a lengthy operation, thereby increasing responsiveness to the user.
In a non multi threaded environment, a server listens to the port for some request and when the request comes, it processes the request and then resume listening to another request. The time taken while processing of request makes other users wait unnecessarily. Instead a better approach would be to pass the request to a worker thread and continue listening to port.

For example, a multi threaded web browser allow user interaction in one thread while an video is being loaded in another thread. So instead of waiting for the whole web-page to load the user can continue viewing some portion of the web-page.

2.Resource Sharing –

Processes may share resources only through techniques such as-

•Message Passing

•Shared Memory

Such techniques must be explicitly organized by programmer. However, threads share the memory and the resources of the process to which they belong by default.

The benefit of sharing code and data is that it allows an application to have several threads of activity within same address space.

3.Economy –

Allocating memory and resources for process creation is a costly job in terms of time and space.

Since, threads share memory with the process it belongs, it is more economical to create and context switch threads. Generally much more time is consumed in creating and managing processes than in threads.

In Solaris, for example, creating process is 30 times slower than creating threads and context switching is 5 times slower.

4.Scalability –

The benefits of multi-programming greatly increase in case of multiprocessor architecture, where threads may be running parallel on multiple processors. If there is only one thread then it is not possible to divide the processes into smaller tasks that different processors can perform.

Single threaded process can run only on one processor regardless of how many processors are available.

Multi-threading on a multiple CPU machine increases parallelism.

(**b)write down the advantages of thread over process.**

You'd prefer multiple threads over multiple processes for two reasons:

1. Inter-thread communication (sharing data etc.) is significantly simpler to program than inter-process communication.
2. Context switches between threads are faster than between processes. That is, it's quicker for the OS to stop one thread and start running another than do the same with two processes.

Advantages:

- Much quicker to create a thread than a process.
- Much quicker to switch between threads than to switch between processes.
- Threads share data easily

(c)write down the differences between user level thread and kernel level thread.

S. No.	**Parameters**	**User Level Thread**	**Kernel Level Thread**
1.	Implemented by	User **threads** are implemented by users.	Kernel threads are implemented by Operating System (OS).
2.	Recognize	Operating System doesn’t recognize user level threads.	Kernel threads are recognized by Operating System.
3.	Implementation	Implementation of User threads is easy.	Implementation of Kernel thread is complicated.
4.	Context switch time	Context switch time is less.	Context switch time is more.
5.	Hardware support	Context switch requires no hardware support.	Hardware support is needed.
6.	Blocking operation	If one user level thread performs blocking operation then entire process will be blocked.	If one kernel thread perform blocking operation then another thread can continue execution.
7.	Multithreading	Multithread applications cannot take advantage of multiprocessing.	Kernels can be multithreaded.
8.	Creation and Management	User level threads can be created and managed more quickly.	Kernel level threads take more time to create and manage.
9.	Operating System	Any operating system can support user-level threads.	Kernel level threads are operating system-specific.

S. No.	Parameters	User Level Thread	Kernel Level Thread
10.	Thread Management	The thread library contains the code for thread creation, message passing, thread scheduling, data transfer and thread destroying	The application code does not contain thread management code. It is merely an API to the kernel mode. The Windows operating system makes use of this feature.
11.	Example	Example: **Java thread**, POSIX threads.	Example: Window Solaris.
12.	Advantages	• User Level Threads are simple and quick to create. • Can run on any operating system • They perform better than kernel threads since they don't need to make system calls to create threads. • In user level threads, switching between threads does not need kernel mode privileges.	• Scheduling of multiple threads that belong to same process on different processors is possible in kernel level threads. • Multithreading can be there for kernel routines. • When a thread at the kernel level is halted, the kernel can schedule another thread for the same process.
13.	Disadvantages	• **Multithreaded applications** on user-level threads cannot benefit from	• Transferring control within a process from one thread to another necessitates a mode

S. No.	Parameters	User Level Thread	Kernel Level Thread
		multiprocessing. • If a single user-level thread performs a blocking operation, the entire process is halted.	switch to kernel mode. • Kernel level threads takes more time to create and manage than user level threads.