End Semester Examination (5th Sem), 2021

→ Subject Name :→ Operating Systems.

→ Subject Code :→ IT 3102

→ Date of Examination :→ 04.12.2021

→ Name :→ Aniket Majhi.

→ Examination Roll Number :→ 510819019.

→ G Suite ID :→ 510819019.aniket@students.iiests.ac.in

→ Number of sheets uploaded → 13

~~~~~~ o ~~~~~~.
~~~~~~

(1)

(a.) A process state data defines the status of the Process when it is suspended, allowing the OS to restart it later. This always includes the content of general purpose CPU registers. During context Switch, the running processes should be stopped and another process must run. The kernel must stop the execution of the running Process and assings it's resources to another process which can be get from PCB's.

(b.) The different stages of the process :→

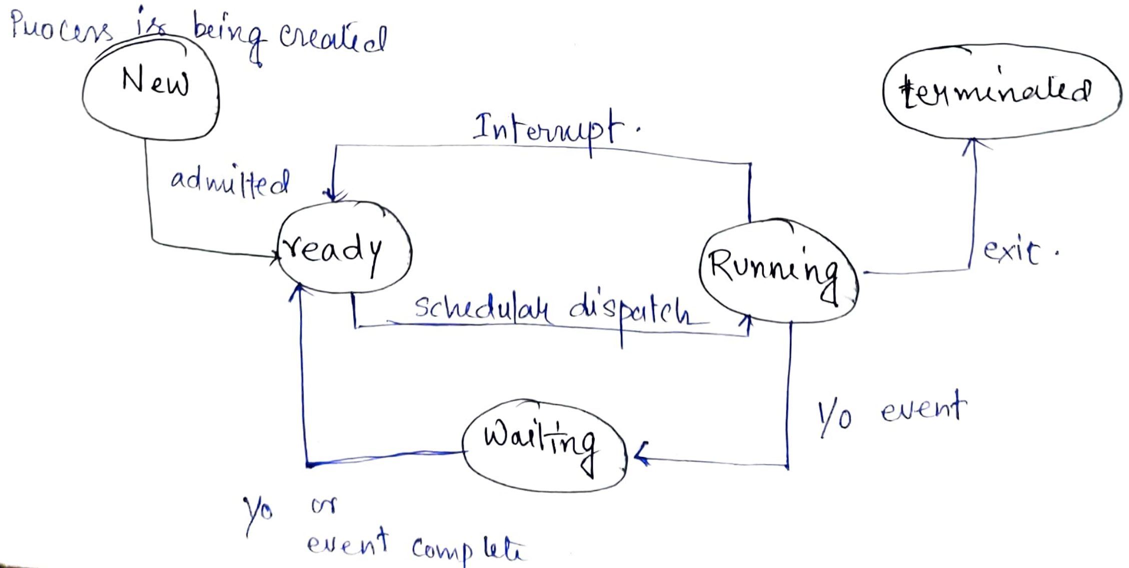
(i) new :→ In this stage the process is being created.

(ii) ready :→ In this stage the process is waiting to be assigned to a processor.

(iii) running :→ Here, the instructions are being executed.

(iv) waiting :→ Here the process is waiting for some event to occur.

(v) terminated :→ The process has finished it execution



(c.) Action taken by the Kernel to switch context between Processes :—

The actions taken by the Kernel to switch Context between processes are ———

(i) The OS must save the pc and user stack pointer of the currently executing Process, in response to a clock interrupt and transfers control to the Kernel clock interrupt handler.

(ii) save the rest of the registers, as well as other machine state such as the state of the floating pointer registers, in the procss PCB is done by the clock interrupt handler.

(iii) The schedular to determine the next Process to execute is invoked the OS;

(iv) Then the state of the next ~~point~~ process from its PCB is retrieved by OS and restores the registers. The restore operation takes the processor back to the state in which the previous process was Previously interrupted, executing in user code with user-mode privilages.

(2)

(a) Multilevel feedback queue scheduling algorithm :→

We know that there are various scheduling algorithms by which we can assign the cpu from each processes in a order. Among them multilevel feedback queue scheduling algorithm is one of them.

Here, the advantage is that of low scheduling overhead, but it is inflexible. It allows a process to move between queues. The idea is to separate process according to the characteristic of their cpu burst.

If a process uses too much of the cpu, it will be moved to the lower priority queue. This scheme leaves I/O bound and ~~it~~ interactive processes to the higher priority queues. If a process waits too much in the higher priority queue may be moved to a higher priority queue.

In general, the multilevel feedback queue schedular is defined by the following parameters ——

i) The number of queues.

ii) The scheduling algorithm for each queue.

iii) The method is used to determine when to upgrade to a higher priority queue.

(iv) This method determines when to demote a process to a lower priority queue.

(v) This method is used to determine which queue a process will enter when that process needs service.

☒ <u>Drawbacks</u> :→

The main drawbacks is it is very complex.

(b.) <u>Semaphores</u> :→ Semaphore is a technique to manage concurrent processes by using a simple integer value, ~~which is known as~~

It is simply a variable which is positive and is shared between threads. This variable is used to solve the critical-section problem to acheive process synchronization.

☒ A semaphore can be accessed by only two standart atomic operations,

wait() → P()
signal() → V().

wait()

```
P (semaphore s) {
    while (s<=0);
    s--;

}
```

signal()

```
V(semaphore s) {
    S++

}
```

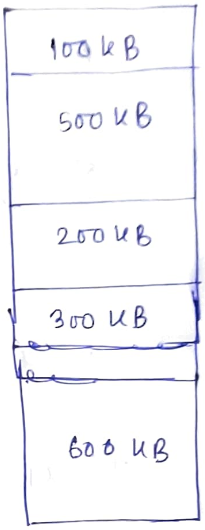
☒ There are mainly two semaphores —

(i) Binary semaphores [s = {0,1}]

(ii) Count semaphores(s = Number of instance of the resources).

(4.) The memory partitions are → 100 KB, 500 KB, 200 KB, 300 KB, and 600 KB.

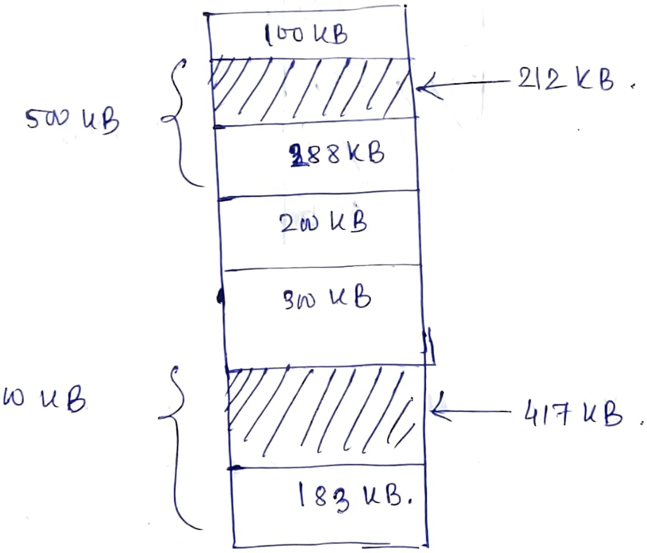
(i) For the First-Fit :→.



i) 212 KB is placed to 500 KB block as it is the first block to meet the condition.

ii) 417 KB will then be placed to 600 KB block.

Now the partition work like,



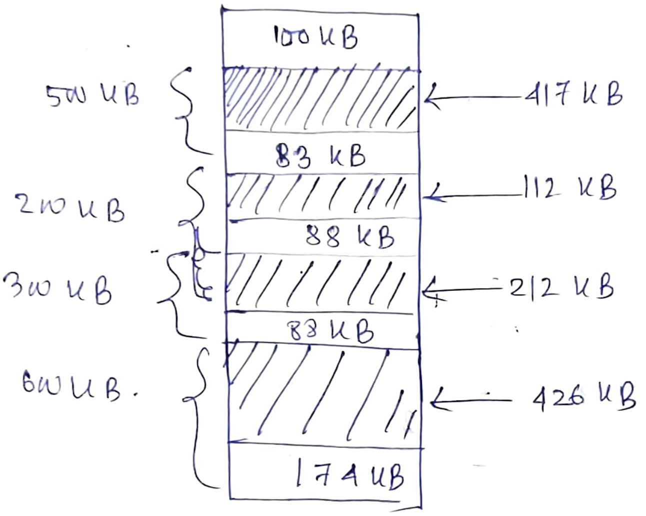
(iii) 112 KB is put the 288 KB (new block).

(iv) 426 KB must wait as there are no specific block to assign.

(ii) Best Fit :-

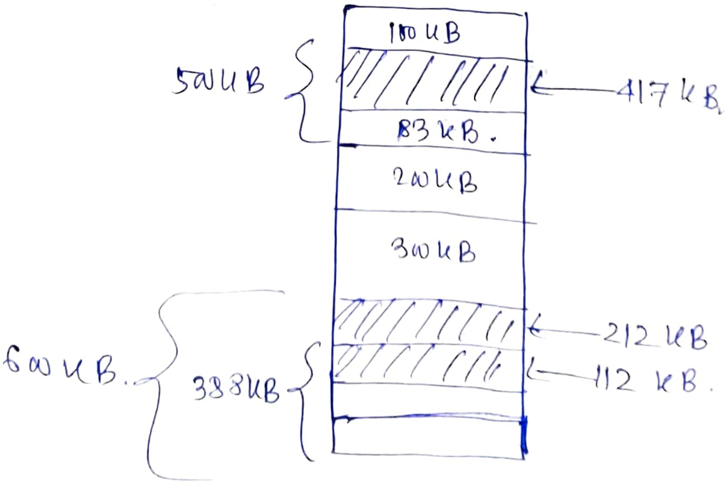
(i) 212 KB is put in 300 KB partition.
(ii) 417 KB is put in 500 KB partition.
(iii) 112 KB is put in 200KB partition.
(iv) 426 KB is put in 600KB partition.

The memory would look like,



(iii) Worst Fit :-

(i) 212 KB is put in 600 KB partition
(ii) 417 KB is put in 500 KB partition
(iii) 112 KB is put in 388 KB (600 − 212) next.
(iv) 426KB must wait.

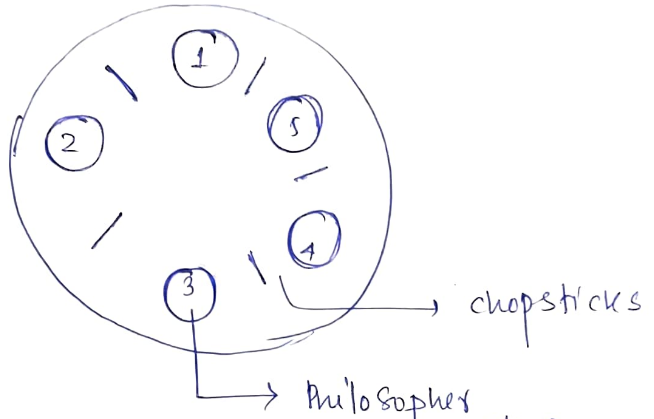


H

In this example the Best Fit Algorithms makes the most efficient use of memory.

(3.) The solution strategis of the "Dining Philosopher problem":-

Basically, ~~a~~ ~~[illegible]~~ The "Dining Philosopher problem" is one of the classic problem' in process synchronization



The problem is like, 5 Philosopher is ~~sitting~~ sited in a rounded dining table, ~~a~~ ~~[illegible]~~ a meal is served in front of each of them, and 5 chopsticks are available in total, and they are placed in between the meals as shown in the above figure. Now the problem is if a Philosopher wants to eat the meal he has to use both the chopsticks closest to him.

The main problem here is if at the begining all the Philosophers take ~~[illegible]~~ one one chopsticks then no body will be able to start for meal and every will be waiting for others to puts down' This makes the condition deadlocked.

There are few of the straligies to avoid that,

(i) Allow at most four Philosopher to be sitting together at the table.

(ii) Allow a Philosopher to pick up his chopsticks only if both chopsticks are available.

(iii) use an asymmetric solution; that is, an odd philosopher picks up first his left one and then his right one, on the other hand, even Philosopher Picks his right first then left.

(b.) Deadlock avoidance :→

There are few methods to avoid deadlocks.

(i) Here we will be discussing about banker's Algorithm ——

☒ Bankers Algo :→

~~The resources allocation~~

(i) When a new process enters into the system it must declare the max number of each resource type it may needs.

(ii) When a user requests a set of resources, the system must determine whether the allocation of this resources will leave the system is safe state/not.

(a..) If it will the resources will be allocated otherwise the process must wait until Some other process releases enough resources.

Example,

~~lets say~~

lets asay in a system there are five Processes available, and three resources are there.

~~that is~~

Lets the processes be, P1, P2, P3, P4, P5

and the resources be, A, B, C.

A has max 10 instances.

B " " 5 "

C " " 7 ".

Initialy, A = 10, B = 5, C = 7.

Process	Allocation			Max need			Available			Remaining		
	A	B	C	A	B	C	A	B	C	A	B	C
P1	0	1	0	7	5	3	3	3	2	7	4	3
P2	2	0	0	3	2	2	5	3	2	1	2	2
P3	3	0	2	9	0	2	7	4	3	6	0	0
P4	2	1	1	4	2	2	7	4	5	2	1	1
P5	0	0	2	5	3	3	7	5	5	5	3	1
Tot →	7	2	5				10	5	7			

(i) At starting available resources are,

A = 10 − 7
= 3

B = 5 - 2
= 3

C = ~~8~~ 7 − 5
= 2.

from allocation column
(means already allocated)

(i) Next, In this available resources, the process 2 (P2) will first get the resource and terminate its execution.

So, Now available resources are,

A = 5,

B = 3,

C = 2

(ii) Next, P4 will get the resources and terminate

so, A = 7, B = 4, C = 3

(iii) Next, P5 will get the resources and terminate

so, A = 7, B = 4, C = 5.

(iv) Next, P1 will get the resources and terminate,

so, A = 7, B = 5, C = 5.

(v) Next, P3 will get the resource and terminate

✓

so, A = 10, B = 5, C = 7.

The sequence to avoid deadlock is,

P2 → P4 → P5 → P1 → P3

(5)

(a.) Demand Paging :—

The process of loading the page into memory on demand (whenever page fault occurs) is known as demand paging.

The steps associated with it are :—

(1.) If the CPU tries to refer to a page that is curently not available in the main memory, it generates an interrupt indicating a memory access fault.

(2.) The OS puts the interrupted process in a blocking state. For the execution to proceed, the OS must bring the required page into the memory.

(3.) The OS will search for the required page in the logical address space.

(4.) The required page will be brought from logical address space to physical address space. The page replacement algorithms are used for the decision making of replacing the page in physical address space.

(5) The page table will be updated accordingly

(6.) The signal will be sent to the CPU to continue the program execution and it will place the process back into the ready state.

(h.)

In OS, for each process page table will be created, which will contain the page table entry (PTE). The PTE generally contains the frame number and some other bits. The PTE will tell where in the main memory the actual page is residing.

Now, the problems is where to place the Page table, such that overall access time will be less.

The idea is, Place the page table entries in registers, for each request granted from the cpu, it will be ~~mapped~~ matched to the appropriate page number in page table, which will tell where in the main memory the corresponding page resides. but the registers are in small size but pages may be large. so it is not the practical approach.

To overcome the problem, the entire page table was kept in main memory but there were some problems like we have to find out two main memory references.

one, to find the frame number,

two, to get the address specified by the page number.

To overcome the above Problem a high speed cache is set up for page table entries called, Translation lookaside buffer (TLB).