BUBT
Committed to Academic Excellence

BANGLADESH UNIVERSITY OF BUSINESS AND TECHNOLOGY

Midterm Answer Sheet

Course: Operating Systems

Course code: CSE 309

Submitted to:

Suman Saha

Assistant Professor

Dept. Of CSE

Bangladesh University of Business and Technology

Submitted by:

Syeda Nowshin Ibnat

ID: 17183103020

Intake: 39

Section: 1

Program: B.Sc. in CSE

Date of submission: 01/02/2021

Answer to the question 1(a)

Part-1

Soln:

In a system where a cpu can switch from one process to another to do execution is called multiprogramming system.

Issues related to multiprogramming system are discussing below:

1) Single user cannot keep CPU and I/O devices busy at all time.

Explanation: cpu cannot execute 2 or more jobs parallelly but can do concurrently. CPU will switch from one job to another. Single users frequently have multiple programs running. Multiprogramming increase cpu utilization by organizing jobs so that the cpu always has one to execute. The idea is as follows: The operating system keeps several jobs in memory simultaneously.

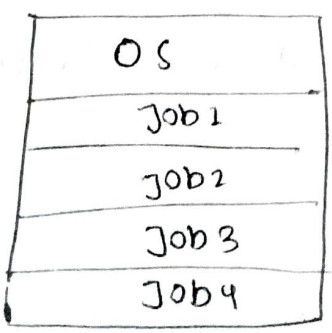


Fig: Memory layout of Multiprogram System.

2) Multiprogramming organizes jobs so CPU always has one to execute.

Explanation: Multiprogramming organizes jobs in the main memory in a certain way so that CPU will execute one job at a time. Because CPU can works with more than one job but can execute one job at a time.

3) A subset of total jobs in a system is kept in memory one job selected and run via job scheduling.

Explanation: The programs we want to execute, will have to kept on the main memory. Then CPU will execute them concurrently. For example, if there are 3 programs in main memory. So the degree of multiprogramming is 3. Now, if the number of program increase, then the degree of multiprogramming will also increase. And CPU will execute one program at a time.

4) When ~~if CPU~~ it has to wait (For I1O as example), OS switches to another job.

(3)

Explanation: The set of jobs in memory can be a subset of the jobs kept in the job pool. The operating system picks and begins to execute one of the jobs in memory. Eventually, the job may have to wait for some task, such as an I/O operation, to complete. In a non-multiprogrammed system, the CPU would sit idle. In a multiprogramming system, the operating system simply switches and executes another job. When that job needs to wait CPU switches to another job and so on. Eventually, the first job finishes waiting and gets the CPU back.

Part-2

Handheld electronic devices such as tablets and smartphones continue to offer amazing features. The challenges a developer will be face for designing the operating system of handheld devices are given below:

1) Display size: Smart phone or tablet's display size is smaller then computer or laptop.

2) Less storage capacity: It means the operating system

has to manage memory carefully.

3) Power consumption: The operating system must also manage power consumption carefully for handheld device.

4) Processing power: Less processing power plus fewer processors mean the operating system must carefully appontion.

5) Battery run time is a major consideration in handheld device ~~desi~~ OS design.

6) To develop an OS for handheld device, development cost is high. The resulting must be affontable for the customers to purchase.

Answer to the question 1(b)

Soln: One of the main task of an operating system is to ensure connect operation of the computer system. To achieve this, some facilities are generaly provided by an operating system.

(1)

CPU and number of device controllers that are connected through a common bus that provides access to shared memory. I/O devices and CPU can execute concunnently,

computing for memony cyeles.

(2)

Each device controller is in charge of a specific type of device (Ex: disk drive, audio device etc.). The CPU and device controllers can execute in parallel, competing for memory cycles.

(3)

For a computer to start running - for instance, whe it is powered up or rebooted it needs to have an initial program to run. This initial program or bootstrap program, tends to be simple.

(4)

When interrupt occurs, Interrupt transfers control to the appropriate interrupt service routine. While an ~~intrup~~ interrupt is services, typically other interrupts cannot come, or they are disabled. Mostly mondern OS are interrupt driven.

(5)

Handled interrupts: When an interrupt occurs hardware transfer control to OS. OS preserve the state of the CPU

by storing the registers and the program counter. Then it determines what type of interrupt has occurred and the corresponding code in OS is executed depending upon interrupt.

Other than these: 6) Each device controller has a local buffer.

7) CPU moves data from/to main memory to/from local memory.

8) I/O is from the device to local buffer of controller. etc.

Answer to the question 2(a)

Soln: A Process switch may occure any time that the operating system has gained control from the currently running process. At some time, a running process is interrupted and the operating system assigns another process to the running state and turns control over the process.

Now, considering there are 3 processes; process A, Process B and process C.

The following diagram ~~dig~~ depicts the process of context switching ~~be~~ among 3 processes: Process A, Process B and C.

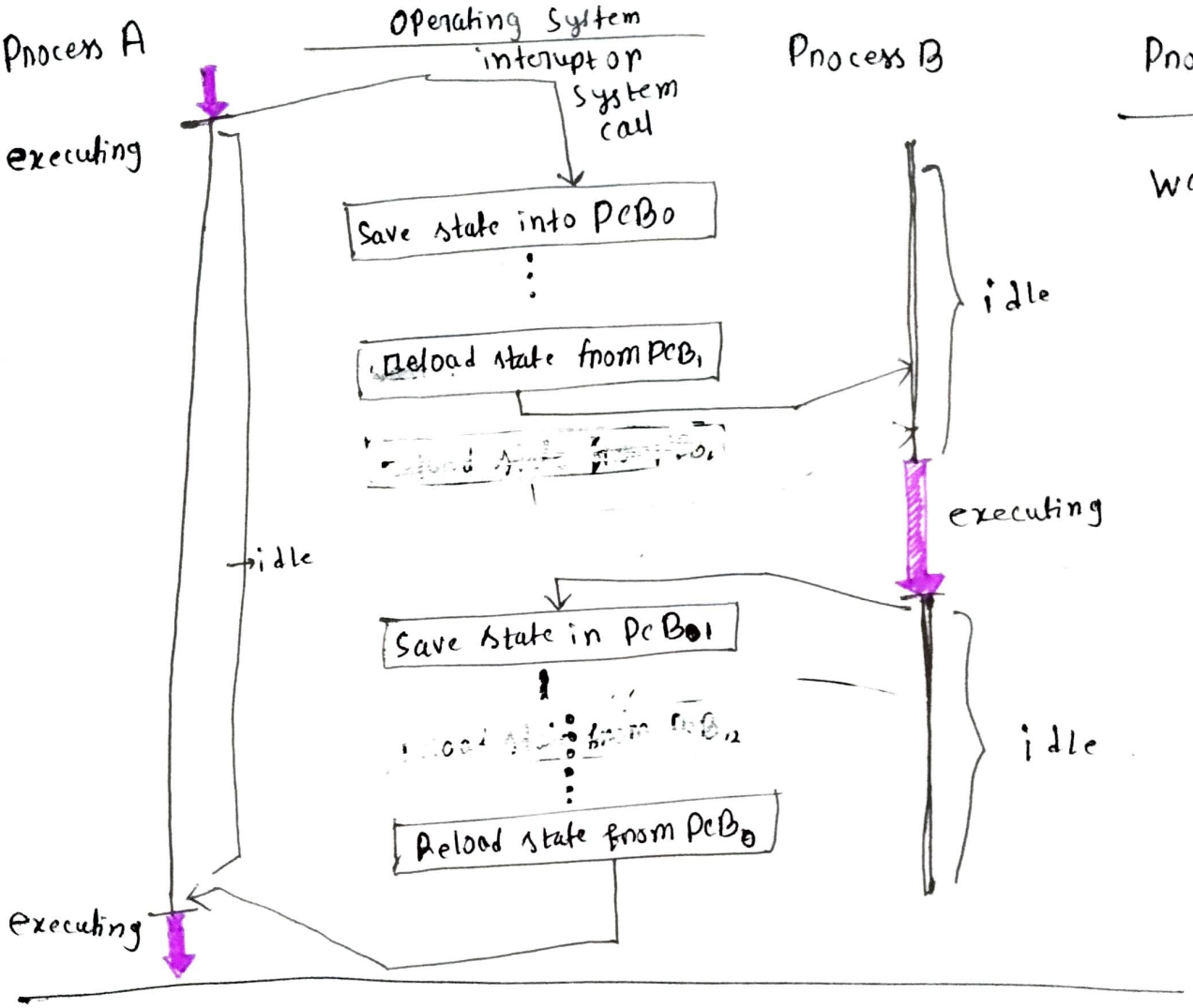


Fig: CPU switch from process to process

In the above figure we can see that initially, the ~~Process: A, B and C~~ process A is in the running state and process B is in the ready state and C is in waiting state. Now, when some ~~process~~ interruption

occures then we have to switch the the process A from running to ready state after saving the context and the process B from ready to running state. And process C waiting to ready state. The steps are:

(1) Finstly, the context of the process A i.e the process present in the running state will be saved in the process control block of process A i.e. PCB1.

(2) From the ready state, we have to select the new process that is to be executed.

(3) Now, we have to Update the PCB of process B. by setting the process state running.

(4) Similarly, if we want to run process A again, then have to follow the previous steps.

(5) Now to execute process C, at finst we have to move the PCB to relevent queue i.e waiting queue.

(6) For context switching to happen, two process are at least required. Here we did it for 3 process: A, B, C.

Answer to the question 2(b)

Sol^n:

Part-1

Process

Process means any program is in execution.

About process:

(1) It takes more time for creation.

(2) It takes huge or more time then threads for context switching

(3) Process is less efficient in term of communication.

(4) Process is called heavy weight process.

Threads

Threads is are the segment of a process means a process can have multiple threads.

(1) It takes less time for creation.

(2) Thread consume less resources.

(3) Threads takes less time to terminate as compared to process and like process threads do not isolate.

So, transition between threads ~~my~~ may be more economical and faster than transition between process.

Part-2

Some system don't allow a child to exist if it's parent has terminated:

In such systems, if a process terminates, then all it's children must also terminated. This phenomenon, refenned to as cascadind termination, is nonmally initiated by the OS. To illustnate process execution and terminration, consider that, in linux and UNIX systems, we can terminate a process by using the exit() system call, providing an exit status as a parameter:

```
/* exit with status 1 */
exit (1);
```

In fact, under nonmal termination, exit() may be called either directly (as shown above) or indirectly. A parent process may wait for the termination of a child process by using

the wait() system call. The ~~wait() syst~~ wait() system call is passed a parameter that allows the parent to obtain the exit status of the child. This system is also called for returns the process identifier of the terminated child so that the parent can tell which of its children has terminated:

```
pid_t pid;
int status;
pid = wait(&status);
```

When a process terminates, its resources are deallocated by the OS. However, its entry in the process table must remain there until the parent calls wait(), because the process table contains the process's exit status. A process that has terminated, but whose parent has not yet called wait(), is known as a zombie process. Now, if a parent did not invoke wait() and instead terminated, thereby leaving its child processes as orphans.

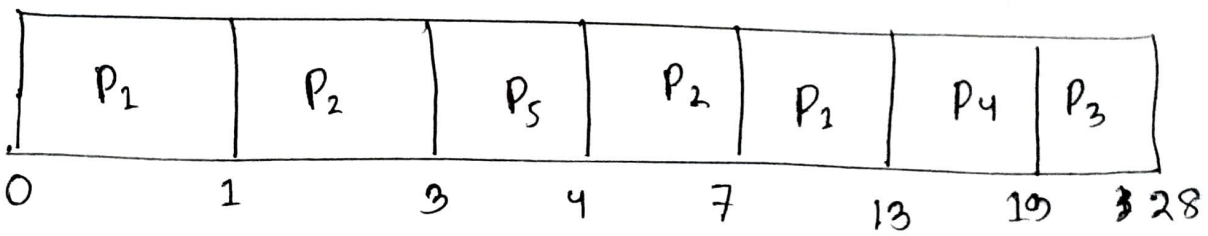
Answer to the question 3(a)

Soln:

(i)

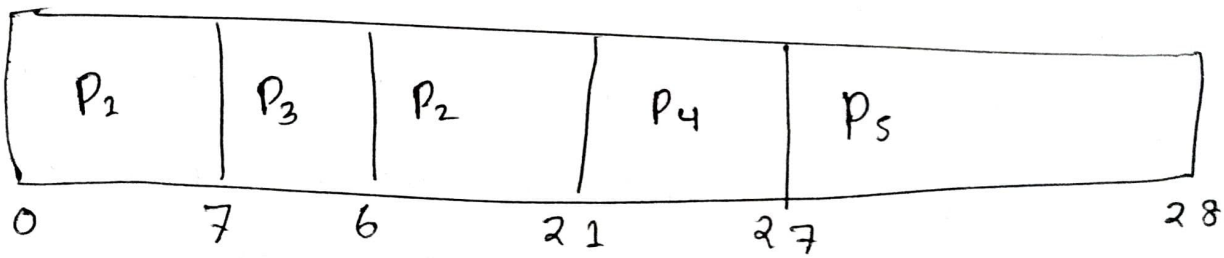
Pneemptive SJF

Gantt chart



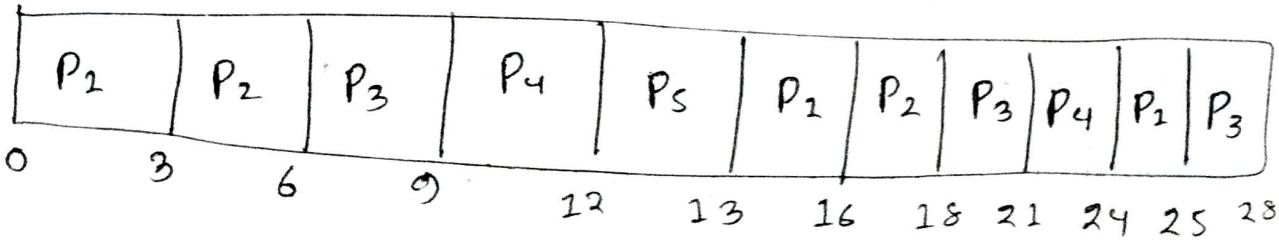
Non-pneemptive Priority

Gantt chart :



Round Robin (T.Q = 3)

Gantt chart :



(ii) (For Round Robin)

We know.

Waiting time = End time - Arrival time - Burst time

for, $P_1 = 24 - 0 - 7 = 17$

$P_2 = 17 - 1 - 6 = 10$

$P_3 = 27 - 1 - 9 = 17$

$P_4 = 23 - 2 - 6 = 15$

$P_5 = 12 - 3 - 1 = 8$

Avg. waiting time = $(17 + 10 + 17 + 15 + 8) / 5$

$= 67/5$

$= 13.4$

(For preemptive priority)

Waiting time,

for. $P_1 = 27 - 0 - 7 = 20$

$P_2 = 15 - 1 - 5 = 9$

$P_3 = 10 - 1 - 9 = 0$

$P_4 = 21 - 2 - 6 = 13$

$P_5 = 28 - 3 - 1 = 24$

Avg. waiting time = $(20 + 9 + 0 + 13 + 24) / 5$

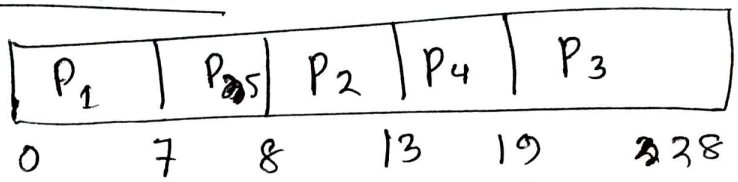
$= 66/5 = 13.2$

Ans:

(iii)

Non-preemptive SJF

Grantt chart



We know, turn around time = completion time − arrival time

and Response time = Start time − Arrival time

	Start time	completion time	Arrival time	Burst time	Turn around t.	Response time
P_1	0	7	0	7	7	0
P_2	8	13	1	5	12	7
P_3	19	28	1	9	27	18
P_4	13	19	2	6	17	11
P_5	7	8	3	1	5	4
					total = 68	total = 40

$\therefore$ Avg. Turnaroun time $= \frac{68}{5} = 13.6$

and $\therefore$ Avg. Response time $= \frac{40}{5} = 8$

Ans

Answer to the question 3(b)

Considering the given scenarios where we have 8 process having burst time of 3ms, 2, 8, 4, 10, and 15 ms, 5 ms and 25 ms respectively. For this i will prefer to use Shortest-job-first (SJF) scheduling. Because this scheduling algorithm works for or works with burst time. SJF is an algorithm in which the process having the smallest execution time is chosen for the next process execution. This Scheduling method can be preemptive and non-preemptive. It significantly reduces the average time for other processes awaiting for execution.

- It is associated with each job as a unit of time to complete.
- It improves job output by offering shorter jobs, which should be executed first, which mostly have a shorter turnaround time.

So, I will prefer SJF, hence the task is to determine avg. waiting time.