AUTUMN MID SEMESTER EXAMINATION-2018

School of Computer Engineering

KIIT Deemed to be University, Bhubaneswar-24

SOLUTION AND EVALUATION SCHEME

Operating Systems

[CS3009]

Q1) Answer all questions.

(a) Describe the importance of medium term scheduler with relation to long term scheduler.

Solution/Evaluation Scheme:

- The main role of the long term scheduler is to control the multi-programming which can be one way achieved by allowing multiple processes to reside in main memory by swapping out processes that are waiting (need I/O) or low priority processes and swapping in other processes that were in ready queue. This swap out and swap in can be done by medium term scheduler. **[1 mark]**

(b) Consider a method used by processes P1 and P2 for accessing critical section. Initial values of shared Boolean variable S1 and S2 are randomly assigned. The code for P1 and P2 is as follows:

P1	P2

while(S1==S2);
 	
Critical Section
 	
S1=S2;
	while(S1!=S2);
	
	Critical Section
	
	S2=not(S1);

Which required conditions are satisfied by the above mentioned solution for critical section ?

Solution/Evaluation Scheme:

Mutual Exclusion is satisfied as both processes cannot enter into the critical section at the same time
Progress is not satisfied as a process cannot re-enter into the critical section more than once even if the other process is executing in the reminder section.
Bounded waiting is not satisfied. **[1 mark]**

(c) Consider the below mentioned code.

```
Semaphore wrt = 1, mutex1=1, mutex2=1;
int readcount = 0;
```

Writer()	Reader()

Writer()
 	
{
 	
11. P(wrt);
12. ***write();***//Writing is done here
13. V(wrt);
 	
}
	Reader()
	{
	1. P(mutex1);
	2. readcount++;
	3. if (readcount == 1)
	{
	4. P(wrt);
	}
	V(mutex1);
	5. ***read();***//Reading is done here
	6. P(mutex2);

	7. readcount--;
	8. if (readcount == 0)
	{
	9. V(wrt);
	}
	10. V(mutex2);
	}

Assuming that there are *n* number of readers ($R_1...R_n$) and *n* number of writers ($W_1....W_n$). Using the above code, point out the code (line no) where mutual exclusion is violated.

Solution/Evaluation Scheme:

Since in the reader code the two semaphore blocks , i.e. mutex1 and mutex2, are manipulating a single shared variable called ***readcount***, so mutual exclusion is violated at certain situations. **[1 mark]**

(d) "Round Robin Scheduling is most suitable for interactive processes". Justify.

Solution/Evaluation Scheme:

Round Robin Scheduling is most suitable for interactive processes as every process is bound to get the CPU attention after a fixed interval of time. If there are *n* processes and *t* is the length of time slice, then every process will get the CPU after maximum (n-1)×t time units. **[1 mark]**

(e) State the advantage of monitor over semaphore.

Solution/Evaluation Scheme:

monitor is a high level synchronization tool as opposed to semaphore which is a programmer level tool. The human errors like misplacing the wait and signal operations are not observed with monitor.
[1 mark]

Q2) (a) What is a Process Control Block? What information about a process are stored in it? Justify.

Solution/Evaluation Scheme:

Explanation: OS Keeps information about the process for identification. **[1 mark]**

Various information like process id, state, PC value etc **[1.5 mark]**

(b) State different criteria for evaluating CPU scheduling algorithms and evaluate various CPU scheduling algorithms based on the criteria's.

Solution/Evaluation Scheme:

Explanation of different criteria like CPU Utilization, Throughput, Turn around time **[1 mark]**

Evaluation of algorithms like: SJF :Waiting time

Round robin: Response time **[1.5 marks]**

Q3) Assuming following processes arrive in a system with one processor.

Process Name	Expected Execution Time	Arrival Time	Priority
A	60	0	4
B	20	10	3
C	10	10	2

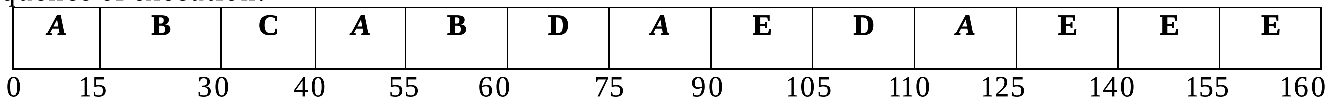
D	20	40	4
E	50	65	1

What will be the waiting time and sequence of execution of processes for the following CPU scheduling algorithms.

(a)RR scheduling with a quantum time 15.

Solution/Evaluation Scheme:

Sequence of execution:



<u>Waiting Time</u>

A=0+25+20+20=65
B=5+25=30
C=20
D=20+30=50
E=25+20=45

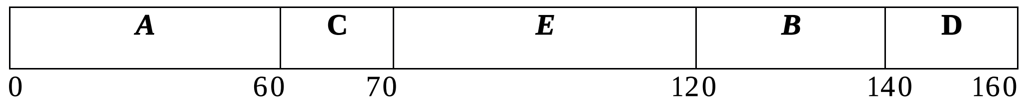
[**2.5 marks**]

(b) Priority scheduling algorithm.(**Note:** ***consider lowest number as highest priority***)

Solution/Evaluation Scheme:

<u>Non-Preemptive</u>

Sequence of execution:

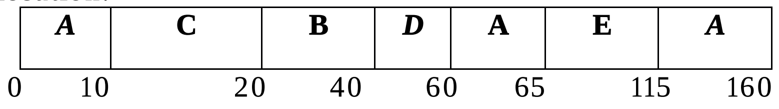


<u>Waiting Time</u>

A=0
B=120-10=110
C=60-10=50
D=140-40=100
E=70-65=5

<u>Preemptive</u>

Sequence of execution:



<u>Waiting Time</u>

A=0+50+50=100
B=10
C=0
D=0
E=0

[**2.5 marks**]

Partial marks may be awarded by considering the sequence of execution and correctness of the solution.

Q4) Consider two processes P0 and P1 sharing a Boolean array ***flag[2]*** initialized to false. Verify whether the below mentioned code for the processes satisfy the conditions needed for solution to critical section problem. (Assume ***random_sleep()*** makes a process suspended for a finite time).

P0	P1

flag[0]=true;
while(flag[1])
{
    flag[0]=false;
    random_sleep();
    flag[0]=true;
}
Critical Section
flag[0]=false;
	flag[1]=true;
	while(flag[0])
	{
	    flag[1]=false;
	    random_sleep();
	    flag[1]=true;
	}
	Critical Section
	flag[1]=false;

Solution/Evaluation Scheme:

Mutual exclusion

Mutual exclusion is satisfied as when a process ***i*** enters into the critical section, at that time fgag[j] must be false and flag[i] must be true. During that time other process called ***j*** can only be enter into critical section if flag[i] is false. This flag[i] can only be false when process i will come out from the critical section.
As per the while condition's position, flag[i] and flag[j] cannot be false at that position at any particular time (As the process has just made it true at the last statement of loop body iteration before checking the while condition).

P0	P1

flag[0]=true;
while(flag[1])
{
    flag[0]=false;
    random_sleep();
    flag[0]=true;
}
Critical Section
flag[0]=false;
	flag[1]=true;
	while(flag[0])
	{
	    flag[1]=false;
	    random_sleep();
	    flag[1]=true;
	}
	Critical Section
	flag[1]=false;

Progress

The progress condition that a process who is executing in the reminder section should not influence the decision whether an interested process will enter into the critical section or not is satisfied. This is because as long as a process is executing in the reminder section(Assuming it is after setting the own flag to false), its flag is false so the other process can enter into its ctitical section any number of times.

P0	P1
flag[0]=true;	flag[1]=true;
while(flag[1])	while(flag[0])
{	{
flag[0]=false;	flag[1]=false;
random_sleep();	random_sleep();
flag[0]=true;	flag[1]=true;
}	}
Critical Section	**Critical Section**
flag[0]=false;	flag[1]=false;
Reminder section	Reminder section

But the second requirement of progress that the selection must be done in a finite time is also not ensured. If both the processes will set their flag to true at the same time, they will enter into the loop body, but if both of them always make the flag to true and check the condition at the same time, the decision will be delayed for ever. So the progress condition is violated.

P0	P1
flag[0]=true;	flag[1]=true;
while(flag[1])	while(flag[0])
{	{
flag[0]=false;	flag[1]=false;
random_sleep();	random_sleep();
flag[0]=true;	flag[1]=true;
}	}
Critical Section	**Critical Section**
flag[0]=false;	flag[1]=false;
Reminder section	Reminder section

<u>Bounded Waiting</u>

Bounded waiting is not satisfied as one process may keep on waiting and the other process may enter into the critical section multiple times. This will happen if there is a context switch after making the flag to false in the loop body and at the same time the other process checks the while condition to enter into its critical section.

P0	P1
flag[0]=true;	flag[1]=true;
while(flag[1])	while(flag[0])
{	{
flag[0]=false;	flag[1]=false;
random_sleep();	random_sleep();
flag[0]=true;	flag[1]=true;
}	}
Critical Section	**Critical Section**
flag[0]=false;	flag[1]=false;

[5 marks]

Marks may be awarded by considering the correctness of the solution and explanation.

Q5)(a) Explain about deadlock prevention mechanism in a system.

Solution/Evaluation Scheme:

Deadlock prevention mechanism to ensure at least any one necessary condition will not held:

Mutual Exclusion
Hold and Wait
No Preemtion
Circular Wait

[2.5 marks]

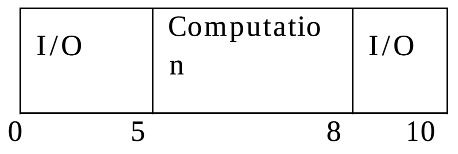
Marks may be awarded by considering the correctness of the explanation.

(b) Consider three processes (P1, and P2, and P3), all arriving at time zero, with 10, 20, and 30 time units of execution respectively. P1 spends the first 50% of execution time doing I/O, the next 30% of time doing computation, and the last 20% of time doing I/O again, but P2 and P3 spend the first 20% of execution time doing I/O, the next 70% of time doing computation, and the last 10% of time doing I/O again. The operating system uses a shortest remaining compute time first scheduling and schedules a new process either when the running process finishes its compute burst or when the running process gets blocked on I/O. Assume that all I/O operations can be overlapped. For what percentage of time does the CPU remain idle?

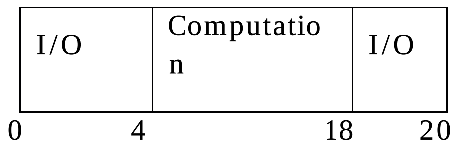
Solution/Evaluation Scheme:

The structure of the processes are as follows.

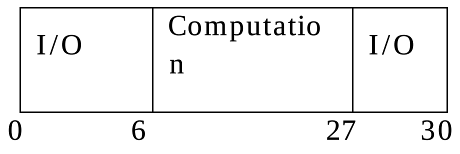
P1



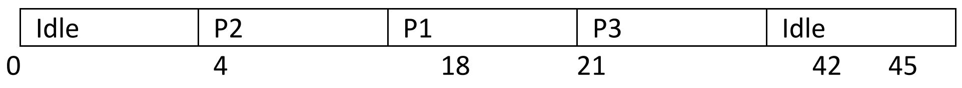
P2



P3



Execution of processes



Percentage of time CPU is idle=((4+3)/45)*100=15.5%

[2.5 marks]

Marks may be awarded by considering the correctness of the solution and explanation.