Operating System

Assignment-1.

1. Describe the action taken by a kernel to context switch between processes.

Ans. 1. In response to a clock interrupt, the OS saves the PC and user stack pointer of currently executing process and transfers control to the kernel clock interrupt handler.

2. The clock interrupt handler saves the rest of the registers, as well as other machine state, such as the state of floating point registers, in the process PCB.

3. The OS invokes the scheduler to determine the next process to execute.

4. The OS then retrieves the state of the next process from its PCB, & restores the registers. This restore operation takes the processor back to the state in which this process was previously interrupted, executing in user code with user mode privileges.

2. Describe the difference among short-term, medium-term, & long-term scheduling.

Ans. **Short-term (CPU scheduler):-** selects a process from those that are in memory & ready to execute, and allocates the CPU to it.

Mediom-term (memory manager):- selects process from the ready or blocked queue & removes them from

memory, then reinitiates them later to continue running.

Long term scheduler :- (Job scheduler)

determines which jobs are brought into the system for processing

The primary difference is in frequency of their execution The short-term must select a new process quite often Long-term is used much less often since it handles placing jobs in the system & may wait for a job to finish before it admits another one.

3. List the four steps that are necessary to run a program on a completely dedicated machine a computer that is running only that program.

Ans. The steps are as follows:-

1. Reserve machine time.
2. Manually load program into memory.
3. Load starting address & begin execution.
4. Monitor & control execution of program from console.

4. In a multiprogramming & time-sharing environment several users share the system simultaneously. This situation can result in various security problems.

a) What are 2 such problems?

Ans. Some of the problems are :-

1. Privacy: One user can read the private data of another user.
2. Integrity: One user can corrupt the private data of another users.
3. Denial of service: One user can prevent another user from getting anything done.

b.) Can we ensure the same degree of security in a time-shared machine as we have in a dedicated machine? Explain it.

Ans. There are two answers for this, either one is correct.

Yes:- If we can ensure that the OS prevents any sharing of data b/w users, either for reading or writing, and fairly shares the computer, then we can achieve the same level of security.

No:- We can never be sure that our software doesn't have bugs, so we can never be sure that we prevent all sharing of data & fairly allocate computer resources.

5. Some computer systems do not provide a privileged mode of operation in hardware. Is it possible to construct a secure operating system for these computer systems? Give arguments both that it is & that it is not possible.

Ans. An OS for a machine of this type would need to remain in control at all times. This could be accomplished by two methods.

i. Software interpretation of all user programs (like some BASIC, JAVA & LISP system, for example). The software interpreter would provide, in software, what the hardware doesn't provide.

ii. Requirement that all programs be written in high level language so that all object code is compile produced. The compiler would generate the protection checks that the hardware is missing.

6. Define the essential properties of following types of OS.

a.) Batch :-

Jobs with similar needs are batched together & run through the computer as a group by an operator or automatic job sequences. Performance is increased by attempting to keep CPU and I/O devices busy at all times through buffering, off-line operation spooling, & multi-programming. Batch is good for executing large jobs that has little interaction; it can be submitted & picked up later.

b.) **Interactive :-**

This system is composed of many short transactions where the results of next transcation may be unpredictable. Response time needs to be short (seconds) since the user submits and waits for result.

c.) **Time sharing :-**

This system uses CPU scheduling and multiprogramming to provide economical interactive use of a system. The CPU switches rapidly from one user to another. Instead of having a job define by spooled card images, each program reads its next control card from the terminal and output is normally printed immediately to the screen.

d.) **Real time :-**

Often used in a dedicated application this system reads information from sensors & must respond within a fixed amount of time to ensure correct performance.

e.) **Network :-** provide OS features across a network such as file sharing.

f.) **Parallel :-** Used in systems where there are multiple CPU's each running the same copy of OS. Comm^n takes place across the system.

g.) **Distributed :-**

This system distributes computation among several physical processors. The processors don't share memory or a clock. Instead, each processor has its own local memory. They communicate with each other through various comm^n lines, such as high-speed bus or local area network.

h.) **Clustered :-**

A cluster system combines multiple computers into a single system to perform computer task distributed across the cluster.

i.) **Handheld :-**

A small computer system that performs simple tasks such as calendars, email & web browsing. Handheld system differ from traditional desktop system with smaller memory & display screens & slower processors.

7. Describe a mechanism for enforcing memory protection in order to prevent a program from modifying the memory associated with other programs.

Ans. The processor could keep track of what locations are associated with each process & limit access to locations that are outside of a program's extent. Information regarding the extent of a program's memory could be maintained by using base & limits registers and by performing a check for every memory access.

8. Describe how you could obtain a statical profile of the amount of time spent by a program executing different section of its code. Discuss the importance of obtaining such a statical profile.

Ans. One could issue periodic timer interrupts and monitor what instructions or what sections of code are currently executing when the interrupts are delivered. A statistical profile of which piece of code were active should be consistent with the time spent by the program in different section of its code. Once such a statical profile has been obtained, the programmer could optimize those sections of code that are consuming more of the CPU resources.

9. How could a system be design to allow a choice of OS from which to boot ? What would be bootstrap program need to do?

Ans. Consider a system that would like to run both Window XP and three different distributions of Linux. Each OS will be stored on disk. During system boot up, a special program will determine which OS to boot into. This means that rather initially booting to an OS, the boot manager will first run during system startup. It is this boot manager that is responsible for determining which system to boot into. Typically, boot manager must be stored at certain location of hard disk to be recognized during system startup. Boot managess often provide the user with

a section of systems to boot into; boot managers are also typically designed to boot into a default OS, if no choice is selected by the user.

10. What is the purpose of the command interpreter? Why is it usually separate from the kernel? Would it be possible for the user to develop a new command interpreter using the system call interface provided by the OS?

Ans. An interrupt is a hardware-generated change of-flow within the system. An interrupt handler is summoned to deal with the cause of the interrupt; control is then returned to the interrupted context and instruction. A trap is a software-generated interrupt. An interrupt can be used to signal the completion of an I/O to obviate the need for device polling. A trap can be used to call OS routines or to catch arithematic errors.

11. It is sometimes difficult to achieve a layered approach if two components of the OS are dependent on each other. Identify a scenario in which it is unclear how to layer 2 system component that require tight coupling of their function-abilities.

Ans. • The virtual memory subsystem and the storage subsystem are typically tightly coupled and require careful design in a layered system due to the following interactions.

- Many systems allow files to be mapped into the virtual memory space of an executing process. On the other hand, the virtual memory subsystem typically uses the storage system to provide the backing store for pages, that do not currently reside in memory.

- Also, updates to the file system are sometimes buffered in physical memory before it is flushed to disk, thereby requiring careful co-ordination of the usage of memory b/w the virtual memory subsystem and the file system.

Assignment - 3

1. A CPU-scheduling algo. determines an order for the execution of its scheduled processes. Given n-proces to be scheduled on one processor, how many different schedules are possible?

Ans. n! schedules are possible.

$\Rightarrow n \times (n-1) \times (n-2) \times \dots \times 2 \times 1$.

2. Suppose that a scheduling algo (at level of short term CPU scheduling) favours those process that have used the least processor time in the recent past. Why will this algo. favour I/O bound programs and yet not permanently starve CPU-bound programs?

Ans. It will favour the I/O bound programs because of the relatively short CPU burst request by them, however, the CPU-bound programs will not starve because the I/O bound programs will relinquish the CPU relatively often to do their I/O.

3. Consider a system running I/O bound tasks and one CPU-bound task. Assume that the I/O bound tasks issue an I/O operation once for every msec. of CPU computing & that each I/O operation takes 10 msec. to complete. Also assume that context switching overhead is 0.1 msec & that all processes are long running tasks.

Describe the CPU utilization for a round robin scheduler when:

a) The time quantum is 1 msec.

b) The time quantum is 10 msec.

Ans.

a) Irrespective of which process is scheduled, the scheduler incurs a 0.1 msec, context switching cost for every context-switch. The result in a CPU utilization of $\frac{1}{1.1} \times 100 = 91\%$.

b) The I/O bound tasks incur a context switch after using up only 1 msec, of time quantum. The time required to cycle through all the processes is therefore $10 * 1.1 + 10.1$.

The CPU utilization is

$$\frac{20}{21.1} \times 100 = 94\%.$$

4. Explain the difference in how much the following scheduling algo discriminates in favour of short processes.

a) FCFS :-

discriminates against short jobs since any jobs arriving after long jobs will have a long waiting time.

b.) **RR :-**

treats all jobs equally so short jobs will be able to leave the system faster since they will finish first.

c) **Multilevel feedback queues :-**

It works similar to the RR algo., they discriminate favorable towards short jobs.

5. Consider a system implementing multilevel queue scheduling. What strategy can a computer user employ to maximize the amount of CPU time allocated to the user's process?

Ans. The program could maximize the CPU time allocated to it by not fully utilizing its time quantums. It could use a large fraction of its assigned quantum, but relinquish the CPU before the end of the quantum, therefore, increasing the priority associated with the process.

For round-robin, its okay to answer that there is no strategy exist, or another would be to break a process up into many smaller processes & dispatch each to be strated and scheduled by the OS.

6. Consider the exponential avg. formula used to predict the length of next CPU burst. What are the implications of assigning the following values to the parameters used by the algo.?

a) $\alpha = 0$ & $\tau_0 = 100$ msec.

b) $\alpha = 0.99$ & $\tau_0 = 10$ msec.

Ans. When $\alpha = 0$ & $\tau_0 = 100$ msec, the formula makes a prediction of 100 msec for the next CPU burst.

When $\alpha = 0.99$ & $\tau_0 = 10$ msec, the most recent behaviour of the process is given much higher weight that the past history associated with the process, Consquently, the scheduling algo. is almost memoryless and simply predicts the length of the previous burst the next quantum of CPU execution.

7. Consider a variant of the RR scheduling algorithm in which the entries in the ready queue are pointers to the PCBs.

a) What would be the effect of putting two pointers to the same process in the ready queue?

Ans. In effect, that process will have increased its priority since by getting time more often it is receiving preferential treatment.

b) What would be two major advantages and two disadvantages of this scheme?

Ans. The advantage is that more important jobs could be given more time, in other words, higher priority in treatment. The consequence, of course, is that shorter jobs will suffer.

c) How would you modify the basic RR algo to achieve the same effect without the duplicate pointers?

Ans. Allot a longer amount of time to processes deserving higher priority, on other words, have two or more quantums possible in the RR scheme.

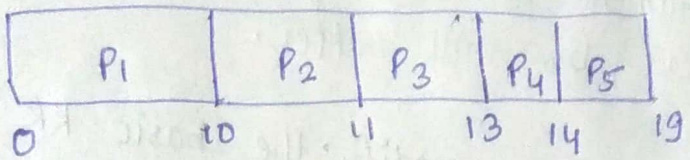
8. Consider the following set of processes, with the length of the CPU brust given in ms.

Process	B.T.	Priority
P_1	10	3
P_2	1	1
P_3	2	3
P_4	1	4
P_5	5	2

The processes are assumed to have arrived in order P_1, P_2, P_3, P_4, P_5 all at $t=0$.

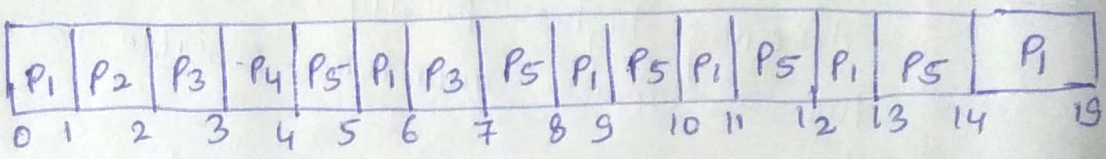
a) Draw four Gantt charts to illustrate the execution of these processes using the following scheduling algo.

1) FCFS.

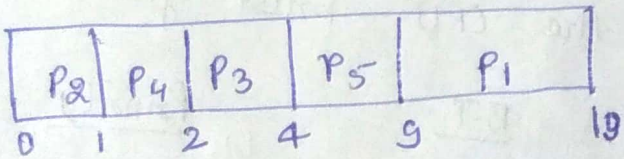


2) RR.

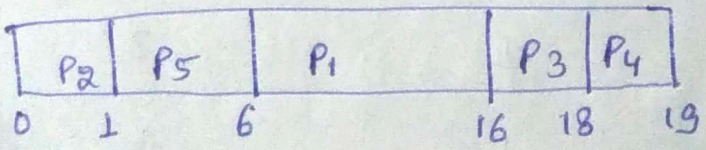
q=1.



SJF.



Priority.



b.) What is turn around time of each process for each of scheduling algo in Part (a)?

Ans.

Process	Algo FCFS	RR	SJF	Priority
P_1	10	19	19	16
P_2	11	2	1	1
P_3	13	7	4	18
P_4	14	4	2	19
P_5	19	14	9	6

c.) what is waiting time of each process for each algo?

Ans. Waiting time = Turn around time − Brust time

Process	Algo FCFS	RR	SJF	Priority
P_1	0	9	9	6
P_2	10	1	0	0
P_3	11	5	2	16
P_4	13	3	1	18
P_5	14	9	4	1

d.) Which of algo. result in min. avg. waiting time (on all process)?

Ans. SJF (Shortest job first) scheduling algorithm.
with avg. waiting time = $\frac{16}{5} = 3.2$ unit time.

9. Consider a pre-emptive priority scheduling algo. based on dynamically changing priority. Large priority not imply higher priority.

a.) what is the algo. that results from $\beta > \alpha > 0$?

Ans FCFS (first come first serve) scheduling algorithm.

$$[P(t) = a*(t - t_0)]$$

$$\text{Eqn}^n :- \quad P(t) = P_0 + a_0(t - t_0)$$

Since process start at zero, ($P_0 = 0$), any process that have been in the system (either running or waiting) have higher priority. Therefore, new processes go to the back of the queue. When a process runs, its priority keeps increasing at the rate of β, which is more of an increase than for the processes in the ready queue. Therefore, every time the process has the time run out. It goes to the front of the

ready queue and gets dispatched again. This is equivalent FCFS.

b) What is the algo. that results from $\alpha < \beta < 0$?

Ans. LCFS (Last Come First Serve) Scheduling algo.

This time, any new process entering the system have higher priority than any old ones, since priority starts at zero & then become negative when the process waits or runs. New process go in at the front of the queue, when a process runs or waits its priority decrease, with the waiting processes decreasing faster than the running process. This is a LIFO algo.

10. Suppose that following process arrive for execution at time indicated, each process will run for amount of time listed.

Process	Arrival Time	B.T.
P_1	0.0	8
P_2	0.4	4
P_3	1.0	1

a) What is the avg. turn around time for these process with the FCFS scheduling algo?

Soln:- $\left(\left(P_1\text{'s } W.T + P_1\text{'s brust time}\right) + \left(P_2\text{'s } W.T + P_2\text{'s } B.T\right) + P_3\text{'s } W.T + P_3\text{'s } B.T\right) \Big/ 3.$

$\Rightarrow \left((0+8) + (7.6+4) + (11+1)\right) \Big/ 3$

$= 10.533$ time unit.

b.) What is avg. turnaround time with SJF sc. algo?

Soln:- $\left((0+8) + (8.6+4) + (7+1)\right) / 3 = 9.533$

Ans → 9.533 Unit time.

c.) The SJF algo is supposed to improve performance, but notice that we close to run process P1 at time 0 bcz we did not allow that two shorter process would arrive soon. Compute what the avg. turnaround time will be if CPU is left idle for the first 1 unit & then SJF scheduling is used. Remember that Processes P_1 & P_2 are waiting during this idle time, so waiting time may increase. This algo. could be used as future-knowledge scheduling.

Soln:- Avg. turnaround time $= \left((6+8) + (1.6+4) + (0+1)\right) / 3$

$= 6.866$ time unit.