OPERATING SYSTEM

For
COMPUTER SCIENCE

OPERATING SYSTEM

SYLLABUS

Processes, threads, interprocess communication, concurrency and synchronization. Deadlock. CPU scheduling. Memory management and virtual memory. File systems.

ANALYSIS OF GATE PAPERS

Exam Year	1 Mark Ques.	2 Mark Ques.	Total
2003	2	5	12
2004	3	4	11
2005	-	2	4
2006	1	8	17
2007	2	6	14
2008	2	5	12
2009	2	5	12
2010	3	2	7
2011	4	2	8
2012	1	4	9
2013	2	4	10
2014 Set-1	2	3	8
2014 Set-2	1	3	7
2014 Set-3	1	3	7
2015 Set-1	2	4	10
2015 Set-2	2	3	8
2015 Set-3	2	2	6
2016 Set-1	1	4	9
2016 Set-2	1	3	7
2017 Set-1	2	2	6
2017 Set-2	2	2	6
2018	4	6	10

CONTENTS

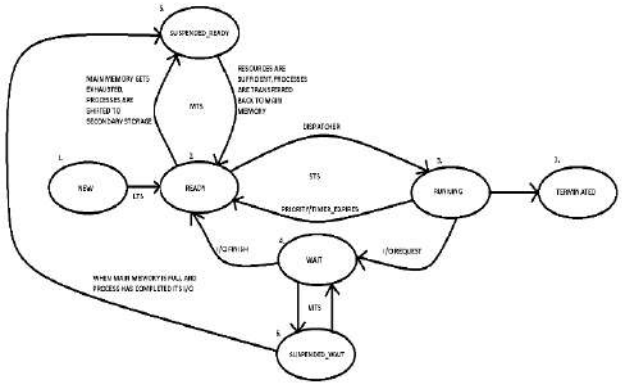
1 PROCESS & DEADLOCK

1.1 PROCESS

Process is just an executing program, including the current values of the program counter, registers, and variables.

1.2 PROCESS STATES

Figure shows all process states.



State 2, 3, and 4: Process in these states reside in RAM.
State 5, 6: Process is shifted to secondary storage deliberately for some time.

1.2.1 Three types of scheduler:

(a) LTS (Long Term Scheduler): It is the responsibility of LTS to create a process and bring it in ready state. LTS decides which programs are admitted to system for execution.

(b) MTS (Medium Term Scheduler): Its responsibility is in two cases. When ready stated becomes crowded so that OS doesn't have resources (RAM) to accommodate large number of processes in ready state, they can be moved by MTS to suspended_ready state.
Also, when resources (RAM) gets exhausted in wait state due to crowding, MTS shifts them to suspended wait state (i.e. secondary storage).MTS reduces degree of multiprogramming.

(c) STS (Short Term Scheduler): It is responsible to pick one process from ready state and allocate it to CPU.STS saves and loads context of a process in its PCB while switching.

LTS: Job Scheduler
MTS: Swapping
STS: CPU Scheduler/Dispatcher
STS is faster than LTS.

Process with respect to their execution time are divided into two types:

(a) CPU Bound Process: - Processes which require more CPU time are called CPU Bound Process. The process will spend more time in ready state.

(b) IO Bound Process: - Processes which require more IO time are called IO Bound Process.

1.3 Degree of Multiprogramming

The number of processes in the memory at any point of time is called degree of multiprogramming. LTS should select good combination of CPU bound and IO bound processes. LTS controls degree of multiprogramming.

1.4 PROCESS CONTROL BLOCK

Every process has its own PCB. PCB is stored in RAM.
Attributes: pid, process state, PC (program counter),GPR(general purpose registers),Priority, List of open files, Protection & security.

The set of these attributes is called context. Context is stored in PCB.

1.5 THREADS

Threads are light weight processes. Each thread has its own Stack, Registers and Program Counter. Thread share data, code, memory, files, address space, File descriptor table, so on.

User Level Threads	Kernel Level Threads
User level threads are dependent.	Kernel level threads are independent.
They are faster.	They are comparatively slower but faster than a process.
They are scheduled by thread library.	They are scheduled by OS.
They have less context.	They have more context.
Blocking IO calls: If one thread other thread of that process can run.	Non-Blocking IO calls: If one thread blocks for IO, OS schedules another for execution.

1.6 CPU SCHEDULING

SCHEDULING CRITERIA

The scheduling criteria that are used includes

i) CPU utilization

The CPU should be kept as busy as possible. CPU utilization may range from 0 to 100 per cent. In a real system, it should range from 40 per cent (for a lightly loaded system) to 90 per cent (for a heavily used system).

ii) Throughput

If the CPU is busy executing processes, then work is being done. One measure of work is the number of processes that are completed per time unit, called throughput. For long processes, this rate may be one.

Various Time Measures related to CPU Scheduling:

(a) Arrival Time

The time when the process arrived into ready state is called arrival time.

(b) Burst Time

The time required for the process to complete its execution is called Burst Time.

(c) Completion Time

The time when the process has completed its execution is called completion time.

(d) Turnaround time

The interval from the time of submission of a process to the time of completion is the turnaround time. TAT = CT – WT.

(e) Waiting time

The amount of time a process spends waiting in the ready queue.
WT = TAT-BT.

(f) Response time

Response time, is the amount of time it takes to start responding, but not the time that it takes to output that response. The turnaround time is generally limited by the speed of the output device.

The goal is to maximize CPU Utilization, minimize average Turn

around Time, and minimize Waiting Time of processes.

1.7 SCHEDULING ALGORITHMS

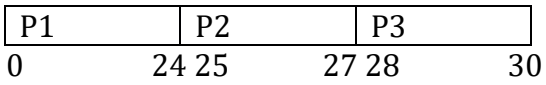
1.7.1 First-Come, First-Served Scheduling (FCFS)

Criteria: Arrival Time
Mode : Non Preemptive
Data Structure: Queue
Tie Breaker : Process Number

The average waiting time under the FCFS policy, however, is often quite long.
Consider the following set of processes that arrive at time 0, with the length of the CPU- burst time given in milliseconds:

PROCESS	ARRIVAL TIME	BURST TIME	CT	TAT	WT
P_1	0	24	24	24	0
P_2	0	3	27	27	24
P_3	0	3	30	30	27

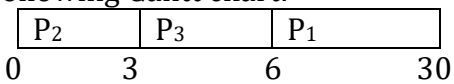
GANTT CHART:



Now, the waiting time is 0 milliseconds for process P_1, 24 milliseconds for process P_2, and 27 milliseconds for process P_3. Thus, the average waiting time = $\frac{0+24+27}{3}$ = 17milliseconds.

This example also shows Convoy effect. FCFS suffers from convoy effect.

If the processes arrive in the order P_2, P_3, P_1, however, the results will be as shown in the following Gantt chart:



Now, the average waiting time = $\frac{6+0+3}{3}$ = 3milliseconds.

This reduction is substantial. Thus, the average waiting time under a FCFS policy is generally not minimal.
The FCFS scheduling algorithm is non-preemptive. Once, the CPU has been allocated to a process, that process keeps the CPU until it releases the CPU, either by terminating or by requesting I/O.

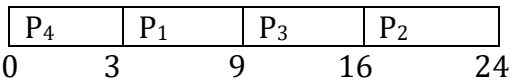
1.7.2 Shortest –Job-First Scheduling (SJF)

Criteria: Burst Time
Mode : Non Preemptive
Data Structure: Min Heap
Tie Breaker : Arrival Time
Shortest Job First has minimum average waiting time among all CPU scheduling algorithms.

Consider the following set of process, with the length of the CPU burst time given in milliseconds:

PROCESS	BURST TIME
P_1	6
P_2	8
P_3	7
P_4	3

Using SJF scheduling, we would schedule these processes according the following Gantt chart:



The waiting time is 3 milliseconds for process P_1, 16 milliseconds for process P_2, 9 milliseconds for process P_3, and 0 milliseconds for process P_4.
Thus, the average waiting time = $\frac{3+16+9+0}{4}$ = 7 milliseconds.
If we were using the FCFS scheduling scheme, then the average waiting time would be 10.25 milliseconds.
The SJF scheduling algorithm is provably optimal, in that it gives the minimum average waiting time for a given set of

processes. If two processes have the same length next CPU burst, FCFS scheduling is used to break the tie.

SJF ADVANTAGES

(a) Maximum Throughput.
(b) Minimum Average WT and Minimum Average TAT.

SJF DISADVANTAGES

(a) Starvation to longer jobs.
(b) It is not implementable because BT of process cannot be known ahead.

Solution:

SJF with predictive Burst Time.

Prediction Techniques:

I. **Static :**
   a. Process Size
   b. Process Type (OS ,Interactive, Foreground, Background)

II. **Dynamic:**
   a. Simple Averaging
   b. Exponential Averaging

a. **Simple Averaging:**
Number of process: n (P1, P2, P3, ... , Pn)
t_i: Actual BT ,where i= 1 to n
τ_i : Predicted BT
$\tau_{n+1} = (1/n) * \sum t_i$, where i=1 to n.

b. **Exponential Averaging:**
Let t_n be the length of the nth CPU burst, and let τ_{n+1}, be our predicted value for the next CPU burst. Then, for $0 \le \alpha \le 1$, define $\tau_{n+1} = \alpha t_n + (1-\alpha)\tau_n$

This formula defines an exponential average. The value of t_n contains our most recent information; τ_n stores the past history. The parameter a controls the relative weight of recent and past history in our production.

If $\alpha=0$, then $\tau_{n+1} = t_n$, and recent history has no effect (current conditions are assumed to be transient);

If $\alpha=1$, then $\tau_{n+1} = t_n$ and only the most recent CPU burst matters (history is assumed to be old and irrelevant). More commonly, $\alpha = 1/2$.

To understand the behavior of the exponential average, we can expand the formula for τ_{n+1} by substituting for τ_n, to find

$$\tau_{n+1} = \alpha t_n + (1-\alpha)\alpha\tau_{n-1} + \ldots + (1-\alpha)^j \alpha t_{n-j} + (1+\alpha)^{n+1}\tau_0$$

Since both α and (1 - α) are less than or equal to 1, each successive term has less weight than its predecessor. The SJF algorithm may be either preemptive or non-pre-emptive. The choice arises when a new process arrives at the ready queue while a previous process is of the currently executing process. A preemptive SJF algorithm will preempt the currently executing process; whereas a non-pre-emptive SJF algorithm will allow the currently running process to finish its CPU burst Preemptive SJF scheduling is sometimes called shortest-remaining-lime-first scheduling.

Consider the following four processes, with the length of the CPU-burst time given in milliseconds:

PROCESS	ARRIVAL TIME	BURST TIME	CT	TAT	WT
P_1	0	8	17	17	9
P_2	1	4	5	4	0
P_3	2	9	26	24	15
P_4	3	5	10	7	2

Preemptive SJF schedule is as depicted in the following Gantt chart:

P1	P2	P4	P1	P3
0	1	5	10	17	26

Process P_1 is started at time 0, since it is the only process in the queue. Process P_2 arrives at time 1. The remaining time for process P_1 (7 milliseconds) is larger than the time required by process P_2 (4 milliseconds), so process P_1 is preempted, and process P_2 is scheduled.
The average waiting time
= 9+0+15+2/4
=26/4

=6.5milliseconds.
A non-pre-emptive SJF scheduling would result in an average waiting time of 7.75 milliseconds.

1.7.3 Shortest Remaining Time First Scheduling (SRTF)
Criteria: Burst Time
Mode : Preemptive
Data Structure: Min Heap
Tie Breaker : LEVEL 0: Arrival Time
LEVEL 1: Process No.
A preemptive version of shortest job first is **shortest remaining time next**. With this algorithm, the scheduler always chooses the process whose remaining run time is the shortest. Again here, the run time has to be known in advance. When a new job arrives, its total time is compared to the current process' remaining time. If the new job needs less time to finish than the current process, the current process is suspended and the new job started. This scheme allows new short jobs to get good service.

1.7.4 Round Robin Scheduling
Criteria: Time Quantum and Arrival Time
(Practically implementable since it doesn't depend on Burst Time).
Mode : Preemptive
Data Structure: Queue
Tie Breaker : Process Number
No Starvation.
Minimum Response Time.

What would happen if Time Quantum is very small and TQ is very large?
Small TQ: More Context Switching. Therefore, reduced throughput.
Large TQ: Starvation for some process.

Example:

1. **TQ=4 units**

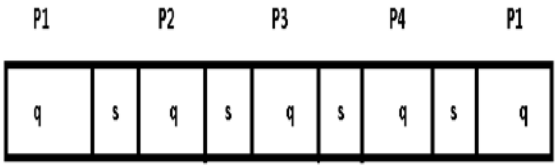
Pno	AT	BT	CT	TAT	WT
1	0	4	4	4	0
2	1	5	19	18	13
3	2	2	10	8	6
4	3	1	11	8	7
5	4	6	21	17	13
6	5	3	18	12	9

P1	P2	P3	P4	P5	P6	P2	P5
0	4	8	10	11	15	18	19	21

As you increase TQ, Context Switching decreases, Response Time increases.
As you decrease TQ, Context Switching increases, Response Time decreases.
If TQ = infinity, Round Robin reduces to FCFS.

Example 2:
Let Time Quantum = 'q' units.
Context Switching Time ='s' units.
Each process is guaranteed to get turn at CPU for't' seconds.
Number of processes = 'n'
What must be the value of 'q' so that 's' is reduced?

Solution:



n * s + (n-1)*q <= t
q <= (t-ns)/ (n-1)

For Process P1:

4 * S + (4-1)*Q <= T

1.7.5 Highest Response Ratio Next
Criteria: Response Ratio
Response Ratio = (w+s)/s
W: Waiting Time for a process so far.
S: Service Time of a process or Burst Time
Mode : Non- Preemptive

Analysis:

(a) Process with lowest BT will have highest RR.
(b) Also, the process with longest waiting time will have higher RR.

So, HRRN limits the waiting time of longer jobs and favors shorter jobs simultaneously.
HRRN overcomes the problem of starvation of SJF where we take only shortest jobs first.
HRRN is better than SJF but both fails at implementation level because Burst Time is one of their criteria which can't be known priory.

Pno	AT	BT	CT	TAT	WT	RT
0	**0**	**3**	**3**	**3**	**0**	**0-0=0**
1	**2**	**6**	**9**	**7**	**1**	**3-2=1**
2	**4**	**4**	**13**	**11**	**7**	**9-4=5**
3	**6**	**5**	**20**	**14**	**9**	**15-6=9**
4	**8**	**2**	**15**	**7**	**5**	**13-8=5**

At t=9 P3, P3 and P4 are available
RR2 = ((9-4) +4)/4 = 2.25 □ Maximum (Scheduled Next)
RR3 = ((9-6) +5)/5 = 8/5
RR4 = ((9-8) +2)/2 = 3/2=1.5

At t=13 P3 and P4 are available
RR3 = ((13-6) +5)/5 = 12/5
RR4 = ((13-8) +2)/2 = 7/2=3.5 □ Maximum (Scheduled Next)

Gantt chart:

P0	P1	P2	P4	P3	
1	3	9	13	15	20

1.7.6 Non Pre-emptive Priority Scheduling
Criteria: Priority
Mode : Non- Preemptive
Tie Breaker: Level – 0 Arrival Time
Level—1 Process Number.
First prefer lower arrival time, then prefer lower process number.

Pno	PRIORITY	AT	BT	CT	TAT	WT	RT
1	2	0	4	4	4	0	0
2	4	1	2	25	24	22	22
3	6	2	3	23	21	18	18
4	10	3	5	9	6	1	1
5	8	4	1	20	16	15	15
6	12	5	4	13	8	4	4
7	9	6	6	19	13	7	7

Gantt chart:

P1	P4	P6	P7	P5	P3	P2
0	4	9	13	19	20	23	25

1.7.7 Pre-emptive Priority Scheduling

Criteria: Priority

Mode : Preemptive

Tie Breaker: Level – 0 Arrival Time

Level—1 Process Number.

First prefer lower arrival time, then prefer lower process number.

Pno	PRIORITY	AT	BT	CT	TAT	WT	RT
1	2	0	4	25	25	21	0
2	4	1	2	22	21	19	0
3	6	2	3	21	19	16	0
4	10	3	5	12	9	4	0
5	8	4	1	19	15	14	14
6	12	5	4	9	4	0	0
7	9	6	6	18	12	6	6

Gantt chart:

P1	P2	P3	P4	P4	P6	P4	P7	P5	P3	P2	P1

0 1 2 3 4 5 9 12 18 19 21 22 25

After all processes have arrived, preemptive priority scheduling becomes non-preemptive priority scheduling.

If Arrival Time is same for all processes, Preemptive = Non-Preemptive. This is true for SJF also.

1.7.8 Multilevel Queue Scheduling

Another class of scheduling algorithms has been created for situations in which processes are easily classified into different groups. For example, a common division is made between foreground (interactive) processes and background (batch) processes. These two types of processes have different response-time requirements, and so might have different scheduling needs. In addition, foreground, foreground processes may have priority (externally defined) over back ground processes.

A multilevel queue-scheduling algorithm partitions the ready queue into several separate queues (Figure below).

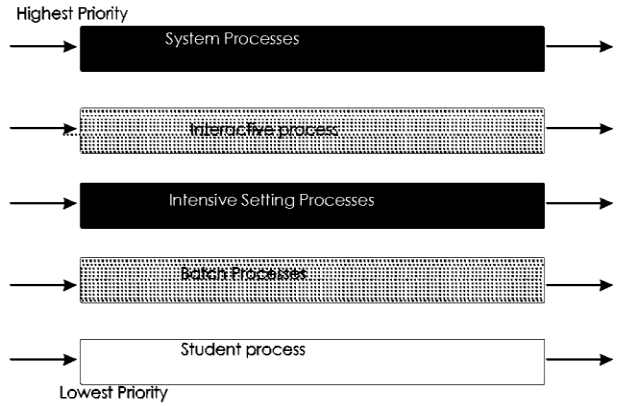


Fig. Multilevel queue scheduling

Advantages: Different Scheduling algorithms can be applied for different queues.

Disadvantages: As long as there is a single process in highest queue, no process from lower queues will get a chance.

1.7.9 Multilevel Feedback Queue Scheduling

Multilevel feedback queue scheduling, however, allows a process to move between queues. The idea is to separate processes with different CPU-burst characteristics. If a Process uses too much CPU time, it will be moved to a lower-priority queue. This scheme leaves I/O-bound and interactive processes in the higher-priority queues. Similarly, a process that waits too long in a lower-priority queue may be moved to a higher-priority queue. This form of aging prevents starvation.

In general, a multilevel feedback queue scheduler is defined by the following parameters:

a) The number of queues
b) The scheduling algorithm for each queue
c) The method used to determine when to demote a process to a lower-priority queue
d) The method used to determine which queue a process will enter when that process needs service. The definition of a multilevel feedback queue scheduler makes it the most general CPU scheduling algorithm.

Advantage: No starvation at all.

SOME POINTS TO PONDER:

(a) All process are IO Bound: CPU Utilization decreases, Throughput decreases.

(b) All process are CPU Bound: CPU Utilization increases. Throughput decreases. Here, throughput decreases because number of process executed per second decreases.

Throughput = Number of process completed (n)/Time to complete (t)

STARVATION: (IN ALL ALGORITHMS)

FCFS	NP-SJF	SRTF	RR	PRIORITY-NP	PRIORITY-P	LJF-NP	LRTF	HRRN	MULTILEVEL QUEUE	MULTILEVEL FEEDBACK QUEUE
NO	YES	YES	NO	YES	YES	YES	YES	NO	YES	NO

1.8 DEADLOCK

1.8.1 Introduction

Under the normal mode of operation, a process may utilize a resource in only the following sequence:

a) Request:
If the request cannot be granted immediately (for example, the resource is being used by another process), then the requesting process must wait until it can acquire the resource.

b) Use:
The process can operate on the resource (for example, if the resource is a printer, the process can print on the printer).

c) Release:
The process releases the resource. The request and release of resources are system calls. Examples are the **request and release device, open** and **close file,** and **allocate** and **free memory** system calls. Request and release of other resources can be accomplished through the wait and signal operations on semaphores. Therefore, for each use, the operating system checks to make sure that the using process has requested and been allocated the resource. A system table records whether each resource is free or allocated, and, if a resource is allocated, to which process. If a process requests a resource that is currently allocated to another process, it can be added to a queue of processes waiting for this resource. To illustrate a deadlock state, consider a system with three tape

drives. Suppose that there are three processes, each holding one of these tape drives. If each process now requests another tape drive, the three processes will be in a deadlock state. Each is waiting for the event "tape drive is released," which can be caused only by one of the other waiting processes. This example illustrates a deadlock involving processes competing for the same resource type. Deadlocks may also involve different resource types. Consider a system with one printer and one tape drive. Suppose that process P_i is holding the tape drive and process P_j is holding the printer. If P_i requests the printer and P_j requests the tape drive, a deadlock occurs.

1.8.2 DEADLOCK CHARACTERIZATION

Necessary Conditions

A deadlock situation can arise if the following four conditions hold simultaneously in a system:

a) Mutual exclusion

At least one resource must be held in a non-sharable mode; i.e., only one process at a time can use the resource. If another process requests that resource, the requesting process must be delayed until the resource has been released.

b) Hold and wait

There must exist a process that is holding at least one resource and is waiting to acquire additional resources that are currently being held by other processes.

c) No preemption

Resources cannot be preempted; i.e., a resource can be released only voluntarily by the process holding it, after that process has completed its task.

4) Circular waits

There must exist a set $\{P_0, P_1, ...P_n\}$ of waiting such that P_0 is waiting for a resource that is held by P_1, P_1 is waiting for a resource that is held by P_2, ……,P_{n-1} is waiting for a resource that is held by P_n, and P_n is waiting for a resource that is held by P_0.

All these four conditions must hold for deadlock to occur.

1.8.3 Methods for Handling Deadlocks

There are three different methods for dealing with the deadlock problem.

a) We can use a protocol to ensure that the system will never enter a deadlock state.

b) We can allow the system to enter a deadlock state and then recover.

c) We can ignore the problem all together, and pretend that deadlocks never occur in the system. This solution is the one used by most operating systems, including UNIX.

To ensure that deadlocks never occur, the system can use either deadlock prevention or a deadlock-avoidance scheme.

1.8.4 Deadlock Prevention

For a deadlock to occur, each of the four necessary conditions must hold. By ensuring that at least of these conditions cannot hold, we can prevent the occurrence of a deadlock. Now consider each of the four necessary conditions separately.

1.8.4.1 Mutual Exclusion

The mutual –exclusion must hold for non-sharable resources. For example, a printer cannot be simultaneously shared by several processes. Sharable resources, on the other hand, do not require mutually

exclusive access, and thus cannot be involved in a deadlock. Read-only files are a good example of a sharable resource. If several Processes attempt to open a read-only file at the same time, they can be granted simultaneous access to the file.

1.8.4.2 Hold and Wait

To ensure that the hold-and –wait condition occurs in the system, we must guarantee that, whether a process requests a resource, it does not hold any other resources. One protocol that can be used requires each process to request and be allocated all its resources before it being execution. We can implement this provision by requiring that system calls requesting resources for a process precede all other system calls.

1.8.4.3 No Preemption

To ensure that this condition does hold, we can use following protocol. If a process that is hold some resources requests another resource that cannot be immediately allocated to it (that is, the process must wait), then all resources currently being held are preempted. That is, these resources are implicitly released. The preempted resources are added to the list of resources for which the process is waiting. The process will be restarted only when it can regain its old resources, as well as the new ones that it is requesting.

An alternative protocol allows a process to request resources only when the process has none. A process may request some resources and use them. Before it can request any additional consider a system with five processes P_0 through P_4 and three resources that it is currently allocated.

Consider a system with five processes P0 through P4 and three resource type A,B,C. Resource type A 7 instance, resources. Suppose that, at time T0, we have the following resource allocation stare:

	Allocation			Request			Available		
	A	B	C	A	B	C	A	B	C
P_0	0	1	0	0		0	0	0	0
P_1	2	0	0	2		0			
P_2	3	0	3	0		0			
P_3	2	1	1	1		0			
P_4	0	0	2	0		0			

We claim that the system is not in a deadlocked state. Indeed, if we execute our algorithm, we will find that the sequence < P_0, P_2 P_3, P_1 P_4 > will result in Finish[i] = true for all i. Now suppose that process P_2 makes one additional request for an instance of type C. The Request matrix is modified as follows:

	Request		
P0	0	0	2
P1	2	0	2
P2	0	0	1
P3	1	0	0
P4	0	0	2

We can say that the system is now deadlocked. Although we can reclaim the resources held by process P_0 the number of available resources is not sufficient to fulfill the requests of the other processes. Thus, a deadlock exists, consisting of processes P_1, P_2, P_3, and P_4.

1.8.4.4 Detection-Algorithm Usage

When should we invoke the detection algorithm? The answer depends on two factors:

i) How often is a deadlock likely to occur?
ii) How many processes will be affected by deadlock when it happens?

1.8.5 Recovery from Deadlock

1.8.5.1 Process Termination

To eliminate deadlocks by aborting a process, we use one of two methods. In both methods, the system reclaims all resources allocated to the terminated processes.

i) Abort all deadlocked processes
This method clearly will break the deadlock cycle, but at a great expense, since these processes may have computed for a long time, and the results of these partial computations must be discarded, and probably must be recomputed later.

ii) Abort one process at a time until the deadlock cycle is eliminated
This method incurs considerable overhead, since, after each process is aborted, a deadlock-detection algorithm must be invoked to determine whether any processes are still deadlocked.

1.8.5.2 Resource Preemptions

To eliminate deadlock using resource preemption, we successively preempt some resources and give these resources to other processes until the deadlock cycle is broken. If preemption is required to deal with deadlocks, then there issues need to be addressed:

a) Selecting a victim
Which resources and which processes are to be preempted? As is process termination, we must determine the order of preemption to minimize cost. Cost factors may include such parameters as the number of resources a deadlock process is holding, and the amount of time a deadlock process has thus far consumed during its execution.

b) Rollback
If we preempt a resource from a process, what should be done with that process? Clearly, it cannot continue with its normal execution; it is missing some needed resource. We must roll back the process to some safe state, and restart it from that state.
Since, in general, it is difficult to determine what a safe state is, the simplest solution is a total rollback: Abort the process and then restart it. However, it is more effective to roll back the process only as far as necessary to break the deadlock.

c) Starvation
How do we ensure that starvation will not occur? That is, how can we guarantee that resources will not always be preempted from the same process? In a system where victim selection is based primarily on cost factors, it may happen that its designated task, a starvation situation that needs to be dealt with in any practical system. Clearly, we must ensure that a process can be picked as a victim only a (small) finite number of times. The most common solution is to include the number of rollbacks in the cost factor.

1.9 THERE ARE THREE TYPES OF PROBLEMS WITH IPC:

Inconsistency.
Loss of data.
Deadlock.

1.10 PRODUCER CONSUMER PROBLEM: Process is just an executing program, including the current values of the program counter, registers, and variables.

PRODUCER (N = Fixed Buffer Size) CONSUMER

Produces item.
IN variable.

(IN = (IN+1) %N)
(OUT = (OUT+1) %N)

COUNT variable is shared between
Both Producer & Consumer.

BUFFER is shared resource between
Both Producer & Consumer.

Consumes item.
OUT variable.

Increments COUNT.

Decrements COUNT.

STEPS:
Produce Item.
Check condition to
Wait (sleep) or polling.
BUFFER [IN] = Item.
Increment IN.
Increment COUNT.

STEPS:
1. Check condition to
wait (sleep) or polling.
2. Increment OUT.
3. Decrement COUNT.
4. Consume Item.

```
Void producer(){

        While(true){

                Produce_item(temp);Step 1
        While(COUNT==N);Step 2
        BUFFER[IN] = Item;Step 3
        IN = (IN+1) %N;        Step 4
        COUNT = COUNT + 1; Step 5
        }

}
```

```
Void consumer(){
Int itemc;
While(true){

  While(COUNT==0);Step 1
        Itemc = BUFFER[OUT];
OUT = (OUT+1) %N; Step 2
  COUNT = COUNT – 1;        Step 3
Consume_item(itemc);        Step 4
}

}
```

Problems:

Step 5 or producer () and Step 3 of consumer() are points of focus here.
Suppose COUNT=4.
Producer puts an item on buffer, but before incrementing COUNT, control goes to consumer.
Consumer consumes one item, but before decrementing COUNT, control goes to producer.
Now, one time increment and one time decrement is to be executed.
COUNT++ : 4+1 =5
COUNT--:4-1=3
COUNT-- : 5-1 =4
COUNT++: 3+1=4
But increment & decrements are not executed in one machine instruction, they require three machine instructions which are interleaved.

COUNT++
R1 = COUNT
R1 = R1 +1
COUNT = R1
COUNT—
R2=COUNT
R2 = R2 – 1
COUNT = R2
This injects inconsistency in the variable states.

Producer Consumer suffers with Inconsistency. This is not compulsory that it must suffer from all problems. If anyone is present, it is an IPC problem

1.11 PRINTER AND SPOOLER PROBLEM

Shared Resource are IN variable and Spooler Directory.
IN: It is used by processes.

Algorithmic steps to enter a file in Spooler Directory:

Load Ri, M [IN] //Read IN.
Store SD [Ri],"Filename" //Make an entry.
Increment Ri.
Store M[IN],Ri //Update IN
Printer & Spooler suffers from Loss of data problem.

Deadlock: Possibility for processes entering deadlock is an IPC problem.
Race Condition: When the final value of a shared variable depends on the order of execution of two processes, it is said to be a race condition.
Example: COUNT variable in Producer Consumer IN variable in Printer & Spooler

There are three conditions to achieve Synchronization:

- Mutual Exclusion: No two processes are present in the critical section at the same time.
- Progress: No process running outside Critical Section should restrict another process to go inside, if Critical section is free.
- Bounded/Busy waiting: No process has to wait forever to enter critical section. If this condition isn't satisfied, there is a possibility of starvation.

1.12 SOLUTIONS TO IPC/SYNCHRONIZATION

Some solutions comply with all three conditions, some fail at one or two conditions.

1.12.1 Software Solutions:

1.12.1.1 Lock Variable

Mutual Exclusion fails.
Progress is achieved.
Bounded/Busy wait fails here.
Entry Section:
Load Ri,M[lock]

```
    Cmp Ri,#0
    Jnz Step 1
    Store M[lock],#1
    Critical Section
    Store M[lock],#0
```

Lock: 0 Critical Section is free.
Lock: 1 Critical Section is busy.

1.12.1.2 Strict Alteration & Decker's Algorithm

Only alternate sequence is allowed.
This approach is restricted to two processes only.

```
P0                          P1
While (true) {              While (true) {
Non_CS ()                   Non_CS ()
While (turn! = 0);          While (turn! = 1);
CS ();                      CS ();
Turn=1;                     turn=0;
}                           }
```

Mutual Exclusion is satisfied.
Progress is not achieved.
Bounded/Busy wait is satisfied.

1.12.1.3 Peterson's Solution

```
#define N 2
#define TRUE 1
#define FALSE 0
Int turn;
        //Shared Variable
Int interested [N];
        //Shared Variable

Void enter_region(int process){
        Int other = 1- process;
        Interested[process] = TRUE;
        Turn = process;
        While(turn ==process  &&
interested[other]==TRUE);
}

Void leave_region(int process){
        Interested [process] = FALSE;
}
```

Mutual Exclusion is satisfied.
Progress is achieved.
Bounded/Busy wait is satisfied.

1.12.1.4 Semaphores

Semaphore is a system variable used to achieve mutual exclusion.
The following operations can be performed on semaphores:

Down() or P() or Wait()
Up() or V() or Release() or Signal()

Semaphores are of two types:

Counting Semaphore:

Down () operations is performed as follows:

```
Value_of_semaphore--;
If(value_of_semaphore<0)
```

Block process,put it in suspended list.

Up () operations is performed as follows:

```
Value = Value +1
If (value <= 0 )
```

Select a process from suspended list & wake up.

Down () is successful if process is not blocked, else unsuccessful.
Up () is always successful.

II. Binary Semaphore:

```
Down()
{
        If(value == 1)
                S=0
        Else if(value == 0)
                Block process, put
                it in suspended list.
}
        Up()
{
```

If list is non empty, select a process & wake up.
Else if S==0 or S==1, make it 1.
}
Down() is successful when S=1,unsuccessful when S=0.
Up() is always successful.

1.13 Solution to PRODUCER CONSUMER PROBLEM using SEMAPHORES

Producer

```
While(true){
        Produce(itemp);
        Down(empty);
        Down(mutex);

        BUFFER[IN] = itemp;
        IN = (IN+1) %N;
        Up(mutex);
        Up(Full);
}
```

Consumer

```
While(true){

        Down(full);
        Down(mutex);

        Itemc = BUFFER[OUT];
        OUT = (OUT+1) %N;
        Up(mutex);
        Up(empty);
}
```

Note:

- Empty + Full = N (Always)
- Mutex: Binary Semaphore to access buffer in mutually exclusive manner.
- Empty: Counting Semaphore
- Full : Counting Semaphore
- COUNT in previous approach is replaced here by Mutex, Empty, Full.

1.14 Solution to READER & WRITER PROBLEM

- Multiple readers are allowed to read database.
- Writer needs exclusive access i.e. no reader, no writer.
- For writer who is waiting outside, we have to maintain rc(reader count). As one reader does rc++,other reader reads old value, so inconsistency may occur. Therefore, rc requires a semaphore (mutex).
- Semaphore db: For access to database.

```
Mutex = 1
Db =1
Rc = 0

Void reader(){

        Down(mutex)
        //increment rc & down db
        Rc = rc+1
        // if 1st reader
        If(rc==1)
                Down(db);
        Up(mutex);
        Read();
        Down(mutex);
        Rc =  rc -1;
        If(rc==0)
                Up(db);
        Up(mutex);
}

Void writer(){
        //decrement rc & up(db)
        While(TRUE){
        //if last reader
                Down(db)
```

```
            Write();
            Up(db);
        }
}
```

The above solution is correct but,

i. Let 10 readers are in reading.
ii. Writer comes and waits.
iii. Until all 10 readers goes out & writer goes in and comes out, no other reader is allowed.

If new reader comes every 2 seconds and every reader takes 5 seconds, then writer will suffer from starvation.

GATE QUESTIONS

Q.1 A processor needs software interrupt to
a) Test the interrupt system of the processor
b) Implement co- routines
c) Obtain system series which need execution of privileged instructions
d) Return from subroutine

[GATE -2001]

Q.2 A CPU has two modes- privileged and non- privileged. In order to change the mode from privileged to non- privileged
a) a hardware interrupt is needed
b) a software interrupt is needed
c) a privileged instruction (which does not generate an interrupt) is needed.
d) a non-privileged instruction (which does not generate an interrupt) is needed.

[GATE -2001]

Q.3 Consider a set of n tasks with known runtimes $r_1, r_2, \ldots\ldots, r_n$ to be run on a uni-processor machine. Which of the following processor scheduling algorithms will result in the maximum throughput?
a) Round Robin
b) Shortest Job First
c) Highest Response Ratio Next
d) First Come First Served

[GATE-2001]

Q.4 Which of the following scheduling algorithms is non-pre-emptive?
a) Round Robin
b) First-in-First out
c) Multilevel queue Scheduling
d) Multilevel Queue Scheduling with Feedback

[GATE-2001]

Q.5 Where does the swap space reside?
a) RAM b) Disk
c) ROM d) On-chip cache

[GATE-2001]

Q.6 Consider Peterson's algorithm for mutual exclusion between two concurrent processes I and j. The program executed by process is shown below.
repeat flag [i] = true;
turn = j ;
while (P) do no-op;
Enter critical section, perform actions, then exit critical section
Flag [i] = false;
Perform other non-critical section actions. Until false;
For the program to guarantee mutual exclusion, the predicate P in the while loop should be
a) flag [j]= true and turn = i
b) flag [j] = true and turn = j
c) flag [i]= true and turn = j
d) flag [i] = true and turn = i

[GATE-2001]

Q.7 Which combination of the following features will suffice to characterize an operating system as a multi-programmed operating system?
A) More than one program may be loaded into main memory at the same time for execution.

B) If a program waits for certain events such as IN/OUT, another program is immediately scheduled for execution.
C) If the execution of a program terminates, another program is immediately scheduled for execution.

a) (A) only b) (A) and (B)
c)(A) and (C) d) (A)(B)and(C)

[GATE-2002]

Q.8 Draw the process state transition diagram of an OS in which (i) each process is in one of the five states: created, ready, running, blocked (i.e. sleep or wait). Or terminated, and (ii) only non- preemptive scheduling is used by the OS Label the transitions appropriately.

[GATE -2002]

Q.9 A uni-processor computer system only has two processes, both of which alternate 10 ms CPU bursts with 90 ms IN/OUT bursts. Both the processes were created at nearly the same time. The I N/OUT of both processes can proceed in parallel. Which of the following scheduling strategies will result in the least CPU utilization (over a long period of time) for this system?

a) First come first served scheduling
b) Shortest remaining time first scheduling
c) Static priority scheduling with different priorities for the two processes
d) Round robin scheduling with a time quantum of 5 ms

[GATE-2003]

Common Data for Questions 10 and 11:
Suppose we want to synchronize two concurrent processes P and Q using binary semaphores S and T. The code for the processes P and Q is shown below.

```
Process P
While (1)
{
W :
Print '0';
Print '0';
X :
}
Process Q
while (1)
{
Y :
print '1'
print '1'
Z :
```

Synchronization statements can be inserted only at points W, X, Y and Z.

Q.10 Which of the following will always lead to an output starting with? 01100110011'?

a) P(S) at W, V(S) at X, P(T) at Y, V(T) at Z, S and T initially 1
b) P(S) at W, V(T) at X, P(T) at Y, V(S) at Z, S initially 1 , and T initially 0
c) P(S) at W, V(T) at X, P(T) at Y, V(S) at Z, S and T initially 1
d) P(S) at W, V(T) at X, P(T) at Y, V(T) at Z, S initially 1, and T initially 0

[GATE-2003]

Q.11 Which of the following will ensure that the output string never contains a substring of the form 01^n0 or 10^n0 where n is odd?

a) P(S) at W, V(S) at X, P(T) at Y, V(T) at Z, S and T initially 1
b) P(S) at W, V(T) at X, P(T) at Y, V(S) at Z, S and T initially 1,

c) P(S) at W, V(S) at X, P(S) at Y, V(S) at Z, S initially 1
d) P(S) at W, V(T) at X, P(S) at Y, P(T) at Z, S and initially 1

[GATE-2003]

Q.12 Consider the following statements with respect to user-level threads and kernel-supported threads:

1. Context switch is faster with kernel-supported threads.
2. for user-level threads, a system call can block the entire process.
3. Kernel supported threads can be scheduled independently.
4. User level threads are transparent to the kernel.

Which of the above statements are true?

a) 2, 3 and 4 b) 2 and 3
c) 1 and 3 d) 1 and 2

[GATE-2004]

Q.13 Which one of the following is NOT shared by the threads of the same process?

a) Stack
b) Address Space
c) File Descriptor Table
d) Message Queue

[GATE -2004]

Q.14 A process executes the following code

```
for (i=0 ; i < n ; i + +)
fork ( );
```

The number of new processes created is

a) n b) $2^n - 1$
c) 2^n d) $2^{n+1} - 1$

[GATE-2004]

Q.15 Consider the following set of processes with the arrival times and the CPU Burst times given in millisecond

Process	Arrival Time	Burst Time
P_1	0	5
P_2	1	3
P_3	2	3
P_4	4	1

What is the average turn-around time for these processes with the pre-emptive shortest Remaining Processing Time First (SRPTF) algorithm?

a) 5.50 b) 5.75
c) 6.00 d) 6.25

[GATE-2004]

Q.16 Consider two processes P_1 and P_2 accessing the shared variables X and Y protected two binary semaphores S_x and S_y respectively, both initialized to 1. P and V denote the usual semaphore operators, where P decrements the semaphore value, and V increments the semaphore value. The pseudo-code of P_1 and P_2 is as follows:

P_1	P_2
While true do{	While true do{
L_1:	L_3: .....
L_2:	L_4: .....
X=X+1;	Y=Y+1;
Y=Y-1;	X=Y-1;
V(S_x);	V(S_y);
V(S_y);	V(S_x);

In order to avoid deadlock, the correct operators at L_1, L_2, L_3 and L_4 are respectively

a) $P(S_y), P(S_x), P(S_x), P(S_y)$
b) $P(S_x), P(S_y), P(S_y), P(S_x)$
c) $P(S_x), P(S_x), P(S_y), P(S_y)$

d) $P(S_x), P(S_y), P(S_x), P(S_y)$

[GATE-2004]

Q.17 Suppose n processes, $P_1, P_2 \ldots, P_n$ share m identical resource units, which can be reserved and released one at a time. The maximum resource requirement of process P_i is S_i where $S_i > 0$. Which one of the following is a sufficient condition for ensuring that deadlock does not occur?

a) $\forall i, si < m$

b) $\forall i, si < n$

c) $\sum_{i=1}^{n} si < (m+n)$

d) $\sum_{i=1}^{n} si < (m*n)$

[GATE-2005]

Q.18 A user level process in Unix traps the signal sent on a Ctrl- C input, and has a signal handling routine that saves appropriate files before terminating the process. When a Ctrl – C input is given to this process, what is the mode in which the signal handling routine executes?

a) kernel mode
b) super user mode
c) privileged mode
d) user mode

[GATE -2005]

Q.19 We wish to schedule three processes P1, P2 and P3 on a uniprocessor system. The priorities, CPU time requirements and arrival times of the processes are as shown below.

Process	Priority	CPU time required	Arrival time (hh:mm:ss)
P1	10 (highest)	20 sec	0:00:05
P2	9	10 sec	0:00:03
P3	8 (lowest)	15 sec	0:00:00

We have a choice of preemptive or non-preemptive scheduling. In preemptive scheduling, a late arriving higher priority process can preempt a currently running process with lower priority. In non-preemptive scheduling, a late–arriving higher priority process must wait for the currently executing process to complete before it can be scheduled on the processor. What are the turnaround times (time from arrival till completion) of P2 using preemptive and no preemptive scheduling respectively?

a) 30 sec, 30 sec
b) 30 sec 10sec
c) 42 sec 42 sec
d) 30sec 42 sec

[GATE -2005]

Q.20 Two shared resources R_1 and R_2 are used by processes P_1 and P_2. Each process has a certain priority for accessing each resource Let T_{ij} denote the priority of P_i for accessing R_j A process P_i can

snatch a resource R_k from process P_j if T_{ik} is greater than T_{jk}

Given the following:

I. $T_{11} > T_{21}$ II. $T_{12} > T_{22}$

III. $T_{11} < T_{21}$ IV. $T_{12} < T_{22}$

Which of the following conditions ensures that P_1 and P_2 can never deadlock?

a) (I) and (IV) b) (II) and (III)
c) (I) and (II) d)None

[GATE -2005]

Q.21 Consider the following code fragment:

if (fork ( ) == 0)
{a = a + 5 , printf ("%d, %d\n" , a, &a),}
else {a=a-5, printf ("%d, %d\n",a, &a),}

Let u, v be the values printed by the parent process, and x, y be the values printed by the child process. Which one of the following is true?

a) u = x + 10 and v = y
b) u = x + 1 and v ≠ y
c) u + 10 = x and v = y
d) u + 10 = x and v ≠ y

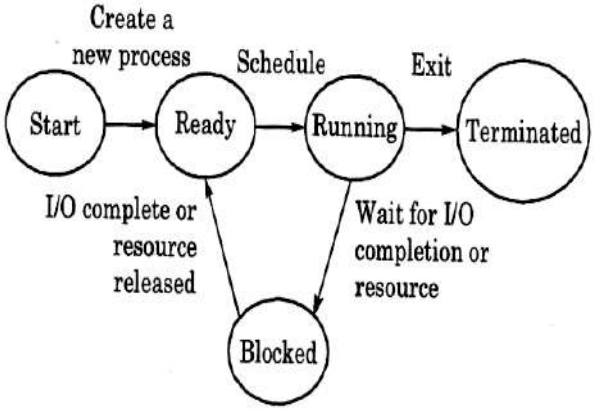
[GATE-2005]

Q.22 Consider three CPU-intensive processes, which require 10, 20 and 30 time units and arrive at times 0, 2 and 6, respectively. Haw many context switches are needed, if time operating system implements a shortest remaining time first scheduling algorithm? Do not count the context switches at time zero and at the end.

a) 1 b) 2
c) 3 d) 4

[GATE-2006]

Q.23 The process state transition diagram of an operating system is as give below.



Which of the following must be FALSE about the above operating system?

a) It is a multi programmed operating system
b) It uses preemptive scheduling
c) It uses non preemptive scheduling
d) It is an multi–user operating system

[GATE -2006]

Q.24 Consider three processes, all arriving at time zero, with total execution time of 10, 20 and 30 units, respectively. Each process spends the first 20% of execution time doing IN/OUT, the 'next 70% of time doing computation, and the last 10% of time doing IN/OUT again. The operating system uses a shortest remaining compute time first scheduling algorithm and schedules a new process either when the at process gets blocked an IN/OUT or when the running process finishes its compute burst. Assume that all IN/OUT operations can be overlapped as much as possible. For what percentage of time does the CPU remain idle?

a) 0% b) 10.6%
c) 30.0% d) 89.4%

[GATE-2006]

Q.25 Consider three processes (process id 0, 1, 2 respectively) with compute time bursts 2, 4 and 8 time units. All processes arrive at time zero. Consider the Longest Remaining Time First (LRTF) scheduling algorithm. In LRTF ties are broken by giving priority to the process with the lowest process id. The average turnaround time is

a) 13 unit b) 14 unit
c) 15 unit d) 16 unit

[GATE-2006]

Q.26 Consider the following, snapshot of a system running n processes. Process i is holding X_i instances of a resource R, $l \le i \le n$. Currently, all instances of R are occupied. Further, for all i process i has placed a request for an additional Y_i instances while holding the X_i instances it already has. There are exactly two processes p and q such that $Y_p = Y_q = 0$. Which one of the following can serve as a necessary condition to guarantee that the system is not approaching a deadlock?

a) min (X_p , X_q)<MAX(Y_k) where k<> p and k<> q.
b) $X_p + X_q \ge Min(Y_k)$ where k<>p & k<>q.
c) max(X_p , X_q) > 1
d) min (X_p , X_q) > 1

[GATE-2006]

Q.27 The atomic fetch-and-set x, y instruction unconditionally sets the memory location x to 1 and fetches the old value of x in y without allowing any intervening access to the memory location x. Consider the following implementation of P and V functions on a binary semaphore S.

```
void P ( binary_semaphore *s) {
unsigned y;
unsigned *x = & (S -> value);
do {
-- fetch-and-set x , y;
}          while (y);
void V (binary_semaphore  *S) {
          s -> value = 0 ;
          }
```

Which one of the following is true?

a) The implementation may not work, if context switching is disabled in P
b) Instead of using fetch-and-set, a pair of normal load/store can be used
c) The implementation of V is wrong
d) The code does not implement a binary semaphore

[GATE-2006]

Statements for Linked Answer Questions 28 and 29

Barrier, is a synchronization construct where a set of processes synchronizes globally, i.e., each process in the set arrives at the barrier and waits for all others to arrive and then all processes leave the Barrier. Let the number of processes in the set be 3 and S be a binary semaphore with the usual P and V functions.

Consider the following C implementation of a barrier with line numbers shown on left :

```
void Barrier (void) {
1 : p(s) ;
2 : process_arrived ++,
3 : V(S) ;
4 : while (process_arrived != 3);
5 : P(S)
```

```
6 : Process_left++ ;
7 : if (process_left ==3) ; {
8 : process_arrived = 0;
9 : process_left = 0;
10 : }
11 : V(S) ;
        }
```

The variables process arrived and process left are shared among all processes and are initialized to zero. In a concurrent program all the three processes call the barrier function when they need to synchronize globally.

Q.28 The above implementation of barrier is incorrect. Which one of the following is true?

a) The barrier implementation is wrong due to the use of binary semaphore S
b) The barrier implementation may lead to a deadlock, if two barrier invocations are used in immediate succession
c) Lines 6 to 10 need not be inside a critical section
d) The barrier implementation is correct, if there are only two processes instead of three

[GATE-2006]

Q.29 Which One of the following rectifies the problem in the implementation?

a) Lines 6 to 10 are simply replaced by process arrived
b) At the beginning of the barrier the first process to enter the barrier waits until process arrived becomes zero before proceeding to execute P(S)
c) Context switch is disabled at the beginning of the barrier and re-enabled at the end
d) The variable process left is made private instead of shared

[GATE-2006]

Q.30 Two processes P_1 and P_2, need to access a critical section of code. Consider the following synchronization construct used by the processes:

```
/*P1 */
While (true) {
 Wants 1 = true ;
 While
(wants 2 == true) ;
/* Critical
Section */
Wants1 = false ;
}  (/* Remainder section)
/*      P2      */
while (true){
wants 2 = true ;
while (wants1 == true) ;
/* Critical
Section */
Wants2 = false ;
/* Remainder section */
```

Here, wants1 and wants2 arc shared variables, which are initialized to false. Which one of the following statements is true about the above construct?

a) It does not ensure mutual exclusion
b) It does not ensure hounded waiting
c) It requires that processes enter the critical section in strict alternation
d) It does not prevent deadlocks, but ensures mutual exclusion

[GATE-2007]

Q.31 List I contains some CPU scheduling algorithms and List II contains some applications. Match entries in List I to entries in List II using the codes given –below the lists.

	List I		List II
P.	Gang Scheduling	1.	Guaranteed scheduling
Q.	Rate Monotonic Scheduling	2.	Real-time Scheduling
R.	Fair share Scheduling	3.	Thread Scheduling

Codes

a) P-3, Q-2, R-1
b) P-1, Q-2, R-3
c) P-2, Q-3, R-1
d) P-1, Q-3, R-2

[GATE-2007]

Q.32 Consider the following statements about user level threads and kernel level threads: Which one of the following statements is false?

a) Context switch time is longer for kernel level threads than for user level threads
b) User level threads do not need any hardware support
c) Related kernel level threads can be scheduled on different processors in a multi-processor system
d) Blocking one kernel level thread Blacks all related threads

[GATE-2007]

Q.33 An operating system uses Shortest Remaining Time First (SRTF) process scheduling algorithm. Consider the arrival times and execution times for the following

Process	Execution Time	Arrival Time
P_1	20	0
P_2	25	15
P_3	10	30
P_4	15	45

Which is the total waiting time for process P_2?

a) 5
b) 15
c) 40
d) 55

[GATE-2007]

Q.34 A single processor system has three resources types X, Y and Z, which are shared by three processes. There are .5 units of each resources type. Consider the following scenario, where the column alloc denotes the number of units of each resource type allocated to each process, and the column request denotes the number of units of each resource type requested by a process in order to complete execution. Which of these processes will finish last?

	Alloc			request		
	X	Y	Z	X	Y	Z
P_0	1	2	1	1	0	3
P_1	2	0	1	0	1	2
P_2	2	2	1	1	2	0

a) P_0
b) P_1
c) P_2
d) None of the above, since is in a deadlock

[GATE-2007]

Q.35 Which of the following is not true of dead lock prevention and deadlock avoidance schemes?

a) In deadlock prevention, the request for resources is always granted, if the resulting state is safe
b) In deadlock avoidance, the request for resources is always granted, if the resulting state is safe
c) Deadlock avoidance is less restrictive than deadlock prevention

d) Deadlock avoidance requires knowledge of resource requirements a priori

[GATE-2008]

Q.36 If the time–slice used in the round– robin scheduling policy is more that the maximum time required to executed any process, then the policy will

a) degenerate to shortest job first

b)degenerate to priority scheduling

c) degenerate to first come first serve

d) None of the above

[GATE -2008]

Q.37 Which of the following statements about synchronous and asynchronous IN/OUT is not true?

a) An ISR is invoked on completion of IN/OUT in synchronous IN/OUT but not in asynchronous IN/OUT

b) In both synchronous and asynchronous IN/OUT, an ISR. (Interrupt Service Routine) is invoked after completion of the IN/OUT

c) A process making a synchronous IN/OUT call waits until IN/OUT is complete, but a process making an asynchronous IN/OUT call does not wait for completion of the INI/OLT

d) in the case of synchronous IN/OUT, the process waiting for the completion of IN/OUT is woken up by the ISR that is invoked after the completion of I IN/OUT

[GATE-2008]

Q.38 A process executes the following code for (i =0,i<n;i++)fork ( );

The total number of child processes created is

a) n b) $2^n - 1$

c) 2^n d) $2^{n+1} - 1$

[GATE -2008]

Q.39 The P and V operations on counting semaphores, where s is a counting semaphore, are defined as follows

P(s) : s = s - 1;

If s < 0 then wait;

V(s) : s = s + 1;

if s <= 0 then wake up a process waiting on s;

Assume that P_b and V_b the wait and signal operations on binary semaphores are provided. Two binary semaphores X_b and Y_b are used to implement the semaphore operations P(s) and V(s) as follows

P(S) : P_b, (X_b);

S = s-1 ;

If (s < 0) (

V_b (X_b) ;

V_b (Y_b) ;

}

Else V_b (X_b) ;

V(S) : P_b (Y_b) ;

S = s + 1 ;

If (s <= 0) V_b (Y_b) ;

V_b (X_b) ;

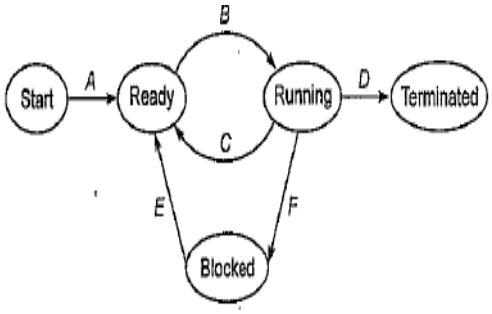
The initial values of X_b, and Y_b are respectively

a) 0 and 0 b) 0 and 1

c) 1 and 0 d) 1 and 1

[GATE-2008]

Q.40 In the following process state transition diagram for a uniprocessor system, assume that there are always some processes in the ready state.



Now consider the following statements:

1. If a process makes a transition D, it would result in another process making transition A immediately.
2. A process P_2 in blocked state can make transition E while another process P_1 is in running state
3. The operating system uses pre-emptive scheduling.
4. The operating system uses non-pre-emptive scheduling.

Which of the above statements are true?

a) 1 and 2 b) 1 and 3
c) 2 and 3 d) 2 and 4

[GATE-2009]

Q.41 The enter CSO and leave CSO functions to implement critical section of a process are realized using test-and set instruction as follows

```
void enter_CSC(X)
{
while (test-and-sex (X));
}
void leave (X)
{
X = 0;
}
```

In the above solution, X is a memory location associated with the CS and is initialized to 0. Now, consider the following statements:

1. The above solution to CS problem is deadlock-free.
2. The solution is starvation-free.
3. The processes enter CS in FIFO order.
4. More than one processes can enter CS at the same time

Which of the above statements is true?

a) 1 only b) 1 and 2
c) 2 and 3 d) 4 only

[GATE-2009]

Q.42 Consider a system with 4 types of resources R_1 (3 unit), R_2 (2 Unit), R_3 (3 unit), R_4 (2 unit). A non-pre-emptive resource allocation policy is used. At any given instance, a request is not entertained if it cannot be completely satisfied. Three processes P_1, P_2, P_3 request the resources as follows, if executed independently.

Process P_1 :	Process P_2 :	Process P_3 :
t = 0 : requests 2 unit of R_2	t = 0 : requests 2 unit of R_3	t = 0 : requests 1 unit of R_4
t = 1 : requests 1 unit of R_3	t = 2 : requests 1 unit of R_4	t = 2 : requests 2 unit of R_1
t = 3 : requests 2 unit of R_1	t = 4 : requests 1 unit of R_1	t = 5 : requests 2 unit of R_1
t = 5 : releases 1 unit of R_2 and 1 unit of R_1	t = 6 : releases 1 unit of R_3	t = 7 : requests 1 unit of R_2
t = 7 : releases 1 unit of R_3	t = 8 : Finishes	t = 8 : requests 1 unit of R_3
t = 8 : requests 2 unit of R_4		t = 9 : Finishes
t = 10 : Finishes		

Which one of the following statements is true, if all three processes run concurrently starting at time t=0?

a) All processes will finish without any deadlock
b) P_1 and P_2 will be in deadlock
c) P_1 and P_3 will be in a deadlock
d) All three processes will be in deadlock

[GATE-2009]

Q.43 A system has n resources R_0,.....,R_{n-1}, & k processes P_0,.....,P_{k-1}. The implementation of the resources request logic of each process P_1, is as follows

```
If (i % 2 = = 0) {
        if (i < n) request Ri ;
        if (i + 2 < n) request R1+2 ;
}
Else {
        if (i < n) request Rn-1 ;
        if (i + 2 < n) request Rn-i-2 ;
}
```

In which one of the following situations is a deadlock possible?

a) n = 40, k = 26 b) n= 21, k = 12

c) n = 20, k = 10 d) n= 41, k = 19

[GATE-2010]

Q.44 Consider the methods used by processes P_1 and P_2 for accessing their critical sections whenever needed, as given below. The initial values of shared Boolean variables S_1 and S_2 are randomly assigned.

Method used by P_1	Method used by P_2
While(S_1==S_2);	While(S_1!=S_2);
Critical section	Critical section
S_1=S_2;	S_2=not(S_1);

Which one of the following statements describes the properties achieved?

a) Mutual exclusion but not progress
b) Progress but not mutual exclusion
c) Neither mutual exclusion nor progress
d) Both mutual exclusion & progress

[GATE-2010]

Q.45 The following program consists of 3 concurrent processes and 3 binary semaphores. The semaphores are initialized as S_0 =1, S_1 = 0, S_2 = 0.

Process P_0	Process P_1	Process P_2
Wait(S_0);	Wait(S_1);	Wait(S_2);
Print'0'	Release (S_0);	Release (S_0);
Release (S_1);		
Release (S_2);		
}		

How many times will process P_0 print '0'?

a) Atleast twice
b) Exactly twice
c) Exactly thrice
d) Exactly once

[GATE-2010]

Q.46 Which of the following statements are true?

I. Shortest remaining time first scheduling may cause starvation.
II. Pre-emptive scheduling may cause starvation. III. Round robin is better than FCFS in terms of response time.

a) I only b) I and III
c) II and III d) I, II and III

[GATE-2010]

Q.47 Let the time taken to switch between user and kernel modes of execution be t_1 while the time taken to switch between two processes be t_2 which of the following is true?

a) $t_1 > t_2$

b) $t_1 = t_2$
c) $t_1 < t_2$
d) Nothing can be said about the relation between t_1 and t_2

[GATE-2011]

Q.48 A computer handles several interrupt sources of which of the following are relevant for this question? Interrupt from CPU temperature sensor
Interrupt from Mouse
Interrupt from Keyboard
Interrupt from Hard Disk
Which one of these will be handled at the HIGHEST priority?

a) Interrupt from Hard Disk
b) Interrupt from Mouse
c) Interrupt from Keyboard
d) Interrupt from CPU temperature sensor

[GATE-2011]

Q.49 A thread is usually defined as a 'light weight process' because an Operating System (OS) maintains smaller data structures for a thread than for a process. In relation to this, which of the following is true?

a) On per thread basis, the operating system maintains only CPU register state
b) The operating system does not maintain a separate stack for each thread
c) On per thread basis, the operating system does not maintain virtual memory state
d) On per thread basis, the operating system maintains only scheduling and accounting information

[GATE-2011]

Q.50 Consider the following table of arrival time and burst time for three processes
P_o, P_1 and P_2.

Process	Arrival Time	Burst Time
P_1	0 ms	9 ms
P_2	1 ms	4 ms
P_3	2 ms	9 ms

The pre-emptive shortest job first scheduling algorithm is used. Scheduling is carried out only at arrival or completion of processes. What is the average waiting time for the three processes?

a) 5.0 ms b) 4.33 ms
c) 6.33 ms d) 7.33 ms

[GATE-2011]

Q.51 A process executes the code
fork ( ) ;
fork ( ) ;
fork ( ) ;
The total number of child processes created is

a) 3 b) 4
c) 7 d) 8

[GATE-2012]

Q.52 Consider the 3 processes, P_1, P_2 and P_3 shown in the table:

Process	Arrival Time	Time units required
P_1	0	5
P_2	1	7
P_3	3	4

The completion order of the 3 processes under the policies FCFS and RR2 (Round Robin scheduling with CPU quantum 2 time units) are

a) FCFS : P_1 , P_2 , P_3 RR2 : P_1 , P_2 , P_3
b) FCFS: P_1 , P_3 , P_2 RR2 : P_1 , P_3 , P_2
c) FCFS: P_1 , P_2 , P_3 RR2 : P_1 , P_3 , P_2

d) FCES: P_1 , P_2 , P_3 RR2 : P_1 , P_2 , P_3

[GATE-2012]

Q.53 Fetch_And_Add (X, i) is an atomic Read-Modify-Write instruction that reads the value of memory location X, increments it by the value i and returns the old value of X. It is used in the pseudo code shown below to implement a busy-wait lock. L is an unsigned integer shared variable initialized to 0. The value of 0 corresponds to lock being available, while any non-zero value corresponds to the lock being not available.

```
AcquireLock (L) {
While (Fetch_And_Add (L, 1)}
L = 1;
}
ReleaseLock (L) {
L = 0;
}
```

This implementation

a) fails as L can overflow
b) fails as L can take on a non-zero value that lock is actually available
c) works correctly but may starve some processes
d) works correctly without starvation

[GATE-2012]

Q.54 Three concurrent processes X, Y and Z execute three different code segments that access and update certain shared variables. Process X executes the P operation (i.e., wait) on semaphores a, b and c; process Y executes the P operation on semaphores b, c and d; process Z executes the P operation on semaphores c, d and a before entering the respective code segments. After completing the execution of its code Segment, each process invokes the V, operation (i.e., signal) on its three semaphores. All semaphores are binary semaphores initialized to one. Which one of the following represents a deadlock-free order of invoking the P operations by the processes?

a) X : P(a) P(b) P(c) Y : P(b) P(c) P(d) Z : P(c) P(d) P(a)
b) X : P(b) P(a) P(c) Y : P(b) P(c) P(d) Z : P(a) P(c) P(d)
c) X : P(b) P(a) P(c) Y : P(c) P(b) P(d) Z : P(a) P(c) P(d)
d) X : P(a) P(b) P(c) Y : P(c) P(b) P(d) Z : P(c) P(d) P(a)

[GATE-2013]

Q.55 A scheduling algorithm assigns priority proportional to the waiting time or a process. Every process starts with priority zero (the lowest priority). The scheduler re-evaluates the process priorities every T time units and decides the next process to schedule. Which one of the following is true if the processes have no I/O operations and all arrive at time zero?

a) This algorithm is equivalent to the first come first serve algorithm
b) This algorithm is equivalent to the round robin algorithm
c) This algorithm is equivalent to the shortest job first algorithm
d) This algorithm is equivalent to the shortest remaining, time first algorithm

[GATE-2013]

Q.56 A shared variable x, initialized to zero, is operated on by four concurrent processes W, X, Y, Z as

follows. Each of the processes W and X reads x from memory increments by one, Stores it to memory, and then terminates. Each of processes Y, and Z reads x from memory, decrements by two, stores it to memory and then terminates. Each process before reading x Invokes the P operation (i.e., wait) on a counting semaphore S after storing x to memory. Semaphore S is initialized to two. What is the maximum possible value of x after all processes complete execution?

a) -2 b) -1
c) 1 d) 2

[GATE-2013]

Q.57 A certain computation generates two arrays a and b such that a [1] = f(i) for 0≤I<n and b[i] = g {a [i]} for 0≤ I < n. Suppose this computation is decomposed into two concurrent processes X and Y such that X computes the array a and Y computes the array b. The processes employ two binary semaphores R and S, both initialized to zero. The array is shared by the two processes. The structure of the processes are shown below.

```
Process X:
Private 1:
for (i= 0; i< n; i++) {
a[i] = f (i) ;
Exit X (R, S) ;
}
Process Y:
private i;
for (i= 0; i< n; i++)
{
Entry y (R, S) ;
b[i] = g (a [i]) ;
}
```

Which one of the following represents the correct implementations of Exit X and Entry Y ?

a)
```
ExitX (R, S) {
P(R) ;
V(S) ;
}
EntryY (R, S) {
P(S) ;
V(R) ;
}
```

b)
```
ExitX (R, S) {
V(R) ;
V(S) ;
}
EntryY (R, S) {
P(R) ;
P(S) ;
}
```

c)
```
ExitX (R, S) {
P(S) ;
V(R) ;
}
EntryY (R, S) {
V(S) ;
P(R) ;
}
```

d)
```
ExitX (R, S) {
V(R) ;
P(S) ;
}
EntryY (R, S) {
V(S) ;
P(R) ;
}
```

[GATE-2013]

Q.58 Consider the procedure below for the Producer-Consumer problem which uses semaphores:-

```
semaphore n = 0
semaphore s = 1

void producer( )
{
    while (true)
    {
        produce( );
        semWait(s);
        addToBuffer( );
        semSignal(s);
        semSignal(n);
    }
}

void consumer( )
{
    while(true)
    {
        semWait(s);
        semWait(n);
        removeFromBuffer( );
        semSignal(s);
        consume( );
    }
}
```

Which one of the following is TRUE? The producer will be able to add the item to the buffer, but the consumer can never consume it. The consumer will remove no more than one item from the buffer. Deadlock occurs if the consumer succeeds in acquiring semaphore "s" when the buffer is empty. The starting value for the semaphore "n" must be 1 and not 0 for deadlock-free operation.

[GATE–2014]

Q.59 Which of the following is FALSE?

a) User level threads are not scheduled by the kernel.
b) When a user level thread is blocked, all other threads of its process are blocked.
c) Context switching between user level threads is faster than context switch between kernel level threads.
d) Kernel level threads cannot share code segment.

[GATE–2014]

Q.60 Consider the following set of processes that need to be scheduled on a single CPU. All the times are given in milliseconds.

Process	Arrival Time	Execution Time
A	0	6
B	3	2
C	5	4
D	7	6
E	10	3

Using the shortest remaining time first scheduling algorithm, the average turnaround time (in m sec) is___.

[GATE–2014]

Q.61 Three processes A, B and C each execute a loop of 100 iterations. In each iteration of the loop, a process performs a single computation that requires t_c CPU milliseconds and then initiates single I/O operation that lasts for t_{io} milliseconds. It is assumed that the computer where the processes execute has sufficient number of I/O devices and the OS of the computer assigns different I/O devices to each process. Also, the scheduling overhead of the OS is negligible. The processes have the following characteristics:

Process Id	t_c	t_{io}
A	100 ms	500 ms
B	350 ms	500 ms
C	200 ms	500 ms

The processes A, B, and C are started at times 0, 5, and 10 milliseconds respectively in a pure time sharing system [round robin scheduling] that uses a time slice of 50 milliseconds. The time in milliseconds at which process "C" would complete its first I/O operation is _____

[GATE–2014]

Q.62 An operating system uses shortest remaining time first scheduling algorithm for pre-emptive scheduling of processes. Consider the following set of processes with their arrival times and CPU burst times (in milliseconds):

Process	Arrival Time	Burst Time
P_1	0	12
P_2	2	4
P_3	3	6
P_4	8	5

The average waiting time (in milliseconds) of the processes is _____.

[GATE–2014]

Q.63 An operating system uses the Banker's algorithm for deadlock avoidance when managing the allocation of three resource types X, Y, and Z to three processes P0, P1, and P2. The table given below presents the current safe state. Here, the Allocation matrix shows the current number of resources of each type allocated to each process and the Max matrix shows the maximum number of resources of each type required by each process during its execution

Process	Allocation			Max Needs		
	X	Y	Z	X	Y	Z
P0	0	0	1	8	4	3
P1	3	2	0	6	2	0
P2	2	1	1	3	3	3

There are 3 units of type X, 2 units of type Y, and 2 units of type Z still available. The system is currently in a safe state. Consider the following independent requests for additional resources in the current state:

REQ1: P0 requests 0 units of X, 0 units of Y, and 2 units of Z
REQ2: P1 requests 2 units of X, 0 units of Y, and 0 units of Z

Which one of the following is TRUE?

a) Only REQ1 can be permitted
b) Only REQ2 can be permitted
c) Both REQ1 and REQ2 can be permitted
d) Neither REQ1 nor REQ2 can be permitted.

[GATE–2014]

Q.64 A system contains three programs and each requires three tape units for its operation. The minimum number of tape units which the system must have such that deadlocks never arise is ___

[GATE–2014]

Q.65 A system has 6 identical resources and N processes competing for them. Each process can request at most 2 resources. Which one of the following values of N could lead to a deadlock?

a) 1 b) 2
c) 3 d) 4

[GATE–2015]

Q.66 Consider the following policies for preventing deadlock in a system with mutually exclusive resources.

I. Processes should acquire all their resources at the beginning of execution. If any resources acquired so far are released.
II. The resources are numbered uniquely, and processes are allowed to request for resources only in increasing resource numbers.

III. The resources are numbered uniquely, and processes are allowed to request for resources only in decreasing resource numbers.

IV. The resources are numbered uniquely. A process is allowed to request only for a resource with resource number larger than it's currently held resources.

Which of the above policies can be used for preventing deadlock?

a) Any one of I and III but not II or IV

b) Any one of I, III, and IV but not II

c) Any one of II and III but not I or IV

c) Any one of I, II, III, and IV

[GATE–2015]

Q.67 Consider a uniprocessor system executing three tasks T1, T2 and T3, each of which is composed of an infinite sequence of jobs (or instances) which arrive periodically at intervals of 3, 7 and 20 milliseconds, respectively. The priority of each task is the inverse of its period and the available tasks are scheduled in order of priority, with the highest priority task scheduled first. Each instance of T1, T2 and T3 requires an execution time of 1, 2 and 4 milliseconds, respectively. Given that all tasks initially arrive at the beginning of the 1st millisecond and task preemptions are allowed, the first instance of T3 completes its execution at the end of _____ milliseconds.

[GATE–2015]

Q.68 The maximum number of processes that can be in Ready state for a computer system with n CPU's is

a) n b) n^2

c) 2^n d) Independent of n

[GATE–2015]

Q.69 For the processes listed in the following table, which of the following scheduling schemes will the lowest average turnaround time?

PROCESS	ARRIVAL TIME	PROCESSING TIME
A	0	3
B	1	6
C	4	4
D	6	2

a) First Come First Serve

b)Non-preemptive Shortest Job First

c) Shortest Remaining Time

d) Round Robin with Quantum value two

[GATE–2015]

Q.70 Two processes X and Y need to access a critical section. Consider the following synchronization construct used by both the processes

Process X	Process Y

/*other code for process X*/
while(true)
{
varP = true;
while(varQ == true)
{
/*critical section*/
varP= false;
}
}
/*other code for process X*/
	/*other code for process Y*/
	while(true)
	{
	varQ = true;
	while(varP == true)
	{
	/*Critical section*/
	varQ= false;
	}
	}
	/*other code for process Y*/

Here, var P and varQ are shared variables and both are initialized

to false. Which one of the following statements is true?

a) The proposed solution prevents deadlock but fails to guarantee mutual exclusion
b) The proposed solution guarantees mutual exclusion but fails to prevent deadlock
c) The proposed solution guarantees mutual exclusion and prevents deadlock
d) The proposed solution fails to prevent deadlock and fails to guarantee mutual exclusion

[GATE–2015]

Q.71 Consider the following proposed solution for the critical section problem. There are n processes: P0.....Pn−1. In the code, function pmax returns an integer not smaller than any of its arguments. For all i, t[i] is initialized to zero.

```
Code for Pi:
do {
c[i]=1; t[i] = pmax(t[0],...,t[n-1])+1;
c[i]=0;
for every j ≠ i in {0,...,n-1}
{
while (c[j]);
while (t[j] != 0 && t[j]<=t[i]);
}
Critical Section;
t[i] = 0;
Remainder Section;
} while (true);
```

Which one of the following is TRUE about the above solution?

a) At most one process can be in the critical section at any time
b) The bounded wait condition is satisfied
c) The progress condition is satisfied
d) It cannot cause a deadlock

[GATE–2016]

Q.72 Consider the following two-process synchronization solution

Process 0	Process 1
Entry:loop	Entry:loop
while(turn==1);	while(turn==0);
(critical section)	(critical section)
Exit: turn=1;	Exit: turn=0;

The shared variable turn is initialized to zero.

Which one of the following is TRUE?

a) This is a correct two-process synchronization solution.
b) This solution violates mutual exclusion requirement.
c) This solution violates progress requirement.
d) This solution violates bounded wait requirement

[GATE–2016]

Q.73 Consider a non-negative counting semaphore S. The operation P(S) decrements S, and V(S) increments S. During an execution, 20 P(S) operations and 12 V(S) operations are issued in some order. The largest initial value of S for which at least one P(S) operation will remain blocked is_____.

[GATE–2016]

Q.74 Consider an arbitrary set of CPU-Bound processes with unequal CPU burst lengths submitted at the same time to a computer system. Which one of the following process scheduling algorithms would minimize the average waiting time in the ready queue?

a) Shortest remaining time first
b) Round-robin with time quantum less than the shortest CPU burst
c) Uniform random

d) Highest priority first with priority proportional to CPU burst length

[GATE–2016]

Q.75 Consider the following processes, with the arrival time and the length of the CPU burst given in milliseconds. The scheduling algorithm used is preemptive shortest remaining-time first.

Process	Arrival Time	Burst Time
P_1	0	10
P_2	3	6
P_3	7	1
P_4	8	3

The average turnaround time of these processes is _____ milliseconds.

[GATE–2016]

Q.76 Threads of a process share
a) global variables but not heap
b) heap but not global variables
c) neither global variables nor heap
d) both heap and global variables

[GATE–2017]

Q.77 Consider the following CPU processes with arrival time (in milliseconds) and length of CPU bursts (in milliseconds) as given below:

Process	Arrival time	Burst time
P_1	0	7
P_2	3	3
P_3	5	5
P_4	6	2

If the pre-emptive shortest remaining time first scheduling algorithm is used to schedule the process, then the average waiting time across all processes is_____ milliseconds.

[GATE–2017]

Q.78 A multithreaded program P executes with x number of threads and uses y number of locks for ensuring mutual exclusion while operating on shared memory locations. All locks in the program are non- reentrant. i.e. if a thread holds a lock l. then it cannot re-acquire lock l without releasing it, If thread is unable to acquire a lock , it blocks until the lock becomes available. The Minimum value of x and the minimum value of y together for which execution of P can result in a deadlock are:
a) x= 1,y= 2 b) x= 2, y=1
c) x=2, y=2 d)x=1,y=1

[GATE–2017]

Q.79 Which of the following is/are shared by all the threads in a process?
I. Program counter II. Stack
III. Address space IV. Registers
a) I and II only b) III only
c) IV only d)III & IV only

[GATE–2017]

Q.80 Consider the set of processes with arrival time (in milliseconds), CPU burst time (in milliseconds), ad priority (0 is the highest priority) shown below. None of the processes have IO burst time

Process	Arrival Time	Burst Time	Priority
P_1	0	11	2
P_2	5	28	0
P_3	12	2	3
P_4	2	10	1
P_5	9	16	4

The average waiting time (in milliseconds) of all the processes using preemptive priority scheduling algorithm is ___.

[GATE–2017]

Q.81 A system share 9 type drives. The current allocation and maximum requirement of tape drives for three process are shown below:

Process	Current Allocation	Max. Requirement
P1	3	7
P2	1	6
P3	3	5

Which of the following best describes current state of the system?

a) Safe , Deadlocked
b)Safe Not Deadlocked
c) Not Safe , Deadlocked
d)Not Safe , Not Deadlocked

[GATE–2017]

Q.82 Consider a system with 3 processes that share 4 instances of the same resource type. Each process can request a maximum of K instances. Resource instances can be requested and released only one at a time. The largest value of K that will always avoid deadlock is_____.

[GATE–2018]

Q.83 In a system, there are three types of resources: 0 E, F and G. Four processes P_0, P_1, P_2 and P_3 execute concurrently. At the outset, the processes have declared their maximum resource requirements using a matrix named Max as given below. For example, Max [P_2,F] is the maximum number of instances of F that P_2 would require. The number of instances of the resources allocated to the various processes at any given state is given by a matrix named Allocation. Consider a state of the system with the Allocation matrix as shown below, and in which 3 instances of E and 3 instances of F are the only resources available.

Allocation			
	E	F	G
P0	1	0	1
P1	1	1	2
P2	1	0	3
P3	2	0	0

Max			
	E	F	G
P0	4	3	1
P1	2	1	4
P2	1	3	3
P3	5	4	1

From the perspective of deadlock avoidance, which one of the following is true?

a) The system is in safe state.
b) The system is not in safe state, but would be safe if one more instance of E were available
c) The system is not in safe state, but would be safe if one more instance of F were available
d) The system is not in safe state, but would be safe if one more instance of G were available

[GATE–2018]

ANSWER KEY:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
(c)	(d)	(b)	(b)	(b)	(b)	(b)	-	(a)	(b)	(c)	(b)	(a)	(b)	(a)
16	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**
(d)	(c)	(d)	(d)	(c)	(c)	(b)	(b)	(b)	(a)	(b)	(a)	(b)	(b)	(d)
31	**32**	**33**	**34**	**35**	**36**	**37**	**38**	**39**	**40**	**41**	**42**	**43**	**44**	**45**
(a)	(d)	(b)	(c)	(a)	(c)	(a)	(b)	(c)	(c)	(a)	(a)	(b)	(a)	(a)
46	**47**	**48**	**49**	**50**	**51**	**52**	**53**	**54**	**55**	**56**	**57**	**58**	**59**	**60**
(d)	(c)	(d)	(c)	(a)	(c)	(c)	(b)	(b)	(b)	(d)	(c)	(c)	(d)	7.2
61	**62**	**63**	**64**	**65**	**66**	**67**	**68**	**69**	**70**	**71**	**72**	**73**	**74**	**75**
1000	5.5	(b)	7	4	(d)	12	(d)	(c)	(a)	(a)	(c)	7	(a)	8.25
76	**77**	**78**	**79**	**80**	**81**	**82**	**83**							
(d)	3	(c)	(b)	29	(b)	2	(a)							

EXPLANATIONS

Q.1 **(c)**

To execute privileged instructions, system services can be obtained using software interrupt.

Q.2 **(d)**

Because we want to change the mode from privileged to non-privileged, to the next instruction to be executed should be non-privileged instruction.

Q.3 **(b)**

Throughput is defined as the measure of the number of processes completed per unit time and if the CPU is busy in executing some process than the work is being done. If the number of processors is increased then amount of work done is increased and time is decreased. In case of uni-processor first come first serve gives the maximum throughput but when the case of known run-time is considered shortest job first is used which is also used in case of turn around. :

Q.4 **(b)**

Non-pre-emption means that once the CPU has been allocated to a process, the process keeps the CPU until it releases the CPU either by terminating the process or by switching to waiting state. Thus, FIFO, i.e., first in first out algorithm is non-pre-emptive as it does not

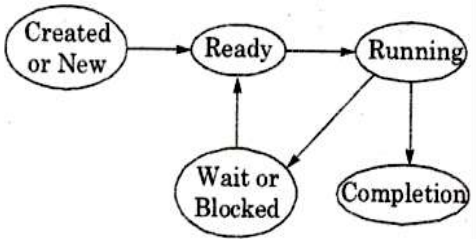
pre-empty the CPU while it is been executed.

Q.5 **(b)**
Swapping requires a backing store which is commonly a disk. Swap space is used by the operating system in many ways. These spaces are put on separate disks so that the load on the input system can be distributed over the system input devices.

Q.6 **(b)**
The condition of mutual exclusion is satisfied when single data is sharable with more than one processor.
For the given program, the following should be executed:
Flag (i) = true;
Turn = j;
Then, flag (j) = true
Turn = j
Now, to enter into the critical section, first flag should be true and then, turn should be done.

Q.7 **(b)**
Both A and B conditions are necessary to characterize an operating system as a multi programmed operating system.
Option C) is characteristic of both multi programmed and single programmed OS.

Q.8 Just draw the diagram by using all criteria as below:



Q.9 **(a)**
FCFS produces least CPU utilization when two: processes do both IN/OUT and computing whereas round robin produces maximum CPU utilization.

Q.10 **(b)**
Let's initialize S = 1 and T = 0.
This initialization will produce a sequence of processes P, Q, P, Q...
Therefore, P (S) at W, V (T) at X
P (T) at Y, V(S) at Z

Q.11 **(c)**
As per the given and the answer found in previous answer,
S is initialized to 1.
The code for process P and process Q is given as

Process P:	Process Q:
Whiled) {	While (1) {
W: P(S)	Y: P(S)
Print'0';	Print'1';'
Print'0';	Print'1'
X: V(S)	Z: V(S)
}	}

Now, in the code,
In process P,
P(S) is inserted at W and V(S) is inserted at X in
process P.
In process Q,
(S) is inserted at Y and V(S) is inserted at Z

Q.12 **(b)**
Statement I False: The context switch is slower with kernel supported threads.
Statement II True: For user level threads, a system call can block the entire system. The statement is true as it is the drawback of user

level threads blocking system call can block the entire process.

Statement III True: Kernel supported threads can be scheduled independently as kernel supported threads have their own areas thus, are scheduled independently.

Statement IV False: User level threads can never be transparent to the kernel.

Q.13 (a)

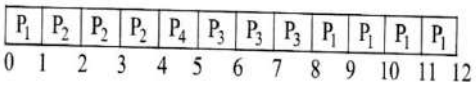
Stack and registers are not shared by the threads of the same process while address space, message queue etc, are shared.

Q.14 (b)

We know that the total number of processes is 2^n but we subtract the main process therefore, total number of children = $2^n - 1$

Q.15 (a)

chart for SRPT algorithm is



Turnaround time

P_1 =12

P_2 =4-1=3

P_3 = 6

P_4 =1

Average =12 + 3 + 6 + $\frac{1}{4} = \frac{22}{4} = 5.50$

Q.16 (d)

Let us assume that P means wait and V means signal. When P_1 issues signal wait(x) at that time, process P_2 issues wait (y). So, like this we take alternatively then there will be no chance of 'deadlock. So, the sequence is $P(S_X), P(S_Y); P(S_X), P(S_Y)$.

Q.17 (c)

In the extreme condition, all processes acquire Si-1 resources and need 1 more resource. So following condition must be true to make sure that deadlock never occurs. So $\sum_{i=1}^{n} Si < (m+n)$.

Q.18 (d)

When user level process trapping the Ctrl+ C signal then the trap signal is going through system call and that's why mode changed to kernel mode from user mode and then the request is handling. One more thing kernel mode and privilege mode are same; answer is kernel mode (privilege mode).

Q.19 (d)

TAT=Completion Time –Arrival Time. The Gantt chart for Non Preemptive scheduling will be (0) P3, (15) P1, 35 P2 (45).

From above this can be inferred easily that completion time for P2 is 45, for P1 is 35 and P3 is 15.

Gantt Chart for Preemptive (0) P3, (1) P3, (2) P3, (3) P2, (4) P2,(5)P1,(25)P2,(33)P3(45).

Similarly take completion time from above for individual process and subtract it from the Arrival time to get TAT.

Q.20 (c)

By following I and II conditions the process P_1 will get both the resources R_1 and R_2.

If R_1 and R_2 are allocated to the Process P_1, then it will compete its job and release it. After that process P_2 will get both the resource and complete its Job

Q.21 (c)

Fork () returns 0 in child process and process ID of child process in parent process. In Child (x),a=a+5 In Parent (u), a=a–5; Therefore x=u+10. The physical addresses of 'a' in parent and child must be different. But our program accesses virtual addresses (assuming we are running on an OS that uses virtual memory). The child process gets an exact copy of parent process and virtual address of 'a' doesn't change in child process. Therefore, we get same addresses in both parent and child.

Q.22 (b)

Context switches are required only at the time during the switching over to some other process. Let the three given processes be P_1, P_2 and P_3.

Process	Arrival time	Burst time
P_1	0	10
P_2	2	20
P_3	6	30

The chart for SRTF scheduling algorithm is

P_1	P_2	P_3	
0	10	30	60

So, there are only two context switches required,

i.e. at the time unit 10 context switch from P_1 to P_2 and at the time unit 30 context switch from P_2 to P_3.

Q.23 (b)

If it were a preemptive scheduling then there would have been a transition from running state to Ready state to Ready state. So it is non–preemptive scheduling

Q.24 (b)

Process	Total burst time	I/O time	CPU time	I/O time
P1	10	2	7	1
P2	20	4	14	2
P3	30	6	21	3

The Gantt chart is

Idle	P1	P2	P3	Idle	
0	2	9	23	44	47

Total time spent = 47

Idle time = 2+3=5

Percentage of idle time = (5/47)*100=10.6%

Q.25 (a)

Gantt chart for LRTF CPU scheduling algorithm is

P0	P1	P2	P2	P2	P2	P2	P1	P2	P1	P2	P0	P1	P2

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14

Turn around time

P_0 =12-0=12

P_1 =13-1=12

P_2 =14-2=12

Average $= \frac{36}{3} = 12 \sim 13$

Q.26 (b)

Since both p and q don't need additional resources, they both can finish and release Xp + Xq resources without asking for any additional resource. If the resources released by p and q are sufficient for another process waiting for Yk resources, then system is not approaching deadlock.

Q.27 (a)

The P and V functions are given from that we determine that fetch and set instructions always set the memory location x to 1 and fetches

the old value of x into y. S takes only 2 values either 0 or 1 when 5 is initialized to 0, statement 3 will start at location x with this value and fetch-set instruction change the value of x from 0 to 1 and y becomes 0. If more than 2 processes were there and context switching was disabled in P then this won't work properly.

Q.28 (b)
From the given implementation it is determined that statement (b) is correct due to line 3 and 7 in the implementation given.

Q.29 (b)
To rectify the problem, statement (b) is true and is required Qution to the problem.

Q.30 (d)
Wants2 enters the critical section, if process P_1's variable wants1 is true and if wants2 is true then wants1 enters critical section. In both cases, there will be deadlock but no mutual exclusion.

Q.31 (a)
The correct matching is as shown

	List I		List II
P.	Gang Scheduling	1	Thread Scheduling
Q.	Rate Monotonic Scheduling	2	Real-time Scheduling
R.	Fair share Scheduling	3	Guaranteed scheduling

Q.32 (d)
Blocking one kernel level threads does not block all related threads. In kernel level threads, in blocking system call, it can happen with the kernel that another thread is scheduled while kernel is executing.

Q.33 (b)
Gantt chart for SRT is waiting time of process P_2 is given as
(20-15) + (40-50), i.e. 15

Q.34 (c)
From the given matrix we found that the number of available resources of X, Y and Z types are 0, 1, and 2 respectively. Therefore, P_1 will avail these resources first and after release the resources available will be (2, 1, 3) which will be used by P_0 as its request is of 1, 0, 3 resources. After P_0 release the resources, the availability becomes (3, 3, 4) and P_2 will use these resources then as per its requirements. Thus, P_2 finishes in the last.

Q.35 (a)
Deadlock prevention does not guarantee the state that the state would be safe when the request for resources is granted instead it is guaranteed by deadlock avoidance and there will be no deadlock at that time.

Q.36 (c)
When time quantum used in round robin scheduling is more than maximum time required to execute any process then its behave like first come first serve.

Q.37 (a)
In both Synchronous and Asynchronous, an interrupt is generated on completion of I/O. In

Synchronous, interrupt is generated to wake up the process waiting for I/O. In Asynchronous, interrupt is generated to inform the process that the I/O is complete and it can process the data from the I/O operation.

Q.38 (b)

Fork ( ) system call creates the child process initially number of processes is 0. After first fork ( ), it creates a single process. After second fork ( ), it creates one parent and two child processes. After n+1 fork (), the total number of process is 2n but we subtract the main process then total number of child processes is $2^n - 1$

Q.39 (c)

From the given code we get that P (S) and V(S), decrement and increment the value of semaphore respectively. So, from the given conditions we conclude that to avoid mutual exclusion in the value of X_b should be 1 and that of Y_b should be 0.

Q.40 (c)

A processor using pre-emptive scheduling keeps the CPU and releases it when it is switching to waiting state which is indicated by transition C and also a process in blocked state can make transition E while other process is in running state.

Q.41 (a)

The given program initializes the memory location X to 0 in the test and set instruction and in the function leaves CS( ).

Q.42 (a)

Available resources at the start are

R_1	R_2	R_3	R_4
3	2	3	2

Here, as per given table, we need to construct the working of each process.

Time frame is t = 0 to t=10

At t = 0,

	Process	Allocate				Available			
		R_1	R_2	R_3	R_4	R_1	R_2	R_3	R_4
P_1: requests 2 unit of R_2	P_1	0	2	0	0	3	0	1	1
P_2: requests 2 unit of R_3	P_2	0	0	2	0				
P_3: requests 1 unit of R_4	P_3	0	0	0	1				

At t =1

	Process	Allocate				Available			
P_1: requests 1 unit of R_3		R_1	R_2	R_3	R_4	R_1	R_2	R_3	R_4
	P_1	0	2	1	0	3	0	0	1
	P_2	0	0	2	0				
	P_3	0	0	0	1				

At t = 3,

	Process	Allocate				Available			
		R_1	R_2	R_3	R_4	R_1	R_2	R_3	R_4
P_1: requests 2 unit of R_1	P_1	1	2	1	0	0	0	0	0
	P_2	1	0	2	1				
	P_3	2	0	0	1				

At t = 4,

	Process	Allocate				Available			
		R1	R2	R3	R4	R1	R2	R3	R4
P_2: requests 1 unit of R_2	P_1								
	P_2								
	P_3								

At t = 5,

	Process	Allocate				Available			
		R_1	R_2	R_3	R_4	R_1	R_2	R_3	R_4
P_1: releases 1 unit of R_2	P_1	1	1	1	0	1	1	0	0
	P_2	1	0	2	1				
P_3: releases 2 unit of R_1	P_3	0	0	0	1				

At t = 6

	Process	Allocate				Available			
		R_1	R_2	R_3	R_4	R_1	R_2	R_3	R4
P_2: requests 1 unit of R_3	P_1	1	1	1	0	1	1	1	0
	P_2	1	0	1	1				
	P_3	0	0	0	1				

At t =7

	Process	Allocate				Available			
		R1	R2	R3	R4	R1	R2	R3	R_4
P_1: releases 1 unit of R_3	P_1	1	1	0	0	1	0	2	0
	P_2	1	0	1	1				
	P_3	0	1	0	1				

At t = 8,

	Process	Allocate				Available			
		R_1	R_2	R_3	R_4	R_1	R_2	R_3	R_4
P_2: FINISH	P_1	1	1	0	0	2	0	3	1
	P_3	0	1	0	1				
P_1: requests 2 unit of R_4		1	1	0	0	2	0	2	1
P_3: requests 1 unit of R_3		0	1	1	1				

At t = 9,

	Process	Allocate				Available			
		R1	R2	R3	R4	R1	R2	R3	R4
P_3: FINISH	P_1	1	1	0	0	2	1	3	2
	P_1	1	1	0	2	2	1	3	0

At t = 10,

	Process	Allocate				Available			
		R1	R2	R3	R4	R1	R2	R_3	R_4
P_3: FINISH						3	2	3	2

This way there is no deadlock occurred here and all the processes are finished.

Q.43 (b)

Considering the options and their outcome, deadlock occurred in option (b):
P_0 requires $R_0 n R_2$,
P_1 requires $R_{20} n R_{18}$,
going on in this series—P_8 will require $R_8 n R_{10}$,
P_9 will require$R_{12} n R_{10}$,
P_{10} will require $R_{10} n R_{12}$ and
P_{11} will require $R_{10} n R_8$.
R_{10} is common requirement for last 4 processes at a time and thus, it will lead to deadlock.
Since, last 4 processes and demanding the R_{10}, deadlock is unavoidable.

Q.44 (a)

As per the methods provided in process 1 and process 2, the process 1 and process 2 cannot exist simultaneously in the critical section. So, when the process A is in the critical section, process B is not and thus, the condition of mutual exclusion is satisfied but not progress.

Q.45 (a)

P_0 uses a semaphore. Hence, anything whose value is 1, P_0 will print 'O'. After the release of S_1 and S_2, we observe that either P_1 or P_2 will release P_0
so that it prints '0' at least 2 times.
Therefore, it is concluded that neither P_1 nor P_2 will go, it is the P_0 that prints '0' 2 times at least.

Q.46 (d)

Statement I True: The shortest job selects the waiting process with the smallest execution time to execute next. Shortest job next maximizes process throughput at the number of processes completed in a given time is the maximum. Thus, the average waiting time is for a process is minimized. The disadvantage here is that it also permits process starvation for the processes that take long time to complete in a condition that short processes are continually added to the queue.
Statement II True: Pre-emptive algorithms are driven by the prioritized computation. The process with the highest priority is the one that is currently using the processor. If a process is currently using the processor and a new process with a higher priority enters, the process on the processor should be removed and returned to the ready list until its priority becomes the highest-priority process.
Statement III True: Round robin has a better response time than FCFS. Round robin calls for the distribution of the processing time equitably among all processes requesting the processor. Run process for one time slice, and then move to back of queue. Each process gets equal share of the CPU. Most systems use some variant of this. All statements are true.

Q.47 (c)

Process switching involves mode switch. Context switching can occur only in kernel mode. $t_1 < t2$

Q.48 (d)

As if CPU temperature is high, the computer immediately shut down.

Q.49 (c)

Threads share address space of Process. Virtually memory is concerned with processes not with Threads.

Q.50 (a)

Waiting time of P_0 =5-1 = 4
Waiting time of P_1 =1-1=0
Waiting time of P_2 =13-2=11
Average waiting time $= \frac{4+0+11}{3} = \frac{15}{3} = 5$

Q.51 (c)

A process executes the code and works as follows:
The number of child processes created $= 2^n - 1 = 2^3 - 1$
= 8 – 1 = 7
Where, n=3∵ 3 fork ( ) are executing.

Q.52 (c)

Process	Arrival time	Time units required
P_1	0	5
P_2	1	7
P_3	3	4

FCFS:
0 5 12 16
Order $P_1 \rightarrow P_2 \rightarrow P_3$
Round robin here, not P_2 because in the ready queue P_2 comes later than P_3.

P1	P2	P1	P3	P2	P1	P3	P2	P2

0 2 4 6 8 10 11 13 15 16
Here, not P_3 because in the ready queue comes first than P_3.
Order $P_1 \rightarrow P_3 \rightarrow P_2$

Q.53 (b)

```
Acquire Lock (L)
Initial value of L = 0
{
        While (Fetch_And_add (L, 1))
        {
            L = 1;
        }
}
Release Lock (L)
{
L = 0;
}
```

When P_1 process executes Acquire lock, after executing it, value of L= 1;
[∵ P_1 Fails the while loop condition because Fetch_And_Add instruction return 0 and it sets the value of L=1]
Now, there is a context switch and still realase lock (L) produce hasn't been executed.
Now, another process P_2 executes the acquire lock (L), it will execute the while loop indefinite period of time.
Now, suppose at this point i.e.,

```
While (Fetch_And_Add (L, 1))
        {
→ L = 1;
        }
```

Process P_2 context switch occurs.
Now, control again comes to P_1 it will execute realease lock procedure and the value of L becomes 0.
Now, lock is free.
Now, after this suppose again context switch occurs, controls return to P_2. P_2 will again set the value of L=l Now, P_2 will again executing the while loop indefinite amount of time.

At this instance lock is free; even then P2 can't get it because value of lock is not zero.
Any other-process will also blocks now in the infinite while loop, even the lock is free.
This implementation fails as L can take on a non-zero value when the lock is actually available.

Q.54 (b)

Three concurrent processes X, Y and Z execute three different code segments that access and update certain shared variables.

Processes

X	Y	Z
P(a)	P(b)	P(c)
P(b)	P(c)	P(d)
P(c)	P(d)	P(a)

(a) X : P(a) P(b) P(c)
Y : P(b) P(c) P(d)
Z : P(c) P (d) P (a)
Suppose X first executes P(a) and P(b) and then switches to process Z. where P(c) and P(d) are executed and wait for P(a). Then again process switches to X and then wait for P(c).
∴ Process X is waiting for C which is occupied by Z and Z is waiting for a is occupied by process X.
So neither can execute and deadlock occurs. (Not Acceptable)
(b) X : P(b) P(a) P(c)
Y : P(b) P(c) p(d)
Z: P (a) P(c) P (d)

X	Y	Z
P(b)	P(b)	P(a)
P(a)	P(c)	P(c)
P(c)	P(d)	P(d)

Execution can be carried out in a proper way without deadlock occurrence and no wait for any variable in the processer. (Acceptable)
(c) X : P(b) P(a) P(c)
Y : P(c) P(b) P(d)
Z: P (a) P(c) P (d)

X	Y	Z
P(b)	P(c)	P(a)
P(a)	P(b)	P(c)
P(c)	P(d)	P(d)

The sequence of variable P{b) and P(c) are reverse and opposite [i.e., P(b) P(c) P(a) and P(a) P(b) P(c)] So, deadlock may occurs in X and Y respectively (Not Acceptable)
(d) X : P(a) P(b) P(c)
Y : P(c) P(b) P(d)
Z: P(c) P (d) P (a)

X	Y	Z
P(a)	P(c)	P(c)
P(b)	P(b)	P(d)
P(c)	P(d)	P(a)

The sequence of variable P(c) and P (a) are opposite in Z and X. So, deadlock may occur (Not Acceptable) Hence, the answer is (b).

Q.55 (b)

The scheduling algorithm works as round robin with quantum time equals to T. After a process's turn comes and it has executed for T units, its waiting time becomes least and its turn comes again after every other process has got the token for T units.

Q.56 (d)

Each of the processes W and X reads x from memory and increment x by 1. Each of the processes Y and Z reads x from c. Memory, and decrement by 2.

1. Start with X and perform P(S) then S = 1 read X = 0 x = X + 1 = 1
2. Then Y will perform P(S) then S = 0 read X - 0 x = x -2 = -2 then store x. V(S), S =1
3. Then Z will perform P(S) then S = 0, read x = -2x = x- 2 = -4 then store X, V(S).S = 1.
4. Then x will store x. V(S), S=2, X= 1
5. Then W will perform P(S), S =1, read x = 1
   x = x + 1 = 2, store x, V(S), S =2, x = 2

Hence, answer is option (d).

Q.57 (c)

Options (a) suppose process X executes Exit X then it will wait for R, and then process Y executes Entry Y then it will wait for S. Since, initially both binary semaphores are 0, no one will increment it and both processes will be stuck up in a deadlock.

Option (a) is incorrect.

Option (b) Here if process X executes for n times repeatedly it will set both semaphores to 1 (since only two values are possible) and after that process y executes. First time it passes the Entry Y and makes both semaphores to 0.

And on second time it finds both semaphores to 0 and cannot pass the Entry Y barrier. Hence it will get stuck.

So, option b is wrong.

Option (c) Considering any sequence of operation of process X and process Y, first process X will wait for S which is increment by process Y and then . Process Y waits for R which is incremented by process X. There is no sequence of operation in which the value of R or S overlaps. Both processes execute one after another.

So, option (c) is correct.

Option (d) suppose first process X executes it sets R = 1 and then waits for S. Now after that process Y executes. It first sets S = 1 and. decrement R = 0. It comes again and. then again sets S = 1(/'.e., it overlaps the value of S) and then again wait for- R.. Clearly hence one iteration of process X is lost due to overlapping of value of S, and after n -1 iteration process X will be stuck.

So option (d) is wrong.

Q.58 (c)

(A)The producer will be able to add an item to the buffer, but the consumer can never consume if given statement is false, because once producer produces an item and places in buffer, the next turn to execute can be given to consumer ( ). [The value of s = 1 and n=1]. So consumer will be definitely able to consume it by performing successful down operations on s and n.

(B)The consumer will remove no more than one item from the buffer.

Given statement is false as if p( ) produces and adds to buffer, (c) will consume the added item, this sequence of alteration (p and c) will always make consumer to remove items from buffer.

This statement would have been true if it was said that the consumer will remove no more than one item from the buffer one after the other. (at a time).

(C)Dead lock occurs if the consumer succeeds in acquiring semaphore's' when the buffer is empty. Given statement is true as when buffer is empty initially if consumer gets the turn to execute.
(D)Even if the starting value for the semaphore 'n' becomes 1, there will be invalid execution of consumer on buffer empty condition, which should not happen. So statement is false.

Q.59 (d)
User threads are supported above the kernel and a managed without kernel support. The thread function library to implement user level threads usually runs on top of the system in user mode. Thus these threads with in a process are invisible to the operating system. Since the kernel is unaware of the existence of such threads; when one user level thread is blocked in the kernel all other threads of its process are blocked. So options (a) and (b) are true (c) The OS is aware of kernel level threads. Kernel threads are scheduled by the OS's scheduling algorithms and require a "lightweight" context switch to switch between (that is, registers, PC and SP must be changed, but the memory context remains the same among kernel threads in the (same process). User level threads are much faster to switch between as there is not context switch (d) False Kernel level threads within the same process share code section, data section and other operating system resources such as open files and signals.

Q.60 (7.2)

P1	P2	P3

A	B	A	C	E	D

0 3 5 8 12 15 21
Average Turn Around time
=(8-0)+(5-3)+(12-5)+(21-7)+(15-10)/5
= 36/5 = 7.2ms

Q.61 (1000) There are three processes A, B and C that run in round robin manner with time slice of 50 ms.
Processes start at 0, 5 & 10 miliseconds.
The processes are executed in below order A, B, C, A
50 + 50 + 50 + 50 (200 ms passed)
Now A has completed 100 ms of computations and goes for I/O now
B, C, B, C, B, C
50+50+50+50+50+50(300ms passed)
C goes for i/ o at 500ms and it needs 500ms to finish the IO.
So C would complete its first IO at 1000 ms

Q.62 (5.5)
The Gantt chart for SRTF scheduling algorithm is as follows:

P_1	P_2	P_3	P_4	P_1

0 2 6 12 17 27
Average waiting time = (15+0+3+4) / 4 = 22/4 = 5.5

Q.63 (b)
REQ1
Once P_0 is allocated with (0, 0, 2), the status of the system will be as follows

Allocation	Max	Need	Available
X Y Z	X Y Z	X Y Z	X Y Z
0 0 3	8 4 3	8 4 0	3 2 0
3 2 0	6 2 0	3 0 0	
2 1 1	3 3 3	1 2 2	

With available (3, 2, 0) only P_1 can be served. Once P_1 is executed, available will be (6, 4, 0), with (6, 4, 0) we can't serve either P0 or P2. Hence there is no safe sequence. Hence REQ1 can't be permitted.

REQ2

Once 1 P is allocated with (2, 0, 0), the status of the system will be as follows

Allocation	Max	Need	Available
X Y Z	X Y Z	X Y Z	X Y Z
0 0 1	8 4 3	8 4 0	1 2 2
5 2 0	6 2 0	1 0 0	
2 1 1	3 3 3	1 2 2	

With available (1, 2, 2), we can serve either P1 or P2.

If we serve P1 then the safe sequence is P1, P2, P0. If we serve P2 then the safe sequence is P2, P1, P0. As true is at least one safe sequence we can permit REQ2.

Q.64 (7)

If 6 resources are there then it may be possible that all three process have 2 resources and waiting for 1 more resource. Therefore, all of them will have to wait indefinitely.

If 7 resources are there, then atleast one must have 3 resources so deadlock can never occur.

Q.65 (4)

It seems to be wrong question in GATE exam. Below seems to be the worst case situation. In below situation P1 and P2 can continue as they have 2 resources each

P1 → 2
P2 → 2
P3 → 1
P4 → 1

Q.66 (d)

If Ist is followed, then hold and wait will never happen. II, III and IV are similar. If any of these is followed, cyclic wait will not be possible.

Q.67 (12)

Periods of T1, T2 and T3 are 3ms, 7ms and 20ms

Since priority is inverse of period, T1 is the highest priority task, then T2 and finally T3

Every instance of T1 requires 1ms, that of T2 requires 2ms and that of T3 requires 4ms

Initially all T1, T2 and T3 are ready to get processor, T1 is preferred

Second instances of T1, T2, and T3 shall arrive at 3, 7, and 20 respectively.

Third instance of T1, T2 and T3 shall arrive at 6, 14, and 49 respectively.

Time-Interval Tasks
0-1 T1
1-2 T2
2-3 T2
3-4 T1
[Second Instance of T1 arrives]
4-5 T3
5-6 T3
6-7 T1
[Third Instance of T1 arrives]

[Therefore T3 is preempted]
7-8 T2
[Second instance of T2 arrives]
8-9 T2

9-10 T1
[Fourth Instance of T1 arrives]
10-11 T3
11-12 T3
[First Instance of T3 completed]

Q.68 (d)

The size of ready queue doesn't depend on number of processes. A single processor system may have a large number of processes waiting in ready queue.

Q.69 (c)

Turnaround time is the total time taken between the submission of a program/process/thread/task (Linux) for execution and the return of the complete output to the customer/user.
Turnaround Time = Completion Time - Arrival Time.
FCFS = First Come First Serve (A, B, C, D); Average Turnaround time= 7.25
SJF = Non-preemptive Shortest Job First (A, B, D, C); Average Turnaround time= 6.75
SRT=Shortest Remaining Time (A(3), B(1),C(4),D(2),B(5));
Average Turnaround time= 6.25
RR =
Round Robin with Quantum value 2 (A(2),B(2),A(1),C(2),B(2),D(2),C(2), B(2)
Average Turnaround time= 8.25

Q.70 (a)

When both processes try to enter critical section simultaneously, both are allowed to do so since both shared variables varP and varQ are true. So, clearly there is NO mutual exclusion. Also, deadlock is prevented because mutual exclusion is one of the four conditions to be satisfied for deadlock to happen. Hence, answer is A.

Q.71 (a)

It satisfies the mutual exclusion, since only one process can be in the critical section at any time.

Q.72 (c)

The given Qution for two process synchronization using "turn" variable satisfies the only mutual exclusion and bounded waiting but progress is violated.
Consider that **Process"0"** executed its critical section and made turn=**1** and **Process"0"** wants to execute the critical section for the 2nd time,
Process"0" has to wait indefinitely for **Process"1"** to start, which will make **turn=0**.
Now we can say that Progress condition is failed, since Critical Section is free and Process "0" wants to enter Critical Section and it can enter only if Process "1" turns up.

Q.73 (7)

S = -20 + 12 = -8
If initial value of S = 8 then 20 P(S) operations will succeed.
As want to block at least one P(S) operation by keeping "S" the largest initial value.
If we keep initial value of S=7, then 19 P(S) operations will succeed and only 20th P(S) operation will get blocked.
The largest initial value of S for which at least one P(S) operation remains blocked is 7.

Q.74 (a)

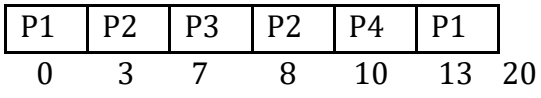
Facts about scheduling algorithms is

1) FCFS can cause long waiting times, especially when the first job takes too much CPU time.
2) Both SJF and Shortest Remaining time first algorithms may cause starvation. Consider a situation when long process is there in ready queue and shorter processes keep coming.
3) If time quantum for Round Robin scheduling is very large, then it behaves same as FCFS scheduling.
4) SJF is optimal in terms of average waiting time for a given set of processes. SJF gives minimum average waiting time, but problems with SJF is how to know/predict time of next job.

Q.75 (8.25)

Pre-emptive Shortest Remaining time first scheduling, i.e. that processes will be scheduled on the CPU which will be having least remaining burst time (required time at the CPU).

The processes are scheduled and executed as given in the below Gantt chart.

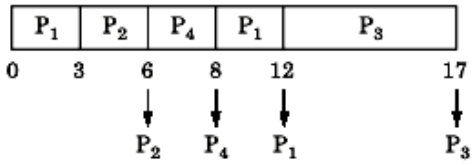


Q.76 (d)

Generally every thread of a process have their own PC and stack. Both heap and global variables are shared by every thread of a process

Q.77 (3.0)

Using preemptive SRTF algorithm Gantt chart will be,



Turnaround time Waiting time

$P_1 \rightarrow 12-0=12$

$P_2 \rightarrow 6-3=3$

$P_3 \rightarrow 17-5=12$

$P_4 \rightarrow 8-6=2$

$P_1 \rightarrow 12-7=5$

$P_2 \rightarrow 3-3=0$

$P_3 \rightarrow 12-5=7$

$P_4 \rightarrow 2-2=0$

Average Waiting Time

$$=\frac{5+0+7+0}{4}=3.0$$

Q.78 (c)

There are 'x' number of threads and 'y' number of locks

Now, considering each option,

- If there is 1 threads and two locks then there won't be situation of deadlock.
- Similarly if there is only 1 lock , there will be no deadlock situation
- But when there are two processes and two threads, deadlock situation can occur because both the process can acquire one lock and may demand another lock which arises deadlock.

Q.79 (b)

All the threads share address space but other entities like, stack, PC resisters are not shared and every thread will have its own.

Q.80 (29)

Process	Arrival Time	Burst Time	Priority	C.T.	A.A.T.	W.T.
P_1	0	11	2	49	49	38
P_2	5	28	0(high)	33	28	0
P_3	12	2	3	51	39	37
P_4	2	10	1	40	38	28
P_5	9	16	4	67	58	42
					Total	145

P_1	P_4	P_2	P_4	P_1	P_3	P_5

0 2 5 33 40 43 51 67

Average waiting time $= \frac{145}{5} = 29$

Q.81 (b)

Process	Current Allocation	Max. Requirement	Remaining need	Current available
P1	3	7	4	9-7=2
P2	1	6	5	5
P3	3	5	2	8
				9

Safe sequence ⇒ P3 →P1→P2
Safe and not deadlocked

Q.82 2

P_1 P_2 P_3
| | |
| No deadlock

Maximum each process can request for 2 resources so, that there will not be any deadlock, because we have only 4 resource available.
So, K value is '2'.

Q.83 (a)

	Max need			Current allocation			Current available			Remaining need		
	E	F	G	E	F	G	E(3)	F(3)	G(0)	E	F	G
P_0	4	3	1	1	0	1	4	3	1	3	3	0
P_1	2	1	4	1	1	2	5	3	4	1	0	2
P_2	1	3	3	1	0	3	6	4	6	0	3	0
P_3	5	4	1	2	0	0	8	4	6	3	4	1

Safe sequence: P_0, P_2, P_1, P_3
Safe state

2 MEMORY MANAGEMENT

2.1 WHAT IS MEMORY MANAGEMENT AND WHY WE DO IT?

CPU is shared between processes.
CPU Utilization increases.
For the same reason, all process must be in RAM. To handle all process in RAM, we do memory management. Memory management requires hardware support.
CPU reads instructions from memory locations as guided by Program Counter.

2.1.1 Address Binding:

i. Hard Disk contains .exe files or executable programs.
ii. These are moved to main memory or RAM and becomes processes. A process can be executed by CPU.
iii. Input Queue: Collection of programs that are waiting on disk to be brought in to memory for execution forms the input queue.

Binding of instructions and data to memory addresses in main memory for a process can be done in any of the following fashions:

i. **Compile Time Address Binding: Absolute Code**
If it is known at compile time where the process will reside in memory, then absolute code can be generated. For example, if it is known a priori that a user process resides starting at location R, then the generated compiler code will start at that location and extend up from there. If at some later time, the sterling location changes, then it will be necessary to recompile this code. The MS-DOS. COM-format programs are absolute code bound at compile time.

ii. **Load Time Address Binding : Relocatable Code**
If it is not known at compile time where the process will reside in memory, then the compiler must generate re-locatable code. In this case, final binding is delayed until load time, if the starting addresses changes; we need only to reload the user code to incorporate this changed value.

iii. **Execution Time Address Binding:**
If the process can be moved during its execution from one memory segment to another, then binding must be delayed until run time. Special hardware must be available for this scheme to work.

2.1.2 Logical Versus Physical Address Space

An address generated by the CPU is commonly referred to as a logical address, whereas an address seen by the memory unit (that is, the one loaded into the memory address register of the memory) is commonly referred to as a physical address. The run-time mapping from virtual to physical address is done by the memory-management unit (MMU), which is a hardware device. As shown in Fig. , this scheme requires hardware support slightly different from the hardware configuration. The base register is now

called a relocation register. The value in the relocation register is added to every address generated by a user process at the time it is sent to memory. For example, if the base is at 14,000, then an attempt by user to address location 0 is dynamically relocated to location 14,000; an access to location 346 is mapped to location 14,346.

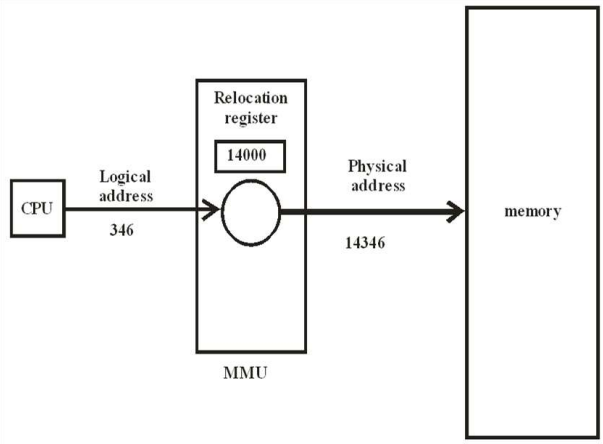


Fig. Dynamic relocation using relocation register.

We have two different types of addresses: Logical addresses (in the range 0 to max) and physical addresses (in the range (R + 0) to (R + max) for a base value R). The user generates only logical addresses and thinks that the process runs in locations 0 to max. The user program supplies logical addresses; these logical addresses must be mapped to physical addresses before they are used.

2.1.3 Swapping

A process needs to be in the memory to be executed. A process, however, can be swapped temporarily out of memory to a backing store, and then brought back into memory for continued execution. For example, assume a multiprogramming environment with round-robin CPU-scheduling algorithm. When a quantum expires, the memory manager will start to swap out the process that just finished, and to swap in another process to the memory space that has been freed.

Swapping requires a backing store. The backing store is commonly a fast disk. It must be large enough to accommodate copies of all memory images for all users, and must provide direct access to these memory images.

The context-switch time in such a swapping system is fairly high. To get an idea of the context-switch time, assume that the user process is of size 100 K and the backing store is a standard hard disk with a transfer rate of 1 megabyte per second. The actual transfer of the 100 K process to or from memory takes 100K/1000K per second = 1/10 second = 100 milliseconds.

Assuming that no head seeks are necessary and an average latency of 8 milliseconds, the swap time takes 108 milliseconds. Since we must both swap out and swap in, the total swap time is then about 216 milliseconds. For efficient CPU utilization, we want our execution time for each process to be long relative to the swap time. Thus, in a round-robin CPU-scheduling algorithm, for example, the time quantum should be substantially larger than 0.216 seconds.

There are other constraints on swapping. If we want to swap a process, we must be sure that it is completely idle. Of particular concern is any pending I/O. If a process is waiting for an I/O operation, we may want to swap that process to free up its memory. However, if the I/O is asynchronously accessing the user memory for I/O buffers, then the process cannot be swapped. Assume that the I/O operation was queued because the device was busy. Then, if we were to swap out process P_1 and swap in process P_2, the I/O operation might then attempt to use memory that now belongs to process P_2. The two main solutions to this problem are

i) Never to swap process with pending I/O.

ii) To execute I/O operations only into operating-system buffers. Transfers between operating system and process memory then occur only when the process is swapped in.

In compile time/load time binding, a swapped out process will be swapped in to the same memory space.

In execution time binding, different memory space can be used because physical addresses are computed during execution time.

2.1.4 Contiguous Allocation

The main memory must accommodate both the operating system and the various user processes. The memory is usually divided into two partitions.
One for the resident operating system and one for the user processes. It is

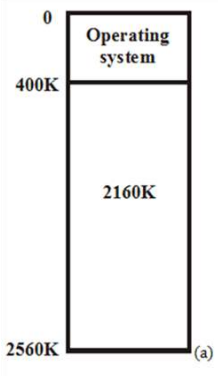
Operating System
User
512K

2.2 SINGLE-PARTITION ALLOCATION

If the operating system is residing in lower section of memory and the user processes are executing in higher sections of memory, it is required to protect the operating-system code and data from changes (accidental or malicious) by the user processes. It is also required to protect the user processes from one another. This protection can be provided by using a relocation register.
The operating system keeps a table indicating which parts of memory are available and which are occupied. Initially, all memory is available for user processes, and is considered as one large block of available memory, a hole. When a process arrives and needs memory, then search for a hole large enough for this process. If we find one, we allocate only as much memory as is needed, keeping the rest available to satisfy future requests.

2.2.1 Multiple-Partition Allocation

Since it is desirable, that there be several user processes residing in memory at the same time, we need to consider the problem of how to allocate available memory to the various processes that are in the input queue waiting to be brought into memory. One of the simplest schemes for memory allocation is to divide memory into a number of fixed-sized partitions.
For example assume that we have 2560K of memory available and a resident operating system of 400K. This situation leaves 2160 K for user processes, as shown in Figure below.
Given the input queue in the figure, and FCFS job scheduling, we can immediately allocate memory to processes P_1, P_2, and P_3 creating the memory map of Figure (a). We have a hole of size 260K that cannot be used by any of the remaining processes in the input queue. Using a round-robin CPU-scheduling with a quantum of 1 time unit, process P_2,



Job Queue

Process	Memory	Time
P1	600K	10
P2	1000K	5
P3	300K	20
P4	700K	8
P5	500K	15

(b)

Fig. Scheduling example

will terminate at time 14, releasing its memory. This situation is shown Figure (b). We then return to our job queue and schedule the next process, process P_4, to produce the memory map of Figure (c). Process P_1 will terminate at time 28 to produce Figure (d); process P_5 is then scheduled, producing Figure (e). This example shows the general structure of the allocation scheme

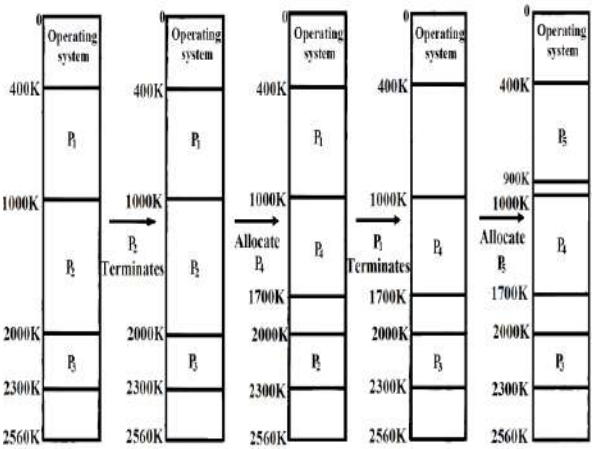


fig. Memory allocation and long-term scheduling

In general, throughout memory when a process arrives and needs memory, we search this set for a hole that is large enough for this process. The set of holes is searched to determine which hole is best to allocate. First-fit and worst-fit are the most common strategies used to select a free hole from the set of available holes.

i. **First-fit**
   Allocate the first hole that is big enough. Searching can start either at the beginning of the set of holes or where the previous first-fit search ended. We can stop searching as soon as we find a free hole that is large enough.

ii. **Best-fit**
   Allocate the smallest hole that is big enough. We must search the entire list, unless the list is kept ordered by size. This strategy produces the smallest leftover hole.

iii. **Worst – fit**
   Allocate the largest hole. Again, we must search the entire list, unless it is than the smaller leftover hole from a best-fit approach.

2.3 EXTERNAL & INTERNAL FRAGMENTATION

As processes are loaded and removed from memory, the free memory space is broken into little pieces. External fragmentation exists when enough total memory space exists to satisfy a request, but it is not contiguous; storage is fragmented into a large number of small holes. This fragmentation problem can be severe. In the worst cast we could have a block of free (wasted) memory between every two processes. If all this memory were in one big free block, we might be able to run several more processes. One solution to the problem of external fragmentation is compaction. The goal is to shuffle the memory contents to place all free memory together in one large block. Resolution is done at assembly or load time, compaction cannot be done; compaction is possible only if relocation is dynamic, and is done at execution time.

Swapping can also be combined with compaction. A process can be rolled out of main memory to a backing store and rolled in again later. When the process is rolled out, its memory is released, and perhaps is reused for another process. If static relocation is used, the process must be rolled into the exact same memory locations that it occupied previously. This restriction may require that other processes be rolled out to free that memory.

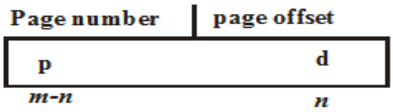
If dynamic relocation (such as with base and limit registers) is used, then a process can be rolled into a different location. In this case, we find a free block, compacting if necessary, and roll in the process.

2.3.1 Paging

Another possible solution to the external fragmentation problem is to permit the logical address space of a process to be noncontiguous, thus allowing a process to be allocated physical memory wherever the latter is available. One way of implementing is through the use of paging scheme.

2.3.2 Basic Method

Physical memory is broken into fixed-sized blocks called frames. Logical memory is also broken into blocks of the same size called pages. When a process is to be executed, its pages are loaded into any available memory frames from the backing store. The backing store is divided into fixed-sized block that are of the same size as the memory frames.
The hardware support for paging is shown in Figure 13.5. Every address generated by the CPU is divided into two parts;
A page number (p) and a page offset (d).
The page number is used as an index into a page table. The page table contains the base address of each page in physical memory. This base address is combined with the page offset to define the physical memory address that is sent to the memory unit). If the size of logical address space is 2^m, and a page size is 2^n addressing units (bytes or words), then the high-order m-n bits of a logical address designate the page number, and the n low-order bits designate the page offset. Thus, the logical address is as follows:



Where p is an index into the page table and d is the displacement within the page.
Consider the memory of Figure. Using a page size of 4 bytes and a physical memory of 32 byte (8 pages), we show an example of how the user's view of memory can be mapped into physical memory. Logical address 0 is page 0, offset 0. Indexing into the page table, we find that page 0 is in frame 5. Thus, logical address 0 maps to physical address 20 (= (5 x 4) + 0).
Logical address 3 (page 0, offset 3) maps to physical address 23 (= (5 x 4) + 3).
Logical address 4 is page 1, offset 0; according to the page table, page 1 is mapped to frame 6.
Thus, logical address 4 maps to physical address 24 (= (6 x 4) + 0).
Logical address 13 maps to physical address 9.
When we use a paging scheme, we have no external fragmentation: Any free frame can be allocated to a process that needs it. However, we may have some internal fragmentation.

Logical Memory

0	a
1	b
2	c
3	d
4	e
5	f
6	g
7	h
8	i
9	j
10	k
11	l
12	m

Page table

0	5
1	6
2	1
3	2

Physical memory

0	
4	i
	j
	k
	l
8	m
	n
	o
	p
12	
	16
20	a
	b

13	n
14	o
15	p

	c d
24	e f g h
28	

Fig. Paging example for a 32-byte memory with 4-byte pages

Since the operating system is managing physical memory, it must be aware of the allocation details of physical memory:
Which frames are allocated, which frames are available, how many total frames there are and so on.
This information is generally kept in a data structure called a frame table. The frame table has one entry for each physical page frame, indicating whether the latter is free or allocated and, if is allocated, to which page of which process or processes.

2.3.3 Protection

Memory protection in a paged environment is accomplished by protection bits that are associated with each frame. Normally, these bits are kept in the page table. One bit can define a page to be read and write or read-only.
A valid-invalid bit. When this bit is set to "valid", this value indicates that the associated page is in process's logical address space, and is thus a legal (valid) page.

2.3.3.1 Multilevel Paging

Most modern computer system supports a very large logical address space (2^{32} to 2^{64}). In such an environment the page table itself becomes excessively large.
For example, consider a system with a 32-bit logical address space. If the page size in such a system is 4K bytes (2^{12}), then a page table may consist of up to 1 million entries ($2^{32} / 2^{12}$) Because each entry consists of 4 bytes, each process may need up to 4 megabytes of physical address space for the page table alone. Clearly, we would not want to allocate the page table contiguously in main memory. One simple solution to this is to divide that page table into smaller pieces.

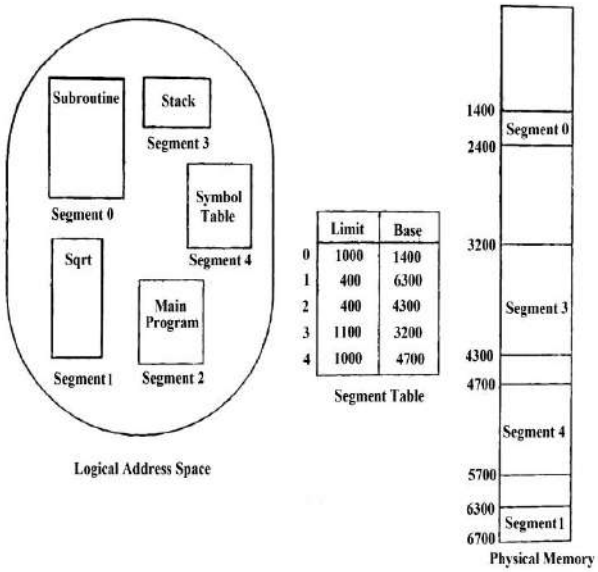
2.4 SEGMENATION

An important aspect of memory management that became unavoidable with paging is the separation of the user's view of memory and the actual physical memory. Consider how you of a program when you are writing it. You think of it as a main program with a set of subroutines, procedures, or modules. There may also be various data structures: tables, arrays, stacks, various, and so on. Each of these modules or data elements is referred to by name. You talk about "the symbol table," "function sqrt," "the main program," without carrying what addresses in memory these elements occupy. You are not concerned with whether the symbol table is stored before or after the Sqrt function of the signet is of variable length; is intrinsically defined by the purpose of the segment in the program.

Segmentation is a memory-management scheme that supports this user view or memory. A logical address space is a collection of segments. Each segment has a name and a length. The address space is a collection of segments. Each segment has a name and a length. The addresses specify both the segment name and the offset within the segment. The user therefore specifies each address by the offset the segment.

For simplicity of implementation, segments are numbered and are referred to by a segment number, rather than by a segment name. Thus, a logical address consists of a two tuple:< segment-number, offset >.

Although the user can now refer to objects in the program by a two-dimensional address, the actual physical memory is still, of course, a one-dimensional sequence of bytes. Thus, we must define an implementation to map two-dimensional user-defined addresses into one-dimensional physical addresses. This mapping is affected by a segment table. Each entry of the segment table has a segment base and a segment limit. The segment base contains the starting physical address where the segment resides in memory, whereas the segment limit specifies the length of the segment. Consider the situation shown in the Figure below. We have five segments numbered from 0 through 4. The segments are stored in physical memory as shown. The segment table has a separate entry for each segment, giving the beginning address of the segment in physical memory (the base) and the length of that segment (the limit). For example, segment 2 is 400 bytes long, and begins at location 4300. Thus, a reference to byte 53 of segment 2 is mapped onto location 4300+53=4353. A reference to byte 1222 of segment 0 would result in a trap to the operating system, as this segment is only 1000 bytes long.



2.5. FRAGMENATION

The long-term scheduler must find all allocate memory for all the segments of a user program. This situation is similar to paging expect that the segments are of variable length; pages are all the same size. Thus, as with the variable- sized partition scheme, a memory allocation is a dynamic storage- allocation problem, usually solved with a best-fit or first-fit algorithm. Segmentation may then cause external fragmentation, when all blocks of free memory are too small to accommodate a segment. In this case, the process may simply have to wait until more memory (or at least a larger hole) becomes available, or compaction may be used to create a large hole. Because segmentation is by its nature a dynamic relocation algorithm, we can compact memory whenever we want.

2.6 VIRTUAL MEMORY

The good of various memory-management strategies that have used in computer systems are: to keep many processes in memory simultaneously to allow multiprogramming. However, they

tend to require the entire process to be in memory before the process can execute.

Virtual memory is a technique that allows the execution of process that may not be completely in memory. This technique frees programmers from concern over memory storage limitations.

An examination of real programs shows us that, in many cases, the entire program is not needed. For instance

i. Program often have code to handle unusual error conditions. Since these errors seldom, if ever occur in practice, this code is almost never executed.
ii. Arrays, lists and table are often allocated more memory than they actually need. An array may be declared 100 by 100 elements, even though it is seldom larger than 10 by 10 elements. An assembler symbol table may have room for 3000 symbols, although the average program has less than 200 symbols.

Even in those cases where the entire program is needed, it may not all be needed at the same time (the case with overlays).

The ability to execute a program that is only partially in memory would have many benefits

i) A program would no longer be constrained by the amount of physical memory that is available. Users would be able to write programs for an extremely large virtual address space, simplifying the programming task.
ii) Because each user program could take less physical memory, more programs could be run at the same time, with a corresponding increase in CPU utilization and throughput, but with no increase in response time or turnaround time.
iii) Less I / O would be needed to load or swap user program into memory, so each user program would run faster.

Virtual memory is the separation of user logical memory from physical memory, this separation allows an extremely large virtual memory to be provided for programmers when only a smaller physical memory is available. With this scheme, we need some form of hardware support to distinguish between those pages that are on the disk. The valid-invalid bit scheme described can be used for this purpose. This time, however, when this bit it set to “valid”, this value indicates that the associated page is both legal and in memory. If the bit is set to "involved", this value indicates that the page either is not valid (that is, not in the logical address space of the process), or is valid but is currently on the disk.

Let p be the probability of a page fault ($0 \leq p \leq 1$). We would expect p to be close to zero; that is, there will be only a few page faults. Then

Effective access time = (1 - p) x ma + p x page fault time. In any case, we are faced with three major components of the page-fault service time:

i) Service the page-fault interrupt.
ii) Read in the page.
iii) Restart the process.

If we take an average page-fault service time 25 milliseconds and a memory access time of 100 nanoseconds, then

Effective access time in nanoseconds
= (1-p) × (100) +p (25 milliseconds)
= (1 - p) ×100 + p×25,000,000
= 100 +24,999,900 × p.

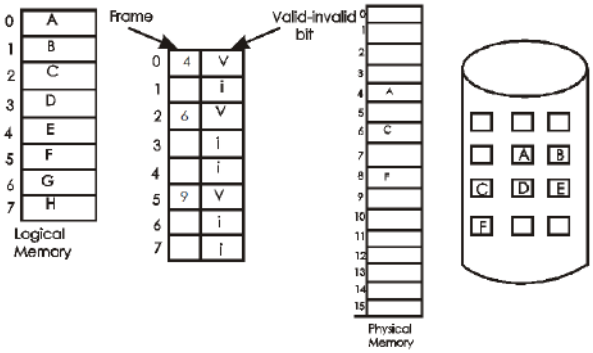
Thus the effective access time is directly proportional to the page-fault rate. In one access out of 1000 causes a page fault, the effective access.

2.7 PAGE REPLACEMENT

If we increase our degree of multiprogramming, we are over-allocating memory. Over-allocating will show up in

the following way. While a user process is executing, a page fault occurs. The hardware traps to the operating system, which checks its internal tables to see that this is a fault and not an illegal memory access. The operating system determines where the desired page is residing on the disk, but then finds there are no free frames on the free-frame list; all memory is in user. The operating system has several options at this point. It could terminate the user process. We could swap out a process, freeing all its frames, and reducing the level of multiprogramming.

Page Replacement



It page replacement takes the following approach. If no frame is free,

i. Find the location of the desired page on the disk.
ii. Find a free frame:
   a) If there is a free frame, use it.
   b) Otherwise, use a page-replacement algorithm to select a victim frame.
   c) Write the victim page to the disk; change the page and frame tables accordingly.
3. Read the desired page into the (newly) free frame; change the page and frame tables.
4. Restart the user process.

We must solve two major problems to implement demand paging:

We must, develop a frame-allocation algorithm and a page-replacement algorithm.

Virtually memory is commonly implemented by demand paging. It can also be implemented in a segmentation system. Several systems provide a paged segmentation scheme, where segment are broken into pages.

2.8 DEMAND PAGING

A demand–paging system is similar to a paging system with swapping. Processes reside on secondary memory (which is usually a disk) When we want to execute a process, we swap it into memory.

The hardware to support demand paging is the same as the hardware for paging and swapping.

- **Page table**
  This table has the ability to mark an entry invalid through a valid-invalid bit or special value of protection bits.
- **Secondary memory**
  This memory holds those pages that are not present in main memory. The secondary memory is usually a high-speed disk. It is known as swap space or backing store device, and the section of disk used for this purpose is known as swap space or backing store.

2.8.1 Performance of Demand Paging

Demand paging can have a significant effect on the performance of a computer system; let us compute the effective access time for a demand-paged memory.

2.9 PAGE-REPLACEMENT

ALGORITHM

We evaluate an algorithm by running it on a particular string of memory reference and computing the number of page faults.

The string of memory references is called a reference string. To illustrate the page-replacement algorithms, we shall use the reference string 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0 1for a memory with three frames.

2.9.1 FIFO Algorithm

The simplest page-replacement algorithm is a FIFO algorithm. A FIFO replacement algorithm associates with each page the time when that page is chosen. It is not necessary to record the time when a page is brought in. We can create a FIFO queue to hold all pages in memory. We replace the page at the head of the queue. When a page is brought into memory, we insert it at the tail of the queue. Reference string

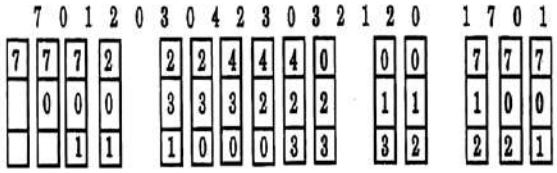


Fig. FIFO page-replacement algorithm.

There are 15 faults altogether. Belady's anomaly reflects the fact that, for some page-replacement algorithms, the page-fault rate may increase as the number of allocated frames increases.

2.9.2 Optimal Algorithm

Belady's anomaly shows the search for an optimal page-replacement algorithm. An optimal page-replacement algorithm has the lowest page-fault rate of all algorithms. An optimal algorithm will never suffer from Belady's anomaly. An optimal page-replacement algorithm exists, and has been called OPT or MIN. It is simply replace the page that will not be used for the longest period of time. Unfortunately, the optimal page-replacement algorithm is difficult to implement, because it requires future knowledge of the reference string. Reference string page frames

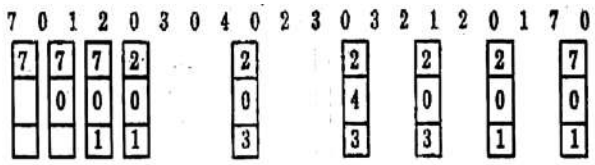
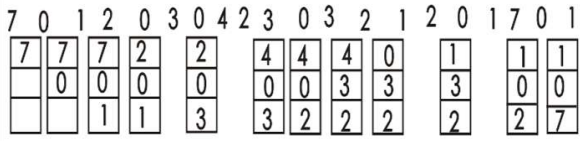


Fig. Optimal page-replacement algorithm



2.9.3 LRU Algorithm

We can replace the page that has not been used for the longest period of time. This approach is the least recently used (LRU) algorithm. LRU replacement associates with each page the time of that page's last use. When a page must be replaced, LRU chooses that page that has not been used for the longest period of time. This strategy is the optimal page-replacement algorithm looking backward in time, rather than forward. The result of applying LRU replacement to out example reference string is shown in Fig. below The LRU policy is often used as a page-replacement algorithm and is considered to be quite good. The major problem is how to implement LRU replacement. An LRU page-replacement algorithm may require substantial hardware assistance. The problem is to determine an order for the frames defined by the time of last use. Two implementations are feasible:

2.9.4 Counters

In the simplest case, we associate with each page-table entry a time –of –use field, and add to the CPU a logical clock or

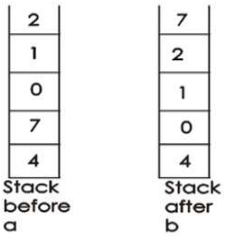
counter. The clock is incremented for every memory reference. Whenever a reference to a page is made, the contents of the clock resister are copied to the time-of use field in the page table for that page. In this way, we always have the "time" of the last reference to each page. We replace the page with the smallest time, This scheme require a search of the page table to find the LRU page, and a write to memory (to the –to –use field in the page table) for each memory access. The times must also be maintained when page tables are changed (due to CPU scheduling). Overflow of the clock must be considered.

2.9.5 Stack

Another approach to implementing LRU replacement is to keep a stack of page numbers. Whenever a page is referenced, it is removed from the stack and put on the top. In this way, the top of the stack is always the most recently used page and the bottom is the LRU. Because entries must be removed from the middle of the stack, it is best implemented a doubly linked list, with a head and tail pointer.

Reference string

4 7 0 7 1 0 1 2 1 2 7 1 2



2.10 COUNTING ALGORITHMS

There are many other algorithm that can be used for page replacement. For example, we could keep a counter of the number of references that have been made to each page, and develop the following two schemes.

2.10.1 LFU Algorithm:

The least frequently used (LFU) page-replacement algorithm requires that the page with the smallest count be replaced. The reason for this selection is that an activity used page should have a large reference count. This algorithm suffers from the situation in which a page is used heavily during the initial phase of a process, but then is never used again.

2.10.2 MFU Algorithm:

The most frequently used (MFU) page-replacement algorithm is based on the argument that the page with the smallest count was probably just brought in and has yet to be used.

2.10.3 Page Buffering Algorithm

Systems commonly keep a pool of free frames. When a page fault occurs, a victim frame is chosen as before. However, that desired page is read into a free frame from the pool before the victim is written out. This procedure allows the process to restart as soon as possible, without waiting for the victim page to be written out.

2.11 ALLOCATION OF FRAMES

The simplest case of virtual memory is the single-user system. Consider a single-user system with 128K memory composed of pages of size 1K Thus, there are 128 frames. The operating system may take 35K, leaving 93 frames for the user process. Under pure demand paging, all 93 page faults would all get free frames from the free-frame list. When a user the free-frame list was exhausted, a page-replacement algorithm would be used to select one of the 93 in-memory pages to be replaced with the ninety-fourth, and so

on. When the process terminated, the 93 frames would once again be placed on the free-frame list.

2.11.1 Minimum Number of Frames

There are, various constraints on our strategies for the allocations of frames, like

i) We cannot allocate more than the total number of available frames (unless there is page sharing).

ii) There are a minimum number of frames that can be allocated. Obviously, as the number of frames allocated to each process decreases, the page fault-rate increases, slowing process execution. The minimum number of frames per process is defined by the architecture, whereas the maximum number is defined by the amount of available physical memory.

2.12 THRASHING

If the number of frames allocated to a low-priority falls below the minimum number require by the computer architecture, we must suspend that process' execution. If the process does not have this number of frames, it will very quickly page fault. At this point, it must replace some page. However, since all its pages are in active use, it must replace a page that will be needed again right away. Consequently, it quickly faults again, and again. The process continues to fault, replacing pages for which it will then and bring back away. This high paging activity is called thrashing. A process is thrashing if it is spending more time paging than executing.

If the initial degree of multiprogramming is increased, the CPU utilization of the system will be very high, but if we still increase the degree of multiprogramming, after some point '' the CPU utilization will degrade.

Cause:

1. High degree of multiprogramming.
2. Lack of frames (Main memory).

Example:

Consider a system with CPU utilization as 10% and the paging disk 90%, then which of the following will improve CPU utilization.

1. Install bigger disk : NO
2. Install faster CPU : NO
3. Increase degree of multiprogramming : NO
4. Decrease degree of multiprogramming : YES
5. Install more main memory : YES
6. Increase Page Size : YES
7. Decrease Page Size : YES

GATE QUESTIONS

Q.1 Consider a machine with 64 MB physical memory and a 32-bit virtual address space. If the page size is 4 Kbyte, what is the approximate size of the page table?

a) 16 Mbyte b) 8 Mbyte
c) 2 Mbyte d) 24 Mbyte

[GATE-2001]

Q.2 Which of the following statements is false?

a) Virtual memory implements the translation of a program's address space into physical memory address space
b) Virtual memory allows each program to exceed the size of the primary memory
c) Virtual memory increases the degree of multi-programming
d) Virtual memory reduces the context switching overhead

[GATE-2001]

Q.3 Consider a virtual memory system with FIFO page replacement policy: For an arbitrary page access pattern, increasing the number of page frames in main memory will

a) always decrease the number of page faults
b) always increase the number of page faults
c) sometimes increase the number of page faults
d) never affect the number of page faults

[GATE-2001]

Q.4 In a system with 32 bit virtual addresses and 1 Kbyte page size, use of one-level page tables for virtual to physical address translation is not-practical because of

a) the large amount of internal fragmentation
b) the large amount of external fragmentation
c) the large memory overhead in maintaining page tables
d) the large computation overhead in the translation process

[GATE-2003]

Common Data for Questions 5 and 6

A processor uses 2-level page tables for virtual to physical address translation. Page tables for both levels are stored in the main memory. Virtual and physical addresses are both 32 bits wide. The memory is byte addressable. For virtual to physical address translation, the 10 most significant bits of the virtual address are used as index into the first level page table while the next 10 bits are used as index into the second level page table.
The 12 least significant bits of the virtual address are used as offset within the page. Assume that the page table entries in both levels of page tables are 4 bytes wide. Further, the processor has a Translation Look-aside Buffer (TLB), with a hit rate of 96%. The TLB caches recently used virtual page numbers and the corresponding physical page numbers. The processor also has a physically addressed cache with a hit rate of 90%. Main memory access time is 10 ns, cache access time is 1 ns, and TLB access time is also 1 ns.

Q.5 Assuming that no page faults occur, the average time taken to access a virtual address is approximately (to the nearest 0.5 ns).

a) 1.5 ns b) 2 ns
c) 3 ns d) 4 ns

[GATE-2003]

Q.6 Suppose a process has only the following pages in its virtual address space: two contiguous code pages starting at virtual address 0x00000000, two contiguous data pages starting at virtual address 0x00400000, and a stack page starting at virtual address 0xFFFFF000. The amount of memory required for storing the page tables of this process is

a) 8 Kbyte b) 12 Kbyte
c) 16 Kbyte d) 20 Kbyte

[GATE-2003]

Q.7 The minimum number of page frames that must be allocated to a running process in a virtual memory environment is determined by

a) the instruction set architecture
b) page size
c) physical memory sire
d) number of processes in memory

[GATE-2004]

Q.8 Consider a system with a two-level paging scheme in which a regular memory access takes 150 ns and servicing a page fault takes 8 ms. An average instruction takes 100 ns of CPU time, and two memory accesses. The TLB hit ratio is 90%, and the page fault rate is one in every 10,000 instructions. What is the effective average instruction execution time?

a) 645 ns b) 1050 ns
c) 1215 ns d) 1230 ns

[GATE-2004]

Q.9 A CPU generates 32-bit virtual addresses. The page size is 4 Kbyte. The processor has a transition look-aside buffer (TLB) which can hold a total of 128 page table entries and is 4-way set associative. The minimum size of the TLB tag is

a) 11 bit b) 13 bit
c) 15 bit d) 20 bit

[GATE-2006]

Q.10 A computer system supports 32-bit virtual addresses as well as 32-bit physical addresses. Since, the virtual address space is of the same size as the physical address space, the operating system designers decide to get rid of the virtual memory entirely. Which one of the following is true?

a) Efficient implementation of multi-user support is no longer possible
b) The processor cache organization can be made more efficient now
c) Hardware support for memory management is no longer needed
d) CPU scheduling can be made more efficient now

[GATE-2006]

Q.11 virtual memory system uses First In First Out (FIFO) page replacement policy and allocates a fixed number of frames to a process. Consider the following statements:

P :Increasing the number of page frames allocated to a process

sometimes increases the page fault rate.

Q :Some programs do not exhibit locality of reference.

Which one of the following is true?

a) Both P and Q are true, and Q is the reason for P
b) Both P and Q are true, but Q is not the reason for P
c) P is false But Q is true
d) Both P and Q are false

[GATE-2007]

Statements for Linked Answer Questions 12 to 13

A process has been allocated 3 page frames. Assume that none of the pages of the process are available in the memory initially. The process makes the following sequence of page references (reference string) 1, 2, 1, 3, 7, 4, 5, 6, 3, 1

Q.12 If optimal page replacement policy is used, how many page faults occur for the above reference string?

a) 7 b) 8
c) 9 d) 10

[GATE-2007]

Q.13 Least Recently Used (LRU) page replacement policy is a practical approximation to optimal page replacement. For the above reference string, how many more page faults occur with LRU than with the optimal page replacement policy?

a) Zero b) 1
c) 2 d) 3

[GATE-2007]

Q.14 A processor uses 36 bit physical addresses and 32 bit virtual addresses, with a page frame size of 4 Kbyte. Each page table entry is of size 4 byte. A three level page table is used for virtual to physical address translation, where the virtual address is used as follows

Bit 30-31 are used to index into the first level page table

Bit 21-29 are used to index into the second level page table

Bit 12-20 are used to index into the third level page table, and

Bit 0-11 are used as offset within the page

The number of bits required for addressing the next level page table (or page frame) in the page table entry of the first, second and third level page tables are respectively

a) 20, 20 and 20
b)24,24 and 24
c) 24, 24 and 20
d)25,25 and 24

[GATE-2008]

Q.15 In which one of the following page replacement policies, Belady's anomaly may occur?

a) FIFO b) Optimal
c) LRU d) MRU

[GATE-2009]

Q.16 The essential content(s) in each entry of a page table is/are

a) virtual page number
b) page frame number
c) Both virtual page number and page frame number
d) access right information

[GATE-2009]

Q.17 A multilevel page table is preferred in comparison to a single level page table for translating virtual address to physical address because

a) it reduces the memory access time to read or write a memory location

b) it helps to reduce the size of page table needed to implement the virtual address space of a process
c) it is required by the translation look-aside buffer
d) it helps to reduce the number of page faults in page replacement algorithms.

[GATE-2009]

Q.18 A system uses FIFO policy for page replacement. It has 4 page frames with no pages loaded to begin with. The system first accesses 100 distinct pages in some order and then accesses the same 100 pages but now in the reverse order. How many page faults will occur?

a) 196 b) 192
c) 197 d) 195

[GATE-2010]

Q.19 Consider the virtual page reference string: 1, 2, 3, 2, 4, 1, 3, 2, 4, 1

On a demand paged virtual memory system running on a computer system that has main memory size of 3 page frames which are initially empty. Let LRU FIFO and PTIMAL denote the number of page faults under the corresponding page replacement policy. Then,

a) OPTIMAL < LRU < FIFO
b) OPTIMAL < FIFO < LRU
c) OPTIMAL = LRU
d) OPTIMAL = FIFO

[GATE-2012]

Statements for Linked Answer Questions Q.20 & Q.21:

A computer uses 46-bit virtual address, 32-bit physical address ree-level paged table organization, the page table base register stores the base address of 1st level page table (T1), which occupies exactly one page, each entry of T1 stores the base address of a page of the second-level page table T2, each entry of T2 stores the base address of a page of third level page level (T3), each entry of T3 stores a page tableentry (PTE) the PTE is 32 bit in size. The process or used in the computer has a 1 MB 16-way set associative virtually indexed tagged cache. Cache block size is 64 bytes.

Q.20 What is the size of a page in KB in this computer?

a) 2 b) 4
c) 8 d) 16

[GATE-2013]

Q.21 What is the minimum number of page colors needed to guarantee that no two synonyms map to different sets in the processor cache of this computer?

a) 2 b) 4
c) 8 d) 16

[GATE-2013]

Q.22 Assume that there are 3 page frames which are initially empty. If the page reference string is 1, 2, 3, 4, 2, 1, 5, 3, 2, 4, 6 the number of page faults using the optimal page replacement policy is ___.

[GATE–2014]

Q.23 A computer has twenty physical page frames which contain numbered 101 through 120. Now a program accesses the pages numbered 1, 2, ….. 100 in that order, and repeats the sequence THRICE. Which one of the following page replacement policies experience the same number of page faults as the

optimal page replacement policy for this program?

a) Least-recently-used
b) First-in-First-out
c) Last-in-First-out
d) Most-recently-used

[GATE–2014]

Q.24 A system uses 3 page frames for storing process pages in main memory. It uses the Least Recently Used [LRU] page replacement policy. Assume that all the page frames are initially empty. What is the total number of page faults _____that will occur which processing the page reference string given below? 4, 7, 6, 1, 7, 6, 1, 2, 7, 2

[GATE–2014]

Q.25 Consider a paging hardware with a TLB. Assume that the entire page table and all the pages are in the physical memory. It takes 10 milliseconds to search the TLB and 80 milliseconds to access the physical memory. If the TLB hit ratio is 0.6, the effective memory access time (in milliseconds) is _____.

[GATE–2014]

Q.26 Consider a system with byte-addressable memory, 32 bit logical addresses, 4 kilobyte page size and page table entries of 4 bytes each. The size of the page table in the system in megabytes is _____.

[GATE–2015]

Q.27 Consider a main memory with five page frames and the following sequence of page references: 3, 8, 2, 3, 9, 1, 6, 3, 8, 9, 3, 6, 2, 1, 3. Which one of the following is true with respect to page replacement policies First-In-First Out (FIFO) and Least Recently Used (LRU)?

a) Both incur the same number of page faults
b) FIFO incurs 2 more page faults than LRU
c) LRU incurs 2 more page faults than FIFO
d) FIFO incurs 1 more page faults than LRU

[GATE–2015]

Q.28 A computer system implements a 40-bit virtual address, page size of 8 kilobytes, and a 128- entry translation look-aside buffer (TLB) organized into 32 sets each having four ways. Assume that the TLB tag does not store any process id. The minimum length of the TLB tag in bits is _____.

[GATE–2015]

Q.29 Consider six memory partitions of sizes 200 KB, 400 KB, 600 KB, 500 KB, 300 KB and 250 KB, where KB refers to kilobyte. These partitions need to be allotted to four processes of sizes 357 KB, 210KB, 468 KB and 491 KB in that order. If the best fit algorithm is used, which partitions are NOT allotted to any process?

a) 200KB and 300 KB
b)200KB and 250 KB
c)250KB and 300 KB
d)300KB and 400 KB

[GATE–2015]

Q.30 Consider a computer system with 40-bit virtual addressing and page size of sixteen kilobytes. If the computer system has a one-level page table per process and each page table entry requires 48 bits,

then the size of the per-process page table is _____ megabytes.

[GATE–2016]

Q.31 Consider a computer system with ten physical page frames. The system is provided with an access sequence (a1, a2,..., a20, a1, a2,... a20), where each ai is a distinct virtual page number. The difference in the number of page faults between the last-in-first-out page replacement policy and the optimal page replacement policy is _____.

[GATE–2016]

Q.32 In which one of the following page replacement algorithms it is possible for the page fault rate to increase even when the number of allocated frames increases?

a) LRU (Least Recently Used)
b)OPT (Optimal Page Replacement)
c) MRU (Most Recently Used)
d) FIFO (First In First Out)

[GATE–2016]

Q.33 Consider a machine with a byte addressable main memory of 2^{32} bytes divided into blocks of size 32 bytes. Assume that a direct mapped cache having 512 cache lines is used with this machine. The size of the tag filed in bits is ___.

[GATE–2017]

Q.34 Consider a 2-way set associative cache with 256 blocks and uses LRU replacement, initially the cache is empty. Conflict misses are those misses which occur due to contention of multiple blocks for the same cache set. Compulsory misses occur due to first time access to the block. The following sequence of accesses to memory blocks (0,128,256,128,0,128,256, 128,1,129,257,129,1,129,257,129) is repeated 10 times . The number of conflict misses experienced by the cache is ___.

[GATE–2017]

ANSWER KEY:

1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
(c)	(d)	(c)	(c)	(d)	(c)	(a)	(d)	(c)	(c)	(b)	(a)	(c)	(d)	(a)
16	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**
(b)	(b)	(a)	(b)	(c)	(c)	7	(d)	6	122	4	(a)	22	(a)	384
31	**32**	**33**	**34**											
1	(d)	18	78											

EXPLANATIONS

Q.1 **(c)**

Size of page table =
(total number of page table entries) *(size of a page table entry)
= (2^20 *2) = 2MB

Q.2 **(d)**

In a system with virtual memory context switch includes extra overhead in switching of address spaces.

Q.3 **(c)**

Increasing the number of page faults in main memory in case of FIFO page replacement policy will cause Belady's anomaly to occur thus, the number of page faults is increased though in general increase in number of page frames. Decreases the number of page faults.

Q.4 **(c)**

We are given with the page size of 1 Kbyte.
We know that
Kbyte = 2 ^ 10 byte; Virtual address is 32 bit long
So, required number of Pages
= 2^21/2 ^ 10 = 2 ^ 22
The required number of pages is very large hence, it is not practical to implement. The reason behind is that we 11 face an issue of large memory overhead in the maintenance of, the page table.

Q.5 **(d)**

From the given,

Average time = [(0.96)(1+0.9+0.1×11)] + [(0.04)(21+ 0.9 + 0.1 × 11)] = 3.87 ns
This is nearest to 4 ns.

Q.6 **(c)**

Taking the values from the given,
Amount of memory required
$= 4 \times 2^{12}$ byte
$= 4 \times 2^2 \times 2^{10}$
$= 16 \times 2^{10}$ byte
= 16 kbyte

Q.7 **(a)**

By instruction set architecture, minimum numbers of page frames that must be allocated are determined. Page frames are created in the virtual memory to minimize the system load and to improve its performance when the processes are in memory.

Q.8 **(d)**

Let the probability of page fault rate

$F = \frac{1}{2} \times 10^4$

Total memory access time
=0.90 * 150 +0.10 x 300 =165ns
Instruction execution time
$= 100 + 2[165 + ½ \times 10^4 \times 8 \times 10^6]$
=100 + 2 × 565 =1230ns

Q.9 **(c)**

Size of a page = 4KB = 2^12
Total number of bits needed to address a page frame = 32 – 12 = 20
If there are 'n' cache lines in a set, the cache placement is called n-

way set associative. Since TLB is 4 way set associative and can hold total 128 (2^7) page table entries, number of sets in cache =2^7/= 2^5. So 5 bits are needed to address a set, and 15(20–5) bits are needed for tag.

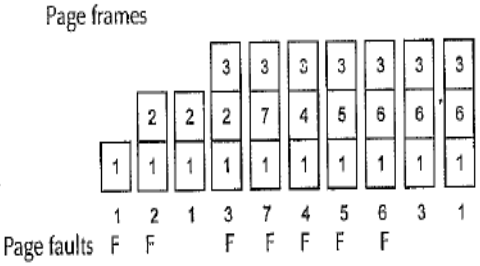
Q.10 (c)

For supporting virtual memory, special hardware support is needed from Memory Management Unit. Since operating system designers decide to get rid of the virtual memory entirely, hardware support for memory management is no longer needed.

Q.11 (b)

As we know that in case of Belady's anomaly, the increase in number of pages results in increase of page fault rate so both P and Q are true but Q is not the reason for P.

Q.12 (a)



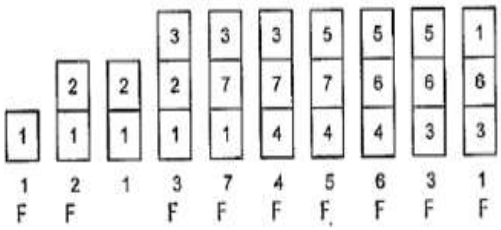
Optimal page replacement policy replaces the page which will not be used for longest time.

Total number of page faults = 7

Q.13 (c)

LRU replaces the page which is used least recently.

Page frames



Total number of page faults = 9

Therefore, an increase of 2 is there (97) in LRU than in optimal.

Q.14 (d)

Virtual address size = 32 bits

Physical address size = 36 bits

Physical memory size = 2^36 bytes

Page frame size=4K bytes=2^12 bytes

No. of bits required to access physical memory frame =36-12= 24

So in third level of page table, 24 bits are required to access an entry.9 bits of virtual address are used to access second level page table entry and size of pages in second level is 4 bytes. So size of second level page table is (2^9)*4 = 2^11 bytes. It means there are (2^36)/(2^11) possible locations to store this page table. Therefore the second page table requires 25 bits to address it. Similarly, the third page table needs 25 bits to address it.

Q.15 (a)

In Belady's anomaly, if number of frames are increased the number of page faults increases. This behaviour found only with FIFO. The First In, First out (FIFO) page replacement algorithm is a low-overhead algorithm. In FIFO, the operating system keeps track of all the pages in memory in a queue. The queue is managed in a way that the most recent arrival is placed at the back, and the earliest arrival in front. T the time of page

replacement, oldest page is selected to get replaced. Practical implementation of FIFO is not very good as the algorithm experiences Belady's anomaly. In FIFO when the number of frames increases then, the number of page faults also increases and this behaviour is found by Belady's anomaly. Thus, FIFO have this Belady's anomaly, e.g., in FIFO for . three frames the reference string 1,2, 3, 4, 1,2, 5, 1,2, 3,4, 5 There are 9 page faults and for four frames for the reference string there will be 10 faults.

Q.16 (b)

A page table is the data structure used to store the mapping between virtual addresses and physical addresses. A page table is used by a Virtual memory system in a computer operating system. Virtual addresses are the addresses which are unique to the accessing process while physical addresses are the addresses that are unique to the CPU, i.e., RAM. Page frame number is essential content in each of the page table. Virtual page number may not be store entirely in the page table and thus, cannot be considered as the essential content.

Q.17 (b)

As we know that the, use of multilevel page table reduces the size of the page table needed to implement the virtual address space of a process.

Q.18 (a)

Let's understand the concept of FIFO page replacement algorithm. The First In, First out (FIFO) page replacement algorithm is a low-overhead algorithm. In FIFO, the operating system keeps track of all the pages in memory in a queue. The queue is managed in a way that the most recent arrival is placed at the back, and the earliest arrival in front. T the time of page replacement, oldest page is selected to get replaced. Practical implementation of FIFO is not very good as the algorithm experiences Belady's anomaly. Given are 4 page frames. The 100 pages are accessed in some order→ 1 to pages will get 100 faults. Now, after the pages are accessed, they are to be accessed in reverse order. But in reverse, there won't be any fault for page numbers 100, 99, 98, 97. Therefore, all page frames are occupied. Now, for remaining 96 accesses, 96 faults would occur.

Total 100 (Direct access)+ 96 (Reverse access) = 196

Alternate method

Fault for n pages = 2n -4

So, for 100 pages = 2 × 100 – 4 =196

Q.19 (b)

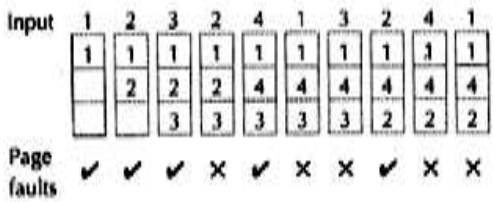
Reference string:

1, 2, 3, 2, 4, 1, 3, 2, 4, 1

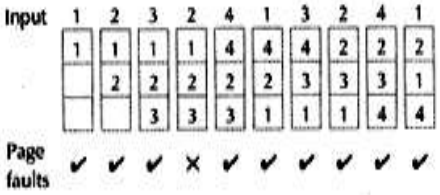
3 page frames.

FIFO

There are 6 page faults.

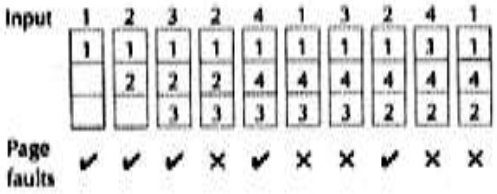


Optimal



There are 5 page faults.

LRU



There are 9 page faults.

Hence, optimal < FIFO < LRU

Q.20 (c)

Let the page size 2^x byte

Then the page off set = x bites.

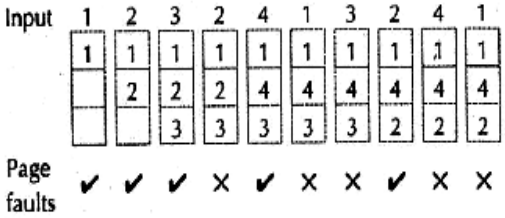
46,x	x

As mentioned, first level page table is contained in one page, each page table entry is 32-bit.

The size of T3 is $=2^{46} \times 2^2/2^x = 2^{46+2-x}$ [∴PTE -32 bit, 3 B = 2^2B]

The size of T2 is = $2^{46+2-x} \times 2^2/ 2^x = 2^{46+4-x}$

The size of T1 is = $2^{46+4-x} \times 2^2/ 2^2 = 2^{46+6-3x}$



∴ 46 = 6 -3x –x ⟹ x= 13 [T1 exactly occupies of page]

That means, page size is of 13 bit or pae size = 2^{13} bytes = 8KB.

Q.21 (c)

As the page size is 2^{13} bytes, we divide cache size by page a size and group 16 pages in one set

No. of pages in cache = 1MB/8KB = 128 pages

No. of sets in cache =128/16 = 8 sets

So minimum use need 8 colors for mapping.

Q.22 (7)

Given three page frames.

Reference string is 1, 2, 3, 4, 2, 1, 5, 3, 2, 4, 6 Initially, there are three page faults and entries are 1 2 3

Page 4 causes a page fault and replaces 3 (3 is the longest distant in future), entries become 1 2 4

Total page faults = 3+1 = 4

Pages 2 and 1 don't cause any fault.

5 causes a page fault and replaces 1, entries become 5 2 4

Total page faults = 4 + 1 = 5

3 causes a page fault and replaces 1, entries become 3 2 4

Total page faults = 5 + 1 = 6

3, 2 and 4 don't cause any page fault.

6 causes a page fault.

Total page faults = 6 + 1 = 7

Q.23 (d)

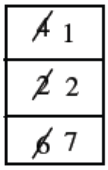
The optimal page replacement algorithm swaps out the page whose next use will occur farthest in the future. In the given question, the computer has 20 page frames and initially page frames are filled with pages numbered from 101 to 120.

Then program accesses the pages numbered 1, 2, ..., 100 in that order, and repeats the access sequence THRICE. The first 20 accesses to pages from 1 to 20

would definitely cause page fault. When 21st is accessed, there is another page fault. The page swapped out would be 20 because 20 is going to be accessed farthest in future. When 22nd is accessed, 21st is going to go out as it is going to be the farthest in future. The above optimal page replacement algorithm actually works as most recently used in this case. As a side note, the first 100 would cause 100 page faults, next 100 would cause 81 page faults (1 to 19 would never be removed), the last 100 would also cause 81 page faults.

Q.24 (6)

Page reference string is
4, 7, 6, 1, 7, 6, 1, 2, 7, 2



Implementing LRU using 3 page frames
Total page faults =6

Q.25 (122)

As both page table and page are in physical memory
T(eff) = hit ratio * (TLB access time + Main memory access time) + (1 - hit ratio) * (TLB access time + 2 * main memory time)
= 0.6*(10+80) + (1-0.6)*(10+2*80)
= 0.6 * (90) + 0.4 * (170) = 122

Q.26 (4)

Number of entries in page table = 2^{32}/ 4KB = 2^{32}/ 2^{12} = 2^{20}
Size of page table = (Number of page table entries) * (Size of an entry)
= 2^{20} * 4 = 2^{22} = 4 MB

Q.27 (a)

3, 8, 2, 3, 9, 1, 6, 3, 8, 9, 3, 6, 2, 1, 3
In both FIFO and LRU, we get following after considering 3, 8, 2, 3, 9, 1, 3 8 2 9 1
FIFO
6 replaces 3
8 2 9 1 6
3 replaces 8
2 9 1 6 3
8 replaces 2
9 1 6 3 8
2 replaces 9
1 6 3 8 2
No more page faults
LRU
6 replaces 8
3 2 9 1 6
8 replaces 2
3 9 1 6 8
2 replaces 1
3 9 6 8 2
1 replaces 8
3 9 6 2 1

Q.28 (22)

Total virtual address size = 40
Since there are 32 sets, set offset = 5
Since page size is 8kilobytes, word offset = 13
Minimum tag size = 40 - 5- 13 = 22

Q.29 (a)

Best fit allocates the smallest block among those that are large enough for the new process. So the memory blocks are allocated in below order.
357 → 400
210 → 250
468 → 500
491 → 600
So the remaining blocks are of 200 KB and 300 KB

Q.30 (384)

Size of memory = 2^{40}
Page size = 16KB = 2^{14}
No of pages= size of Memory/ page size = 2^{40} / 2^{14} = 2^{26}
Size of page table = 2^{26} * 48/8 bytes
= 2^6*6 MB =384 MB

Q.31 (1)

LIFO:

a1 to a10 will result in page faults, So 10 page faults from a1 to a10. Then a11 will replace a10(last in is a10), a12 will replace a11 and so on till a20, so 10 page faults from a11 to a20 and a20 will be top of stack and a9...a1 are remained as such. Then a1 to a9 are already there. So 0 page faults from a1 to a9.a10 will replace a20, a11 will replace a10 and so on. So 11 page faults from a10 to a20. So total faults will be 10 + 10 + 11 = 31.

Optimal

a1 to a10 will result in page faults, So 10 page faults from a1 to a10. Then a11 will replace a10 because among a1 to a10, a10 will be used later, a12 will replace a11 and so on. So 10 page faults from a11 to a20 and a20 will be top of stack and a9...a1 are remained as such. Then a1 to a9 are already there. So 0 page faults from a1 to a9. a10 will replace a1 because it will not be used afterwards and so on, a10 to a19 will have 10 page faults. a20 is already there, so no page fault for a20. Total faults 10+10+10=30.
Difference = 1

Q.32 (d)

In FIFO scheduling algorithm, it is possible that page fault rate increase
even if number allocated frames increases.

Q.33 (18)

MM size =2^{32} B, Block size =32 B
Direct CM
lines = 512
Address format is

32 bit

Tag	LO	WO
18 bit	$\log_2 512$ =9 bit	$\log_2 32$ =5 bit

Q.34 (78)

Number of lines=256

Number of sets (S) = $\frac{256}{2} = 128$

Cache memory

0	~~0~~ 256 ~~0~~ 256	128
1	~~1~~ 257 ~~1~~ 257	129
2		
⋮		
127		

{K mode S = *i*}

LRU

1st time access
0-M (compulsory); 0 mod 128 =0
128-M(compulsory);128 mod 128 =0
256 – M (compulsory and conflict);
256 mod 128 =0
128-H: 128 mod 128 =0
0-M (conflict): 0 mod 128 =0
128- H: 128 mod 128 =0
256 –M (conflict):256 mod 128=0
128-H: 128 mod 128 =0
1-M (compulsory):1 mod 128 = 1

129-M(compulsory):129mod 128=1
257-M(compulsory and conflict): 257 mod 128=1
129-H: 129 mod 128=1
1m (compulsory and conflict):1mod 128=1
129-H: 129 mod 128=1
257–M (compulsory and conflict): 257 mod 128=1
129-H: 129 mod 128=1
Number of conflict miss (C)=6
2nd time access
Cache memory contains some blocks So, 0 becomes conflict miss
1 becomes conflict miss So, two more conflict misses are increases.
9 iteration ⇒ 9×8=72
Total conflict misses = 72+6=78

3 INTERRUPTS

3.1 INTRODUCTION

The two main jobs of a computer are I/O and processing. In many cases, the main job is I/O, and the processing is merely incidental. For instance, when we browse a web page or edit a file, our immediate interest is to read or type in some information; it is not to compute an answer. The role of the operating system in computer I/O is to manage and control I/O operations and I/O devices. In many computer architectures, 3 CPU-instruction cycles are sufficient to poll a device: **read** a device register, **logical-and** to extract a status bit, and **branch** if not zero. The hardware mechanism that enables a device to notify the CPU is called an interrupt.

3.2 INTERRUPTS

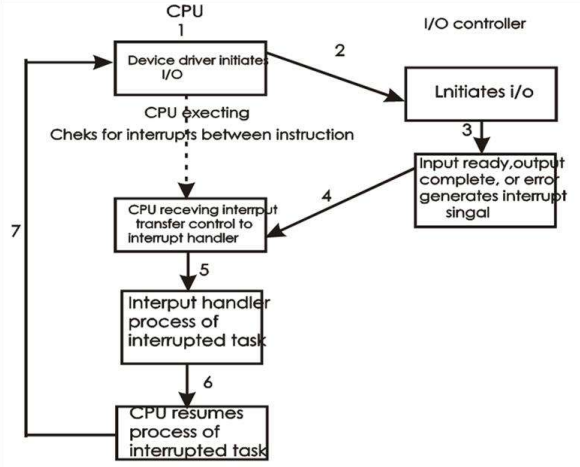
The basic interrupt mechanism works as follows:

The CPU hardware has a wire called the interrupt line that the CPU senses after executing every instruction. When the CPU detects that a controller has asserted a signal on the interrupt request line, the CPU saves a small amount of state, such as the current value of the instruction pointer, and jumps to the interrupt-handler routine at a fixed address in memory. The interrupt handler determines the cause of the interrupt, performs the necessary processing, and executes a **return from interrupt** instruction to return the CPU to the execution state prior to the interrupt. The device controller raises an interrupt by asserting a signal on the interrupt request line, the CPU catches the interrupt and dispatches to the interrupt handler, and the handler clears the interrupt by servicing the device.

Most CPUs have two interrupt request lines:

i) The <u>non-maskable interrupt,</u> which is reserved for events such as unrecoverable memory errors.

ii) The line is <u>maskable</u>: It can be turned off by the CPU before the execution of critical instruction sequence that must not be interrupted. The maskable interrupt is used by device controllers to request service.



Although the hardware aspects of I/O are complex when considered at the level detail of electronic-hardware designers, the concepts that we have just described are sufficient to understand many I/O aspects of operating systems.

The main concepts:

i) A bus

ii) A controller

iii) An I/O port and its registers

iv) The handshaking relationship between the host and a device controller.

v) The execution of this handshaking in a polling loop or via interrupts

vi) The offloading of this work to a DMA controller for large transfers

3.3 APPLICATIONS I/O INTERFACE

Devices vary in many dimensions, as shown in Figure below,

i) Character-stream or block: A character-stream device transfers bytes one by one whereas a block device transfers a block of bytes as a unit.

ii) Sequential or random-access: A sequential device transfers data in a fixed order that is determined by the device, whereas the user of a random-access device can instruct the device to seek to any of the available data storage locations.

ASPECT	VARIATION	EXAMPLE
data-transfer mode	Character block	Terminal disk
access method	sequential random	Modem CD-ROM
transfer schedule	synchronous asynchronous	Tape keyboard
Sharing	dedicated	Tape keyboard
device speed	Latency seek time transfer rate delay	
I/O direction	read only write only read-write	CD-ROM graphics controller

Fig Characteristics of I/O devices

i) Synchronous or asynchronous: A synchronous device is one that performs data transfers with predictable response times. An asynchronous device exhibits irregular or unpredictable response times.

ii) Sharable or dedicated: A sharable device can be used concurrently by several processes or threads; a dedicated device cannot,

iii) Speed of operation: Device speeds range from a few bytes per second to a few Gigabytes per second.

iv) Read-write, read only, or writes only : Some devices perform both input and output, but others support only one data direction.

3.4 BLOCK AND CHARACTER DEVICES

The block-device interface captures all the aspects necessary for accessing disk drives and other block-oriented devices. The exception is that the device understands commands such as read and write, and, if it is a random-access device, it has a seek command to specify which block to transfer next. Applications normally access such a device through a file-system interface. A keyboard is a device that is accessed through a character-stream interface. The basic system calls in this interface enable an application to get or put one character.

3.5 KERNAL I/O SUBSYSTEM

Kernels provide many services related to I/O.

The services that are provided by the Kernel are

i) I/O scheduling,

ii) Buffer caching,

iii) Spooling,

iv) Device reservation, and

v) Error handling.

The I/O system coordinates an extensive collection of services, which are available to applications and to other parts of the kernel.

The I/O subsystem supervises:

i. The management of the name space for files and devices
ii. Access control to files and devices
iii. Operation control (for example, a modem cannot seek)
iv. File system space allocation
v. Device allocation
vi. Buffering, caching, and spooling
vii. I/O scheduling
viii. Device status monitoring, error handling, and failure recovery
ix. Device driver configuration and initialization

The upper levels of the I/O subsystem access devices via the uniform interface provided by the device drives.

3.5.1 Transforming I/O Requests to Hardware Operations

We described the handshaking between a device driver and a device controller, but we did not explain how the operating system connects an application request to a set of network wires or to a specific disk sector. Consider the example of reading a file from disk. The application refers to the data by a file name. Within a disk, it is the job of the file system to map from the file name through the file-system directories to obtain the space allocation of the file.

3.5.2 Typical Life Cycle of a I/O Request

i. A process issues a blocking read system call to a file descriptor of file that has been opened previously.
ii. The system-call code in the kernel checks the parameters for correctness. In the case of input, if the data are already available in the buffer cache, the data are returned to the process and the I/O request is completed.
iii. Otherwise, a physical I/O needs to be performed, so the process is removed from the run queue and is placed on the wait queue for the device, and the I/O request is scheduled. Eventually, the I/O subsystem sends the request to the device driver.

   Depending on the operating system, the request is sent via a subroutine call or via an in-kernel message.
iv. The device driver allocates kernel buffer space to receive the data, and schedules the I/O. Eventually, the driver sends commands to the device controller by writing into the device control registers.
v. The device controller operates the device hardware to perform the data transfer.
vi. The driver may poll for status and data, or it may have set up a DMA transfer into Kernel memory. We assume that the transfer is managed by a DMA controller, which generates an interrupt when the transfer completes.
vii. The correct interrupt handler receives the interrupt via the interrupt-vector table, stores any necessary data, signals the device driver, and returns from the interrupt.
viii. The device driver receives the signal, determines which I/O request completed, determines the request's status, and signals the kernel I/O subsystem that the request has been completed.
ix. The kernel transfers data to return codes to the address space of the

requesting process, and moves the process from the wait queue back to the ready queue.

x. Moving the process to the ready queue unblocks the process. When the scheduler assigns the process to the CPU, the process resumes execution at the completion of the system call.

3.6 PERFORMANCE

I/O is a major factor in system performance. It places heavy demands on the CPU to execute device-driver code and to schedule processes fairly and efficiently as they block and unblock.

3.6.1 Disk Scheduling

One of the responsibilities of the operating system is to use the hardware efficiently. For the disk drives, this means having a fast access time and disk bandwidth. The access time has two major components.

i) The seek time is the time for the disk arm to move heads to the cylinder containing rotate the desired sector. The rotational latency is the additional time waiting for the disk to rotate the desired sector to the disk head.

ii) The disk bandwidth is the total number of bytes transferred, divided by the total time between the first request for service and the completion of the last transfer. We can improve both the access time and the bandwidth by scheduling the servicing of disk I/O requests in a good order. Whenever a process needs I/O to or from the disk, it issues a system call to the operating system. The request specifies several pieces of information like

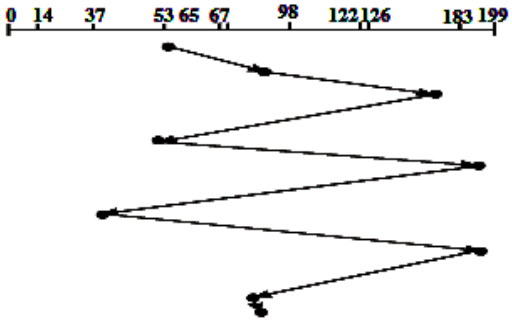
i. Whether this operation is input or output
ii. What the disk address for the transfer is
iii. What the memory address for the transfer is
iv. What the number of bytes to the transferred is

3.6.2 FCFS Scheduling

The simplest form of disk scheduling is, first-come, first-serve (FCFS). This algorithm is intrinsically fair, but it generally does not provide the fastest service. Consider, a disk queue with requests for I/O to blocks the fastest service.

98, 183, 37, 122, 14, 124, 65, 67 in that order. If the disk head is initially at cylinder 53, it will first move from 53, to 98, then to 183, 37, 122, 14, 124 65 and finally to 67, for a total head movement of 640 cylinders. The schedule is diagrammed in the following figure.

The problem with this schedule is illustrated by the wild swing from 122 to 14 and



then back to 124. If the requests for cylinder 37 and 14 could be serviced together, before or after the requests at 122 and 124, the total head movement could be decreased substantially performance could be thereby improved

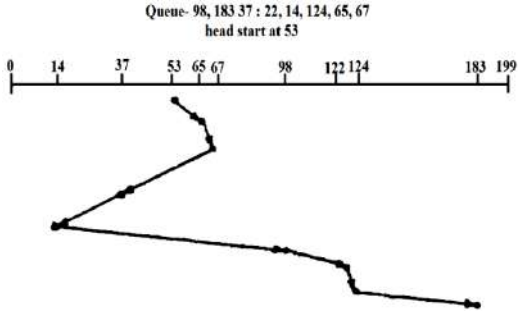
3.6.3 SSTF Scheduling

It is reasonable to service all the requests close to the current head position, before moving the head far away to service other requests. This assumption is the basis for the shortest-seek-time-first(SSTF) algorithm.

For our example request queue, the closest request to the initial head position (53) is at cylinder 65. Once we are at cylinder 65, the next closet request is at cylinder 67. From there, the request at cylinder 37 is closer than 98, so 37 is served next. Continuing, we service the request at cylinder 14, then 98, 122, 124, and finally at 183 (Figure). This scheduling method results in a total head movement of only 236 cylinders, little more than one-third of the distance needed for FCFS scheduling. This algorithm gives a substantial improvement in performance.

SSTF scheduling is essentially a form of shortest-job-first (SJF) scheduling, and, like SJF scheduling, it may cause starvation of some requests.

Although the SSTF algorithm is a substantial improvement over the FCFS algorithm, it is not optimal. In the example, we can do better by moving the head from 53 to 37, even though the latter is not closet, and then to 14, before turning around to service 65, 67, 98, 122, 124 and 183. This strategy reduces the total head movement to 208 cylinders.



3.6.4 SCAN Scheduling

In the SCAN algorithm, the disk arm starts at one end of the disk, and moves toward the other end, servicing requests as it reaches each cylinder, until it gets to the other end of the disk. At the other end, the direction of head movement is reversed, and servicing continues. The head continuously scans back and forth across the disk.

Before applying SCAN to schedule and requests on cylinders 98, 183, 37, 122, 14, 124, 65, and 67, it is required to know the direction of head movement, in addition to the head's current position (53). If the disk arm is moving toward 0, the head will service 37 and then 14. At cylinder 0, the arm will reverse and will move toward the other end of the disk, servicing the requests at 65, 67, 98, 122, 124, and 183. If a request arrives in the queue just in front of the head, it will be serviced almost immediately; a request arriving just behind the head will have to wait until the arm moves to the end of the disk, reverses direction, and comes back.

The SCAN algorithm is sometimes called the elevator algorithm, since the disk arm behaves just like an elevator in a building, first servicing all the requests going up and then reversing to service requests the other way.

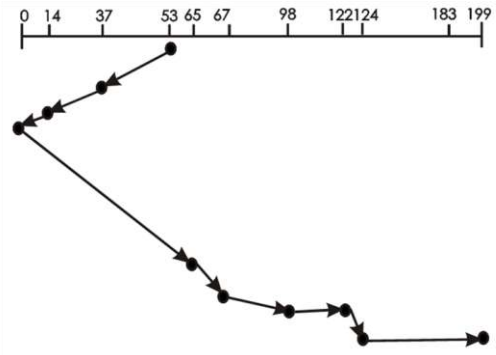


Fig SCAN SCHEDULING

3.6.5 C-SCAN Scheduling

Circular SCAN(C-SCAN) is variant of SCAN that is designed to provide a more uniform wait time, Like SCAN, C-SCAN moves the head from one end of the disk to the other, servicing requests along the way. When the head reaches the other end, however, it immediately returns to the beginning of the disk.

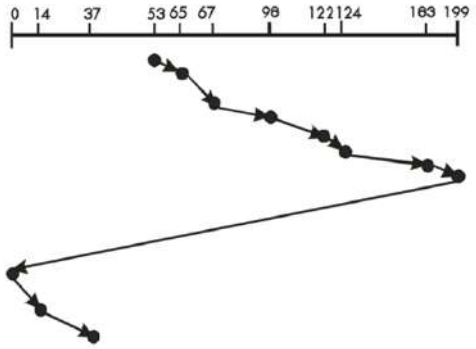


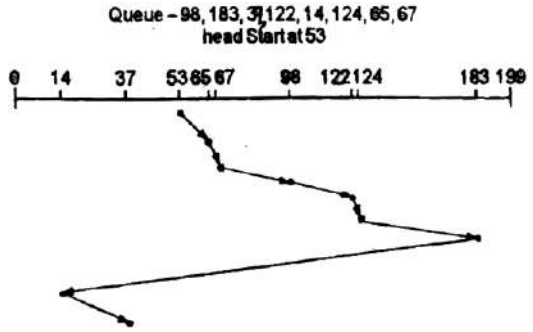
Fig C-SCAN SCHEDULING

3.6.6 LOOK Scheduling

Both SCAN and C-SCAN move the disk arm across the full width of the disk. In practice, neither algorithm is implemented this way. More commonly, the arm goes only as far as the final request in each direction. Then, it reverses direction immediately, without first going all the way to the end of the disk. These versions of SCAN are called LOOK and C-LOOK, because they look for a request before continuing to move in a given direction.

3.6.6 Selection of a Disk-Scheduling Algorithm

SSTF is common and has natural appeal. SCAN and C-SCAN perform better for systems that place a heavy load on the disk, because they are less likely to have the starvation problem.



With any scheduling algorithm, however, performance depends heavily on the number and types of requests. For instance, suppose that the queue usually has just one outstanding request. Then, all scheduling algorithms are forced to heave the same, because they have only one choice for where to move the disk head. They all behave like FCFS scheduling.

The requests for disk service can be greatly influenced by the file-allocation method. A program reading a contiguously allocated file will generate several requests that are close together on the disk, resulting in limited head movement. A linked or indexed file may include blocks that are widely scattered on the disk, resulting in greater head movement. Because of these complexities, the disk-scheduling algorithm should be written as a separate module of the operating system, so that it can be replaced with a different algorithm if necessary.

3.7 STABLE–STORAGE IMPLEMENTATION

We introduced the concept of a write-head log, which required the availability of stable storage. By definition, information residing in stable storage is never lost. To implement such storage, we need to replicate the needed information on multiple storage devices with independent failure modes.

A disk write results in one of three outcomes:

a) **Successful completion:** The data were written correctly on disk.

b) **Partial failure:** A failure occurred in the midst of transfer, so only some of the sectors were written with the new data and the sector being written during the failure may have been corrupted.

c) **Total failure:** The failure occurred before the disk write started, so the previous data values on the disk remain intact. We require that, whenever a failure occurs during writing of a block, the system detects it and invokes a recovery procedure to restore the block to a consistent state. To do that, the system must maintain two physical blocks for each logical block. An output operation is executed as follows:

i. Write the information onto that first physical block.

ii. Went the first write completes successfully, write the same information onto the Second physical block.

iii. Declare the operation complete only after the second write completes successfully

GATE QUESTIONS

Q.1 Which of the following does not interrupt a running process?
a) A device
b) Timer
c) Scheduler process
d) Power failure

[GATE -2001]

Q.2 Consider an operating system capable of loading and executing a single sequential user process at a time. The disk head scheduling algorithm used is First Come First Served (FCFS). If FCFS is replaced by Shortest Seek Time First (SSTF), claimed by the vendor to give 50% better benchmark results, what is the expected improvement in the IN/OUT performance of user programs?
a) 50% b) 40%
c) 25% d) 0%

[GATE-2004]

Q.3 Consider a disk system with 100 cylinders. The requests to access the cylinders occur in following sequence
4, 34, 10, 7, 19, 73, 2, 15, 6, 20
Assuming that the head is currently at cylinder 50, what is the time taken to satisfy all requests, if it takes 1 ms to move from one cylinder to adjacent one and shortest seek time first policy is used?
a) 95 ms b) 119 ms
c) 233 ms d) 276 ms

[GATE-2009]

Q.4 Suppose the following disk request sequence (track numbers) for a disk with 100 tracks is given: 45, 20, 90, 10, 50, 60, 80, 25, 70. Assume that the initial position of the R/W head is on track 50. The additional distance that will be traversed by the R/W head when the Shortest Seek Time First (SSTF) algorithm is used compared to the SCAN (Elevator) algorithm(assuming that SCAN algorithm moves towards 100 when it starts execution) is _____ tracks.

[GATE–2015]

Q.5 Consider a disk queue with requests for I/O to blocks on cylinders 47, 38, 121, 191, 87, 11, 92, 10.
The C-LOOK scheduling algorithm is used. The head is initially at cylinder number 63, moving towards larger cylinder numbers on its servicing pass. The cylinders are numbered from 0 to 199.
The total head movement (in number of cylinders) incurred while servicing these requests is

[GATE–2016]

ANSWER KEY:

1	2	3	4	5										
(a)	(d)	(b)	10	165										

EXPLANATIONS

Q.1 **(a)**

Device can not interrupt a running process but timer, scheduler and power failure can interrupt a running process.

Q.2 **(d)**

It is important to note that SSTF (Shortest Seek Time First) is the optimal scheduling scheme for the access of file only. Therefore, the IN/OUT performance of the user program is determined by many input and output devices and not only by disk. So, when FCFS is replaced by SSTF it improves only disk performance. This does not improve the entire IN/OUT performance.

This implies, improvement performance of user program is 0%.

Q.3 **(b)**

Requests 4, 34, 10, 7, 19, 73, 2, 15, 6, 20

Current 50 34 20 19 15 10 7 6 4 2

Next access 34 20 19 15 10 7 6 4 2 73

Difference 16 14 1 4 5 3 1 2 2 71

Therefore, total moves =16+14+1+4+5+3+1+2+2+71=119ms

Q.4 **(10)**

In Shortest seek first (SSTF), closest request to the current position of the head, and then services that request next. In SCAN (or Elevator) algorithm, requests are serviced only in the current direction of arm movement until the arm reaches the edge of the disk. When this happens, the direction of the arm reverses, and the requests that were remaining in the opposite direction are serviced, and so on.

Given a disk with 100 tracks

And Sequence 45, 20, 90, 10, 50, 60, 80, 25, 70.

Initial position of the R/W head is on track 50.

In SSTF, requests are served as following

Next Served	Distance Traveled
50	0
45	5
60	15
70	10
80	10
90	10
25	65
20	5
10	10

Total Dist = 130

If Simple SCAN is used, requests are served as following

Next Served	Distance Traveled
50	0
60	10
70	10
80	10
90	10
45	65
[disk arm goes to 100, then to 45]	
25	20
20	5
10	10

Total Dist = 140

Extra Distance traveled in SSTF= 140-120=-10

If SCAN with LOOK is used, requests are served as following

Next Served	Distance Traveled
50	0
60	10
70	10
80	10
90	10
45	45
[disk arm comes back from 90]	
25	20
20	5
10	10

Total Dist= 120

Extra Distance traveled in SSTF=130 120=10

Q.5 (165)

The head movements are:

63 →	87	24 movements
87 →	92	5 movements
92 →	121	29 movements
121 →	191	70 movements
191→	10	0 movement
10 →	11	1 movement
11→	38	27 movements
38 →	47	9 movements

Total head movements = 165

4 FILE MANAGEMENT

4.1 INTRODUCTION

The file system consists of two distinct parts:

i) A collection of files, each storing related data, and

ii) A directory structure, which organizes and provides information about all the files in the system.

4.2 FILE CONCEPT

A file is a named collection of related information that is recorded on secondary storage. A file is a sequence of bits, lines or records whose meaning is defined by the file's creator and user.

4.2.1 File Attributes

A file has certain other attributes, which vary from one operating system to another, but typically consist of these:

(i) Name : The symbolic file name is the only information kept in human-readable form.

(ii) Type : This information is needed for those systems that support difference types.

(iii) Location: This information is a pointer to a device and to the location of the file on that device.

Size : The current size of the file (in bytes, words or blocks), and possibly the maximum allowed size are included in this attribute.

(v) Protection: Access-control information controls who can do reading, writing, executing, and so on.

(iv) Time, date and user identification : This information may be kept for

1) creation

2) last modification, and

3) last use.

These data can be useful for protection, security & usage monitoring.

4.2.2 File Operations

A file is an abstract data type.

1) Creating a file: Two steps are necessary to create a file.

a) Space in the file system must be found for the file

b) An entry for the new file must be made in the directory. The directory entry records the name of the file and the location in the file system.

2) Writing a file: To write a file, we make a system call specifying both name of the file and the information to be written to the file. Given the name of the file, the system searches the directory to find the location of the file. The system must keep a write pointer to the location in the file where the next write is to take place. The write pointer must be updated whenever a write occurs.

3) Reading a file: To read from a file, we use a system call that specifies the name of the file and where (in memory) the next block of the file should be put. Again, the directory is searched for the associated directory entry, and the system needs to keep a read pointer to the location in the file where the next read is to take place. Once the read has taken place, the read pointer is updated. Since, in general, a file is either being read or written, most systems keep

only one current-file-position pointer. Both the read and write operations use this same pointer, saving space and reducing the system complexity.

4) **Repositioning within a file:** The directory is searched for the appropriate entry, and the current-file-position is set to a given value. Repositioning within a file does not need to involve any actual I/O. This file operation is also known as a file seek.
5) **Deleting a file:** To delete a file, we search the directory for the named file. Having found the associated directory entry, we release all file space (so that it can be reused by other files) and erase the directory entry.
6) **Truncating a file:** There are occasions when the user wants the attributes of a file to remain the same, but wants to erase the contents of the file. Rather than forcing the user to delete the file and then recreate it, this function allows all attributes to remain unchanged (except for the length) but for the file to be reset to length zero. There are several pieces of information associated with an open file.
7) **File pointer:** On systems that do not include a file offset as part of the read and write system calls, the system must track the last read/write location as a current file-position pointer. This pointer is unique to each process operating on the file, and therefore must be kept separate from the on-disk file attributes.
8) **File open count:** As files are closed, the operating system must reuse its open-file table entries, or it could run out of space in the table. Because multiple processes may open a file, the system must wait for the last file to close before removing the open- file table entry, this counter tracks the number of opens and closes, and reaches zero on the last close. The system can then remove the entry.
9) **Disk location of the file:** Most File operations require the system to modify data within the file. The information needed to locate the file to disk is kept in memory to avoid having to read it from disk for each operation.

4.2.3 File Types

A common technique for implementing file types is to include the type as part of the file name. The name is split into two parts.
A name and an extension, usually separated by a period character.

4.2.4 File Structure

File types also may be used to indicate the internal structure of the file. As mentioned in Section. Source and object files have structures that match the expectations of the programs that read them.

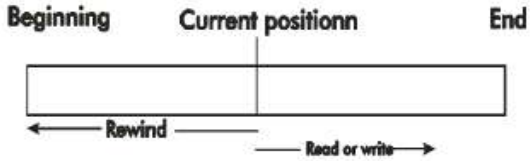
4.2.5 Internal File Structure

In either case, the file may be considered to be a sequence of blocks. All the basic I/O functions operate in terms of blocks. The conversion from logical records to physical blocks is a relatively simple software problem.

4.3 ACCESS METHODS

Files store information. When it is used, this information must be accessed and read into computer memory.

4.3.1 Sequential Access



The simplest access method is sequential access. Information in the file is processed in order, one record after the other.

4.3.2 Direct Access

Another method is direct access (or relative access). A file is made up of fixed- no particular order. The direct access method is based on a disk model of a file, since disks allow random access to any file block. For direct access, the file is viewed as a numbered sequence of blocks or records. A direct-access file allows arbitrary blocks to be read or written. Thus, we may read block 14, then read block 53, and then write block 7. There are no restrictions on the order of reading or writing for a direct-access file. The file operations must be modified to include the block number as a parameter.

Thus, we have read n, where n is the block number, rather than read next, and write n, rather than write next. An alternative approach is to retain read next and write next, as with sequential access, and to add an operation, position file to n, where n is the block number. Then, to affect a read n, we would position to n and then read next. The block number provided by the user to the operating system is normally a relative block number. A relative block number is an index relative to the beginning of the file. Thus, the first relative block of the file is 0, the next is 1, and so on, even though the actual absolute disk address of the block may be 14703 for the first block, and 3192 for the second. The use of relative block numbers allows the operating system to decide where the file should be placed (called the allocation problem, and helps to prevent the user from accessing portions of the file system that may not be part of his file. Some systems start their relative block numbers at 0; others start at 1. Given a logical record length L, a request for record N is turned into an I/O request for L bytes at location L + (N - 1) within the file.

4.3.3 Other Access Methods

Other access methods can be built on top of a direct-access method. These additional methods generally involve the construction of an index for the file. The index, like an index in the back of a book, contains pointers to the various blocks. To find an entry in the file, we first search the index, and then use the pointer to access the file directly and to find the desired entry. With large files, the index file itself may become too large to be kept in memory. One solution is to create an index for the index file. The primary index file would contain pointers to secondary index files, which would point to the actual data items. This organization is usually done in two parts.

i) The file system is broken into partitions, known as minidisks.

ii) Each partition contains information about files within it. This information is kept in entries in a device directory or volume table of contents. The device directory (more commonly known simply as a "directory") records information - such as name, location, size, and type - for all files on that partition.

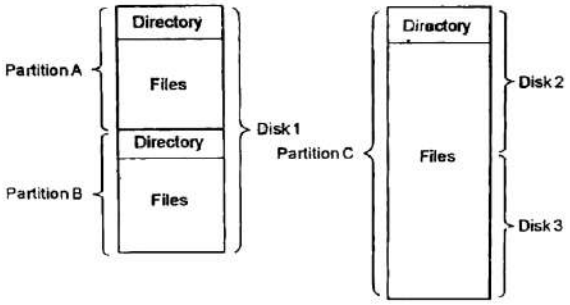


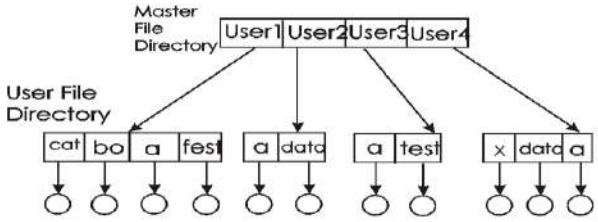
Fig. A typical file-system organization

4.4 SINGLE-LEVEL DIRECTORY

The simplest directory structure is the signal-level directory. All files are contained in the same directory, which is easy to support and understand.

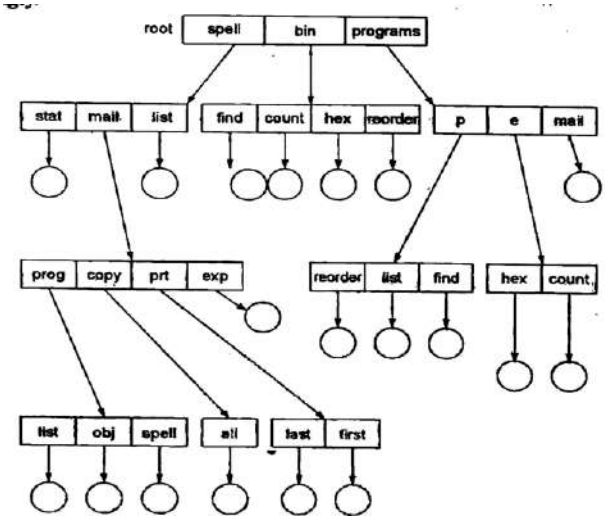
4.4.1 Two-Level Directory

The major disadvantage to a single-level directory is the confusion of file names between different users. The standard solution is to create a separate directory for each user.



4.4.2 Tree-Structure Directories

Once we have seen how to view a two-level directory as a two-level tree, the natural generalization is to extend the directory structure to a tree of arbitrary height. This generalization allows users to create their own sub-directories and to organize their files accordingly.



4.4.3 Protection

When information is kept in a computer system, a major concern is its protection from both physical damage (reliability) and improper access (protection). Reliability is generally provided by duplicate copies of files.

4.5 TYPES OF ACCESS

Protection mechanisms provide controlled access by limiting the types of file access that can be made; Access is permitted or denied depending on several factors, one of which is the type of access requested. Several different types of operations may be controlled:

Read : Read from the file.
Write : Write or rewrite the file.
Execute : Load the file into memory and execute it.
Append : Write new information at the end of the file.
Delete : Delete the file and free its space for possible reuse.
List : List the name and attributes of the file.

Other operations, such as renaming, copying, or editing the file, may also be controlled.

4.5.1 Access Lists and Groups

Many systems three classification of users in connection with each fie:

i) **Owner :** who created the file is the owner.

ii) **Groups :** A set of users who are sharing the file and need similar access is a group, or workgroup.

iii) **Universe :** All other users in the system constitute the universe.

4.5.2 Other Protection Approaches

There is another approach to the protection problem, which is to associate a password with each file. Just as access to the computer system itself is often controlled by a password access to each file can be controlled by a password.

4.6 ALLOCATION METHODS

Three major methods of allocating disk space are in wide use: contiguous, lined, and indexed. Each method has its advantages and disadvantages.

4.6.1 Contiguous Allocation

The contiguous allocation method requires each file to occupy a set of contiguous blocks on the disk. Disk addresses define a linear ordering on the disk.

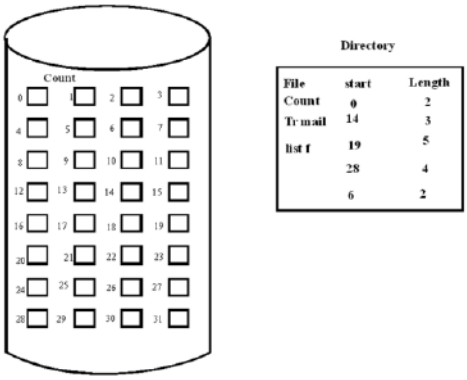


Fig. Contiguous allocation of disk space.

Contiguous allocation of a file is defined by the disk address and length (in block units) of the first block. If the file is n blocks long, and starts at location b, then it occupies blocks b, b + 1, b + 2, ... , b + n - 1. First fit and best fit are the most common strategies used to select a free hole from the set of available holes.

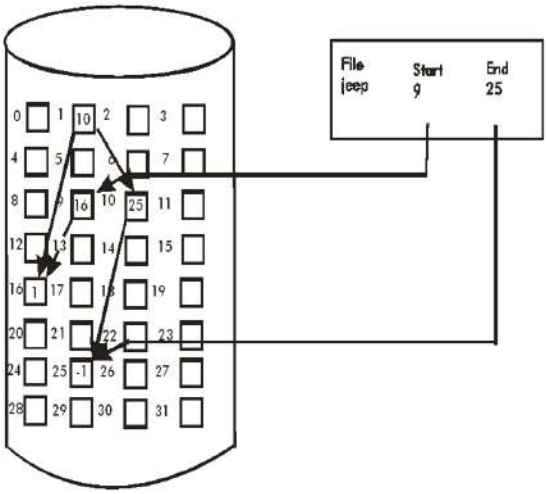
These algorithms suffer from the problem of external fragmentation. A major problem is determining how much space is needed for a file. When the file is created, the total amount of space it will need must be found and allocated. A file that grows slowly over a long period (months or years) must be allocate enough space for its final size, even though much of that space may be unused for long time. The file, therefore, has a large amount of internal fragmentation.

4.6.2 Linked Allocation

Linked allocation solves all problems of contiguous allocation. With linked allocation, each file is a linked list of disk blocks; the disk blocks may be scattered anywhere on the disk. To create a new file, we simply create a new entry in the directory. With linked allocation, each directory entry has a pointer to the first, disk blocks of the file. The pointer is initialized to nil (the end – of- list pointer value) to signify an empty file. A write to the file causes free blocks to be found via the free-space management system, and this new block is then written to, and is linked to the end of the file. To read a file, we simply read blocks by following the pointers from block to block. There is no external fragmentation with linked allocation.

4.6.3 Disadvantages of Linked allocation

i) The major problem is that it can be used effectively for only sequential-access files. To find the i^{th} blocks of a file, we must start at the beginning of that file, and follow the pointers until we get to the i^{th} block. Each access to a pointer requires a disk read, and sometimes a disk seek. Consequently, it is inefficient to support a direct-access capability for linked allocation files.



ii) The space required for the pointers. If a pointer requires 4 bytes out of a 512-byte block, then 0.787 percent of the disk is being used for pointers, rather than for information. Each file requires slightly more space than it otherwise would. The usual solution to this problem is to collect blocks into multiplies, called clusters, and to allocate the clusters rather than blocks. Another problem is reliability. Since the files are linked together by pointers scattered all over the disk, consider what would happen if a pointer were lost or damaged. An important variation on the linked allocation method is the use of a file-allocation table (FAT). This simple but efficient method of disk-space allocation is used by the MS-DOS and OS/2 operating system. A section of disk at the beginning of each partition is set aside to contain the number, the FAT is used much as is a linked list. The directory entry contains the block number of the first block of the file. The table entry indexed by that block number then contains the block number of the next block in the file. This chain continues until the last block which has a special end-of-file value as the table entry.

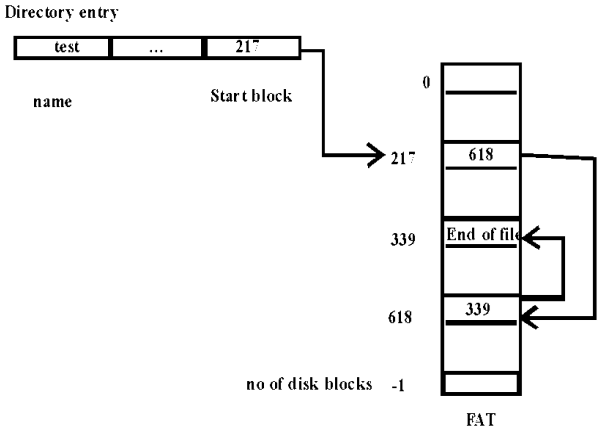
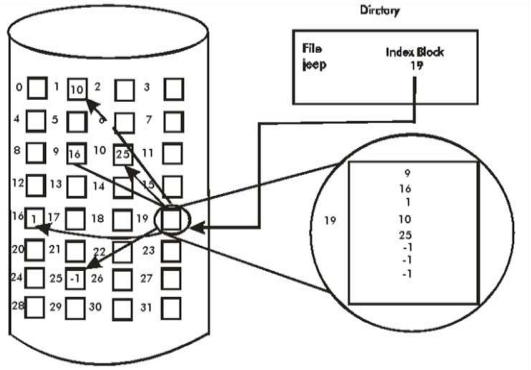


Fig. File-allocation table

4.6.4 Indexed Allocation

Linked allocation solves the external-fragmentation and size-declaration problems of contiguous allocation. However, in the absence of a FAT, linked allocation cannot support efficient direct access, since the pointers to the blocks are scattered with the blocks themselves all over the disk and need to be retrieved in order. Indexed allocation solves this problem by bringing all the pointers together into one location: the index block. Each file has its own index block, which is an array of disk-block addresses. The i^{th} entry in the index block points to the i^{th} block of the file. The directory contains the address of the index block. Indexed allocation supports direct access, without suffering from external fragmentation,

because any free block on the disk may satisfy a request for more space. Indexed allocation does suffer from wasted space. The pointer overhead of the index block is generally greater than the pointer overhead of linked allocation.



4.7 FREE-SPACE MANAGEMENT

Since there is only a limited amount of disk space, it is necessary to reuse the space from deleted files for new files, if possible. To keep track of free disk space, thus system maintains a free-space list.

4.7.1 Bit Vector

Frequently, the free-space list is implemented as a bit map or bit vector. Each block is represented by 1 bit. If the block is free the bits is 1; if the blocks is allocated, the bit is 0. Consider a disk where block 2, 3, 4, 5, 8, 9, 10, 11, 13, 17, 18, 25, 26, and 27 are free, and the rest of the blocks are allocated. The free-space bit map would be
0011110011111110000110000001110000….
The main advantage of this approach is that it is relatively simple and efficient to find the first free block, or n consecutive free blocks on the disk. The calculation of the block number (number of bits per word) x (number of 0-value words) + offset of first 1 bit).

4.7.2 Linked List

Another approach is to link together all the free disk blocks, keeping a pointer to the first free block in a special location on the disk and caching it in memory. The first block contains a pointer to the next free disk block, and so on.

4.7.3 Grouping

A modification of the free-list approach is to store the address of n free blocks in the first free block. The first n -1 of these blocks are actually free. The last block contains the addresses of another n free block, and so on. The importance of this implementation is that the addresses of a large number of free blocks can be found quickly, unlike in the standard linked-list approach.

4.7.4 Counting

Generally, several contiguous blocks may be allocated or freed simultaneously, particularly when space is allocated with the contiguous allocation algorithm or through clustering. Thus, rather than keeping a list of n free disk address, we can keep the address of the first free block and the number n of free contiguous blocks that follow the first block.

4.8 DIRECTORY IMPLEMENTATION

The selection of directory-allocation and directory-management algorithms has a large effect on the efficiency, performance, and reliability of the file system. Therefore, it is important to understand the tradeoffs involved in these algorithms.

4.8.1 Linear List

The simplest method of implementing a directory is to use a linear list of file names with pointers to the data blocks. This method is simple to program but is time-consuming to execute. To create a new file, we must first search the directory to be sure that no existing file has the same name. Then, we add a new entry at the end of the directory. To delete a file, we search the directory for the named file, then release the space allocated to it.
The real disadvantage of a linear list of directory entries is the linear search to find a file. A sorted list allows a binary search and decreases the average search time. However, the requirement that the list must be kept sorted may complicate creating and deleting files, since we may have to move substantial amounts of directory information to maintain a sorted directory.

4.8.2 Hash Table

Another data structure that has been used for a file directory is a hash table. In this method, a linear list stores the directory entries, but a hash data structure is also used. The hash table takes a value computed from the file name and returns a pointer to the file name in the linear list. Therefore, it can greatly decrease the directory search time.

4.8.3 Unix INODE Implementation

Disk Block Size
Owner
Location
Date and Time Stamp
Reference Count
Direct x Direct y . . .
Single Indirect
Double Indirect
Triple Indirect

Total File Size of system =
{ No of direct DBA's + (DB size/DBA) + $(DB\ size/DBA)^2$ + $(DB\ size/DBA)^3$ + ... }

DB Size = Disk Block Size
DBA = Disk Block Address

DB Size/DBA = No of block address possible to place in one disk block.

GATE QUESTIONS

Q.1 Using a larger block size in a fixed block size file system leads to
a) better disk throughput but poorer disk space utilization
b) better disk throughput and better disk space utilization
c) poorer disk throughput but better disk space utilization
d) poorer disk throughput at and poorer disk space utilization

[GATE-2003]

Q.2 A Unix-style i-node has 10 direct pointers and one single, one double and one triple indirect pointer. Disk block size is 1 Kbyte; disk block address is 32 bit, and 48-bit integers are used. What is the maximum possible file size?
a) 2^{24} byte b) 2^{32} byte
c) 2^{34} byte d) 2^{48} byte

[GATE-2004]

Q.3 The data blocks of a very large file in the Unix file system are allocated using
a) contiguous allocation
b) linked allocation
c) indexed allocation
d) an extension of indexed allocation

[GATE-2008]

Q.4 A file system with 300 Gbyte disk uses a file descriptor with 8 direct block addresses, 1 indirect block address and 1 doubly indirect block address. The size of each disk block is 128 byte and the size of each disk block address is 8 byte. The maximum possible file system is
a) 3 Kbyte
b) 35 Kbyte
c) 280 Kbyte
d) dependent on the size of the disk

[GATE-2012]

Q.5 Suppose a disk has 201 cylinders, numbered from 0 to 200. At some time the disk arm is at cylinder 100, and there is a queue of disk access requests for cylinders 30, 85, 90, 100, 105, 110, 135, and 145. If Shortest-Seek Time First [SSTF] is being used for scheduling the disk access, the request for cylinder 90 is serviced after servicing _____ number of requests. **[GATE–2014]**

Q.6 A FAT (file allocation table) based file system is being used and the total overhead of each entry in the FAT is 4 bytes in size. Given a 100×10^6 bytes disk on which the file system is stored and data block size is 10^3 bytes, the maximum size of the file that can be stored on this disk in units of 10^6 bytes is _____

[GATE–2014]

Q.7 in a file allocation system, which of the following allocation scheme (s) can be used if no external fragmentation is allowed ?
I. Contiguous II Linked
III. Indexed
a) I and III only
b) II only
c) III only
d) II and III only

[GATE–2017]

ANSWER KEY:

1	2	3	4	5	6	7
a	b	d	b	3	99.6	d

EXPLANATIONS

Q.1 **(a)**

Using larger block size makes disk utilization poorer as more space would be wasted for small data in a block. It may make throughput better as the number of blocks would decrease.

Q.2 **(b)**

Size of Disk Block = 1Kbyte
Disk Blocks address = 32bits,
but 48 bit integers are used for address
Therefore address size = 6 bytes
No of addresses per block = 1024/6 = 170.66
Therefore 170 ≈ 2^7 addresses per block can be stored
Maximum File Size = 10 Direct + 1 Single Indirect + 1 Double Indirect + 1 Triple Indirect
$= 10 + 2^7 + 22^7*22^7 + 22^7*22^7*22^7$
& apporx; 2^{22} Blocks
Since each block is of size 2^{10}
Maximum files size = $2^{22} * 2^{22} = 2^{32}$

Q.3 **(d)**

Indexed allocation is used for the efficient direct access good use of disk space. However, it cannot allocate very large files. Thus, the data blocks of very large file in UNIX system are allocated using an extension of indexed allocation or EXT2 file system.

Q.4 **(b)**

8 direct block pointers
1 indirect block pointer
1 doubly indirect block pointer
Block size =128 = 2^7 byte
Disc block address = 8 byte = 2^3 byte
Number of pointers in 1 disc block

$$\frac{2^7}{2^3} = 2^4$$

Maximum size of file
$= (8 \times 2^7 + 2^4 \times 2^7 + 2^4 \times 2^4 \times 2^7)$

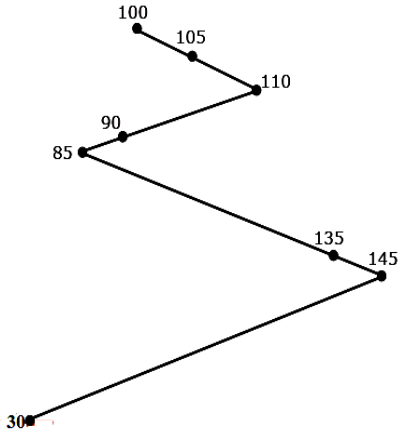
$$= \left(2^3 \times 2^7 \times 2^{11} \times 2^{15}\right)$$

$$= \left(2^{10} + 2^{11} + 2^{15}\right)$$

=1Kbyte+2Kbyte+32Kbyte=35Kbyte

Q.5 **(3)**

Request for cylinder is served after serving 3 requests (100,105 & 110)



Q.6 (99.6)

Number of entries in the FAT = Disk Capacity/Block size= 108/103 = 105
Total space consumed by FAT = 105 * 4 B = 0.4 * 106 B
Maximum size of file that can be stored=100*106–0.4*106=99.6*106 B

Q.7 (d)

Linked and indexed allocated are non contiguous so, they will not suffer from external fragmentation

ASSIGNMENT QUESTIONS

Q.1 Virtual memory is
a) An externally large main memory
b) An externally large secondary memory
c) An illusion of an extremely large memory
d) A type of memory used in super computers

Q.2 Spatial localities refers to the problem that once a location is referenced
a) It will not be referenced again
b) It will be referenced again
c) A nearby location will be referenced soon
d) None of the above

Q.3 Which of the following is an example of a SPOOLED device?
a) The terminal used to enter the input data for a program being executed.
b) The secondary memory device in a virtual memory system
c) A line printer used to print the output of a number of jobs.
d) None of the above.

Q.4 Page fault occurs when
a) The page is corrupted by application software
b) The page is main memory
c) The page is not in main memory
d) One tries to divide a number by 0

Q.5 Overlay is
a) A part of the operating system
b) A specific memory location
c) A single contiguous memory that used in the olden days for Running large programs by swapping.
d) Overloading the system with many user files

Q.6 Determine the number of page faults when references to pages occur in the order -1, 2, 4, 5, 2, 1, 2, 4. Assume that the main memory can accommodate 3 pages and the main memory already has the pages 1 and 2, with page 1 having been brought earlier than page 2. (Assume LRU algorithm is used)
a) 3 b) 5
c) 4 d) None of the above

Q.7 Concurrent processes are processes that
a) Do not overlap in time
b) Overlap in time
c) Are executed by a processor at the same time
d) None of the above

Q.8 The page replacement policy that sometimes leads to more page faults when the is size of the memory is increased is
a) FIFO
b) LRU
c) No such policy exists
d) None of the above

Q.9 The only state transition that is initiated by the user process itself is
a) Block b) dispatch
c) Wakeup d) None of the above

Q.10 Working set (t, k) at an instant of time, t, is the set of
a) K future reference that the operating system will make
b) Future reference that the operating system will make in the next 'k' time units
c) K reference with high frequency
d) Pages that have been referenced in the last k time units

Q.11 Fragmentation is
a) dividing the secondary memory into equal sized fragments
b) dividing the main memory into equal-sized fragments
c) Fragments of memory words used in a page
d) Fragments of memory words unused in a page

Q.12 Which of the following are real-time systems?
a) An on-line railway reservation system
b) A process control system
c) Aircraft control system
d) Payroll processing system

Q.13 Dijkstra's banking algorithm in a operating system solves the problem of
a) Deadlock avoidance
b) Deadlock recovery
c) Mutual exclusion
d) Context switching

Q.14 In page memory systems, if the page size is increased, then the internal fragmentation generally
a) Becomes less
b) Becomes more
c) Remains constant
d)none of the above

Q.15 An operating system contains 3 user processes each requiring 2 units of resource R. The minimum number of units of R such that no deadlock will ever occur is
a) 3 b)4
c) 5 d) 6

Q.16 Critical region is
a) a part of operating system which is not allowed to be accessed by any process
b) a set of instructions that access common shared resource which exclude one another in time
c) the portion of the main memory which can be accessed only by one process at a time
d) None of the above

Q.17 Kernel is
a) Considered as the critical part of the operating system
b) The software which monitors the operating system
c) The set of primitive function upon which the rest of operating system functions are built up.
d) None of the above

Q.18 With a single resource, deadlock occurs
a) If there are more than two processes competing for that resource
b) If there are only two processes competing for that resource
c) If there is a single process competing for that resource
d) None of the above

Q.19 Necessary conditions for deadlock are
a) Non-pre-emption & circular wait
b) Mutual exclusion & partial allocation
c) Both (a) and (b)
d) None of the above

Q.20 In a time-sharing operating system, when the time slot given to a process is completed, the process goes from the RUNNING state to the
a) BLOCKED state
b) READY state
c) SUSPENDED state
d) TERMINATED state

Q.21 At a particular time, the value of a counting semaphore is 10. It will become 7 after
a) 3 V operation
b) 3 P operation
c) 3 V operation and 3 P operations
d) 13 P operation & 10 v operation

Q.22 Supervisor call
a) Is a call made by the supervisor of the system?
b) Is a call with control functions?
c) Are privileged calls that are used to perform resource management functions, which are controlled by the operating system.
d) Is a call made by someone working in root directory?

Q.23 Semaphores are used to solve the problem of
a) Race condition
b) Process synchronization
c) Mutual exclusion
d) None of the above

Q.24 If the property of locality of reference is well pronounced in a program
a) The number of page faults will be more
b) The number of page faults will be less
c) The number of page faults will remains the same
d) Execution will be faster

Q.25 At a particular time of computation, the value of a counting semaphore is 7. Then 20p operations and 'x' v operation were completed on this semaphore. If the final value of the semaphore is 5, x will be
a) 15 b) 22
c) 18 d) 13

Q.26 Pre-emptive scheduling, is the strategy of temporarily suspending a running process
a) Before the CPU time slice expires
b) To allow starving processes to run
c) When it requests I/O
d) None of the above

Q.27 Mutual exclusion problem occurs
a) Between two disjoint processes that do not interact
b) Among process that share resource
c) Among process that do not use the same resource
d) None of the above

Q.28 Sector interleaving in disks is done by
a) The disk manufacturer
b) The disk controller code
c) The operating system
d) None of the above

Q.29 Memory protection is of no use in a
a) Single user system
b) Non-multiprogramming system
c) Non-multitasking system
d) None of the above

Q.30 Some computer system support dual mode operation-the user mode and the supervisor or monitor mode. These refer to the modes
a) By which user program handle their data
b) By which the operating system executes user programs
c) In which the processor and the associated hardware operate.
d) Of memory access

Q.31 Disk scheduling involves deciding
a) which disk should be accessed next
b) the order in which disk access requests must be serviced.
c) the physical location where files should be accessed in the disk
d) none of the above

Q.32 A computer system has 6 tape drives, with 'n' process is competing for them. Each process may need 3 drives. The maximum value of 'n' for which the system is guaranteed to be deadlock free is.
a) 2 b) 3
c) 4 d) 1

Q.33 Dirty bit is used to show the
a) page with corrupted data

b) wrong page in the memory
c) page that is modified after being loaded into cache memory
d) page that is less frequently accessed

Q.34 Fence register is used for
a) CPU protection
b) Memory protection
c) File protection
d) all of the above

Q.35 Which of the following is the service not supported by the operating system?
a) Protection b) Accounting
c) Compilation d)I/O operation

Q.36 The first-fit, best-fit and the worst-fit algorithm can be used for
a) contiguous allocation of memory
b) linked allocation of memory
c) indexed allocation of memory
d) all of the above.

Q.37 Which of the following are single-user operating systems?
a) MS-DOS b) UNIX
c) XENIX d) OS/2

Q.38 In Round Robin CPU scheduling, as the time quantum is increased, the average turnaround time
a) Increases
b) Decreases
c) Remains constant
d) Varies irregularly

Q.39 In a multiprogramming environment
a) the processor executes more than one process at a time
b) the programs are developed by more than one person
c) more than one process resides in the memory
d) a single user can execute many programs at the same time.

Q.40 Which of the following are true?
a) A re-entrant procedure can be called any number of times.
b) A re-entrant procedure can be called even before the procedure has not returned from its previous call.
c) Re-entrant procedures cannot be called recursively.
d) Re-entrant procedures can be called recursively.

Q.41 In a paged memory, the page hit ratio is 0.35. The time required to access a page in secondary memory is equal to 100 ns. The time required to access a page in primary memory in 10 ns. The average time required to access a page is
a) 3.0 ns b) 68.0 ns
c) 68.5 ns d) 75.5 ns

Q.42 A state is safe if the system can allocate resources to each process (up to its maximum) in some order and still avoid deadlock.
Which of the following are true?
a) Deadlock state is unsafe.
b) Unsafe state may lead to a deadlock situation.
c) Unsafe state must lead to a deadlock situation.
d) Deadlock state is a subset of a unsafe state.

Q.43 The size of the virtual memory depends on the size of
a) data-bus b) main memory
c) address bus d) none of the above

Q.44 In a multi-user operating system, 20 requests are made to use a particular resource per hour, on an average. The probability that no request are made in 45 minutes is
a) e^{-15} b) e^{-5}
c) $1 - e^{-5}$ d) $1 - e^{-10}$

Q.45 In which of the scheduling policies does context switching never take place during process execution?

a) Round-robin
b) Non Pre-emptive Shortest job first
c) Pre-emptive
d) First-cum-first-served

Q.46 In which of the following directory systems, is it possible to have multiple complete paths for a file, starting from the root directory?
a) Single level directory
b) Two level directory
c) Tree structured directory
d) Acyclic graph directory

Q.47 Suppose that a process is 'BLOCKED' state waiting for some I/O service. When the service is completed, it goes to the
a) RUNNING state
b) READY state
c) SUSPENDED state
d) TERMINATED state

Q.48 In a system that does not support swapping
a) the compiler normally binds symbolic address (variables) to reloadable addresses.
b) the compiler normally binds symbolic address to physical addresses.
c) the loader binds reloadable addresses to physical addresses.
d) binding of symbolic addresses to physical addresses normally takes place during execution.

Q.49 To obtain better memory utilization, dynamic loading is used. With dynamic loading, a routine is not loaded until it is called for. For implementing dynamic loading,
a) special support from hardware is essential.
b) special support from operating system is essential.
c) Special support from both hardware and operating system are essential.
d) user programme can implement dynamic loading without any special support from the operating system or the hardware.

Q.50 Which of the following is true?
a) The linkage editor is used to edit programs which have to be later linked together
b) The linkage editor links object modules during compilation or assembling
c) The linkage editor links object modules and resolves external references between them before loading.
d) The linkage editor resolved external references between the object modules during execution time.

Q.51 Which of the following is true?
a) Overlays are used to increase the size of physical memory.
b) Overlays are used to increase the logical address space
c) When overlays are used, the size of process is not limited to the size of physical memory.
d) Overlays are used whenever the physical address space is smaller than the logical address space.

The next 5 questions are based on the following information.

Consider a set of 5 processes whose arrival time, CPU time needed and the priority are given below:

Process Priority	Arrival Time (in ms)	CPU Time Needed (in ms)	Priority
P1	0	10	5
P2	0	5	2
P3	2	3	1
P4	5	20	4
P5	10	2	3

Note: Smaller the number, higher the priority.

Q.52 If the CPU scheduling policy is FCFS, the average waiting time will be
a) 12.8 ms b) 8 ms
c) 16 ms d) None of the above

Q.53 If the CPU scheduling policy is SJF, the average waiting time (without pre-emption) will be
a) 12.8 ms b) 6.8 ms
c) 17 ms d) None of the above

Q.54 If the CPU scheduling policy is SJF with pre-emption, the average waiting time will be
a) 8 ms b) 14 ms
c) 5.6 ms d) none of the above

Q.55 If the CPU scheduling policy is priority scheduling without pre-emption, the average waiting time will be
a) 12.8 ms b) 11.8 ms
c) 10.8 ms d) none of above

Q.56 If the CPU scheduling policy is priority scheduling with pre-emption, the average waiting time will be
a) 19 ms b) 7.6 ms
c) 8 ms d) None of above

Q.57 In partitioned memory allocation scheme, the
a) best fit algorithm is always better than the first fit algorithm.
b) first fit algorithm is always better than the best fit algorithm.
c) superiority of the first fit and best-fit algorithms depends on the sequence of the memory request.
d) None of above

Q.58 Cascading termination refers to termination of the all child processes before the parents terminates
a) Normally
b) abnormally
c) Normally or abnormally
d) None of the above

Q.59 For implementing of multi programming operating system
a) Special support from processor is essential
b) Special support from processor is not essential
c) Cache memory must be available
d) More than one processor must be available.

Q.60 Consider a system having 'm' resources of the same type. These resources are shared by 3 processes A, B, C, Which have peak time demands of 3, 4, and 6 respectively. The minimum value of 'm' that ensures that deadlock will never occur is
a) 11 b) 12
c) 13 d) 14

Q.61 A system has 3 processes sharing 4 resources. If each process needs a maximum of 2 units, then deadlock
a) Can never occur
b) may occur
b) Has to occur
d) none of the above

Q.62 'm' processes share 'n' resources of the same type. The maximum needs of each process don't exceed 'n' and the sum of all their maximum needs is always less than m + n. In this set up deadlock
a) can never occur
b) may occur
c) has to occur
d) None of the above

Q.63 A process refers to 5 pages, A, B, C, D and E in the order – A; B; C; D; A; B; E; A; B; C; D; E. If the page replacement algorithm if FIFO, the number of pages which transfer with an empty internal store of 3 frame is
a) 8 b) 10
c) 9 d) 7

Q.64 Distributed system should
a) Meet prescribed time constraints
b) Aim better resource sharing
c) Aim better system utilization
d) Aim low system overhead

Q.65 The main function of shared memory is to
a) Use primary memory efficientl
b) Do intra process communication
c) Aim better system utilization
d) Aim low system overhead

Q.66 Which of the following is the most suitable scheduling scheme in a real–time operating system?
a) round-robin
b) first-come-first-served
c) Pre-emptive scheduling
d) random scheduling

Q.67 In Question number 63, if the number of available page frames is increased to 4 then the number of page transfers
a) decreases
b) increases
c) remains the same
d) none of the above

Q.68 'Aging' is
a) Keeping track of cache contents.
b) Keeping track of what pages are currently residing in the memory.
c) Keeping track of how many times a given page is referenced.
d) Increasing the priority of jobs to ensure termination in a finite time.

Q.69 If there are 32 segments, each of size 1 Kbytes, then the logical address should have
a) 13 bits b) 14 bits
c) 15 bits d) 16 bits

Q.70 Disk requests come to a disk driver for cylinders in the order 10, 22, 20, 2, 40, 6 and 38, at a time when the disk drive is reading from cylinder 20. The seek time is 6 ms per cylinder. The total seek time, if the disk arm scheduling algorithm is first-come-first-served is
a) 360 ms b) 850 ms
c) 900 ms d) None of the above

Q.71 In question 70, if the scheduling algorithm is the closest cylinder next, then the total seek time will b
a) 360 ms b) 876 ms
c) 850 ms d) 900 ms

Q.72 Certain moving arm disk storage with one head has following specification Number of tracks / recording surface = 200 Disk rotation speed = 2400 rpm Track storage capacity = 62500 bits The average latency time (assume that the head can move from one track to another only by traversing the entire track) is
a) 2.5 s b) 2.9 s
c) 3.1 s d) 3.6 s

Q.73 Memory protection is normally done by the
a) Processor and the associated hardware
b) Operating system
c) Compiler
d) User program

Q.74 Which of the following scheduling algorithms gives minimum average waiting time?
a) FCFS b) SJF
c) Round-robin d) Priority

Q.75 In question number 72, the transfer rate will be
a)2.5 Mbits/s b) 4.25 Mbits/s
c)1.5 Mbits/s d)3.75 Mbits/s

Q.76 In a paged segment scheme of memory management, the segment table itself must have a page table because
a) the segment table is often too large to fit in one page

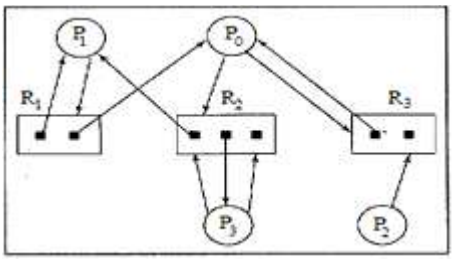
b) each segment is spread over a number of pages
c) segment table point to page tables and not to the physical location of the segment
d) the processor's description base register point to a page table

Q.77 Which of the following scheduling policy is well suited for a time-shared? Operating system?
a) FIFO
b) Pre-emptive scheduling
c) Round Robin
d) both FIFO and Pre-emptive

Q.78 Which of the following scheduling policy is well suited for a time-shared operating system?
a) Shortest job first
b) Round robin
c) First-come-first-serve
d) Elevator

Q.79 The address sequence generated by tracing a particular executing in a pure demand paging system with 100 records per page, with 1 free main memory frame is recorded as follows. What is the number of page faults?
0100,0200,0430,0499,0510,0530,056, 0120,0220,0240,0260,0320,0370.
a) 13 b) 8
c) 7 d) 10

Q.80 A computer system has 4 k word cache organized in a block-set-associative manner, with 4 blocks per set, 64 words per block. The number of bits in the SET and WORDS fields of the main memory address format is



a) 15, 4 b) 6, 4
c) 7, 2 d) 4, 6

Q.81 A computer installation has 1000 k of main memory. The jobs arrive and finish in the following sequence.
Job 1 requiring 200 K arrives
Job 2 requiring 350 K arrives
Job 3 requiring 300 K arrives
Job 1 finishes
Job 4 requiring 120 K arrives
Job 5 requiring 150 K arrives
Job 6 requiring 80 K arrives
Among best fit and first fit, which performs better for this sequence?
a) First fit
b) Best fit
c) Both perform the same
d) None of the above

Q.82 A memory page containing a heavily used variable that was initialized very early and is in constant use is removed. When the page replacement algorithm used is
a) LRU b) FIFO
c) LFU d) None of the above

Q.83 Consider the following heap. The sequence of request for blocks of sizes 300, 25, 125, 50 can be satisfied if we use
a) Either first fit or best fit policy
b) first fit but not best fit
c) Best fit but not first fit
d) None of the above

Q.84 Consider the resources allocation graph. This system is in a deadlock state. This remark is
a) True
b) false
c) Impossible to determine
d) unpredictable

Q.85 Which of the following is a safe sequence?
a) P_0, P_1, P_2, P_3 b) P_1, P_0, P_3, P_2
c) P_2, P_0, P_1, P_3 d) None of the above

Q.86 A demand paging system, with page table had held in registers, takes 5 ms to service a page fault if an empty page is available, or if the page to be replaced in not dirty. It takes 15 ms if the replace page is dirty. Memory access time is 1µs. Assume we want an effective access time of 2µs. And that the page to be replaced is dirty 60% of the time. What is the approximate maximum acceptable page fault rate to meet this access time requirement?
a) 0.1 % b) 1.0 %
c) 2.5 % d) 0.01 %

Q.87 Consider a computer with 8 Mbytes if main memory and a 128 k cache. The cache block size is 4 K. It uses a direct mapping scheme for cache management. How many different main memory can map onto a given a given physical cache block?
a) 2048 b) 256
c) 64 d) None of the above

Q.88 Which of the following applications are well suited for batch processing?
a) Process control
b) Video game control
c) Preparing pay bills of employees
d) Preparing mailing address

Q.89 Locality of reference implies that the page reference being made by a process
a) will always to be the page used in the previous page reference.
b) is likely to be one of the pages used in the last few page reference.
c) will always be one of the pages existing in memory
d) will always leads to page fault.

Q.90 The correct matching for the following pairs
a) Disk scheduling 1)Round robin
b) Batch processing 2) SCAN
c) Time sharing 3) LIFO
d) Interrupt processing 4) FIFO
Is:
a) A–3,B-4,C-2,D-1
b) A–4,B-3,C-2,D-1
c) A–2,B-4,C-1, D-3
d) A–2,B-1,C-4,D-3

Q.91 Thrashing
a) reduces page I/O
b) decreases the degree of multiprogramming
c) implies excessive page I/O
d) improves the system performance

Q.92 Dirty bit for a page in a page table
a) helps avoid unnecessary writes on a paging devise
b) helps maintain LRU information
c) allows only read on a page
d) None of the above

Q.93 Each process P_i, i = 1, 2, 3 ...., 9 is coded as follows
Repeat
P (mutex)
{ critical section }
V (mutex)
Forever
The code for P10 is identical except that it uses V (mutex) instead of P (mutex). What is the largest number of processes that can be inside the critical section at any of the movement?
a) 1 b) 2
c) 3 d) none of the above

Q.94 When an interrupt occurs, an operating system
a) ignores the interrupt
b) always change the state of the interrupted process after processing the interrupt
c) always resumes execution of the interrupted process after processing the interrupt
d) may change the state of the interrupted process to "blocked" and schedule another process.

Q.95 On a system with n CPUs, what is the maximum number of processes that can be in the ready, run and blocked states ?

a) $\frac{n}{2}$ b) $\frac{n}{3}$
c) n d) None.

Q.96 Given system with n processes, how many possible ways can those processes be scheduled ?

a) n b) $\angle\underline{n}$
c) $\frac{n}{2}$ d) None.

Q.97 On a system with 2^{24} bytes of memory and fixed partitions, all of size 65,536 bytes, what is the minimum number of bits needed in an entry in the process table to record the partition to which a process has been allocated ?

a) 2 b) 4
c) 6 d) 8

Q.98 On a system with 2^{24} bytes of memory and fixed partitions, all of size 65, 536 bytes, how many bits must the limit register have

a) 4 b) 8
c) 12 d) 16.

Q.99 On a system using fixed partitions with sizes 2^{16}, 2^{24}, and 2^{32}, how many bits must the limit register have?

a) 16 b) 24
c) 32 d) None.

Q.100 With a single resource, deadlock occurs

a) if there are more than two processes competing for that resource
b) if there are only two process competing for that resource
c) if there is a single process competing for that resource
d) none of the above.

Q.101 Necessary conditions for deadlock are

a) non-preemption and circular wait
b) mutual exclusion and partial allocation
c) both a) and b)
d) none of the above.

Q.102 In a time-sharing operating system, when the time slot given to a process is completed, the process .goes from the RUNNING state to the

a) BLOCKED state
b) READY state
c) SUSPENDED state
d) TERMINATED state.

Q.103 A sequential access file has fixed-size 15-byte records. Assuming the first record is record 1, the first byte of record 5 will be at what logical location?

a) 41 b) 51
c) 61 d) None.

Q.104 A direct access file has fixed-size 15 byte records. Assuming the first record is record 1, the first byte of record 5 will be at what logical location?

a) 41 b) 51
c) 61 d) none.

Q.105 A direct or sequential access file has fixed-size S-byte records. The first byte of record N will start at what logical location ?

a)[(N+ 1)*S]+1 b)[(N-1)*S]-1
c) (N-1)* S+1] d) None.

Q.106 A program has just read the tenth record in a sequential access file. It next wants to read the sixth record. How many records must the program read to input the sixth ?

a) 4 b) 5
c) 6 d) None.

Q.107 On a system using a free hole array, and that has K physical blocks, what is the maximum length of a free hole array? The minimum length?
a) 0 b) 1
c) 2 d) 3.

Q.108 At a particular time, the value of a counting semaphore is 10. It will become 7 after
a) 3 V operations
b) 3 P operations
c) 5V operations and 2P operations
d) 13P operations and 10V operations.

Data of Q.109- Q.113 are given :
One a system using round-robin scheduling, let s represent the time needed to perform a process switch, q represent the round-robin time quantum, and r represent the average time a process runs before blocking on I/0. CPU efficiency is

Q.109 q = ∞
a) r b) $\frac{1}{r+s}$
c) $\frac{r}{r+s}$ d) None

Q.110 q > r
a) r b) $\frac{1}{r+s}$
c) $\frac{r}{r+s}$ d) None.

Q.111 s < q < r
a) q b) $\frac{1}{q+s}$
c) $\frac{q}{q+s}$ d) None.

Q.112 s = q = r
a) $\frac{1}{3}$ b) $\frac{1}{2}$
c) $\frac{1}{5}$ d) None

Q.113 q nearly 0
a) 0 b) 1
c) 1/2 d) None

Q.114 On a system using multilevel feedback queues, a totally CPU-bound process requires 40 seconds to execute. If the first queue uses a time quantum of 2 and at each level the time quantum increases by 5 time-units, how many times will the job be interrupted ?
a) 2 b) 3
c) 4 d) None

Q.115 On a system using non-preemptive scheduling, processes with expected run times of 5, 18, 9 and 12 are in the ready queue. In what order should they be run to minimize wait time?
a) 5, 12, 9, 18 b) 5, 9, 12, 18
c) 12, 18, 9, 5 d) None.

Q.116 On a computer system with a single 32-bit 100 MHz bus (10ns cycle), the disk controller uses DMA to transfer data to/from memory at a rate of 40Mb per second. Assume the computer fetches and executes one 32-bit instruction every cycle when no cycles are stolen. By what percentage will the disk controller slow down instruction execution ?
a) 5% b) 10%
c) 15% d) 20%.

Q.117 A computer system that uses memory-mapped I/O has a 16-bit address space Addresses with ones in the two most significant bits refer to devices. What is the maximum amount of memory that can be referenced in such a system?
a) 2^7 b) 2^9
c)2^{11} d) 2^{14}

Q.118 A disk file contains 32-byte fixed-size records. Physical 1/0 is in terms of 512 byte blocks that are stored in an operating system buffer. If a process sequentially reads though a file's records, what

percentage of the read requests will result in I/O operations?
a) 5.25% b) 6.25%
c) 7.25% d) None.

Data of Questions Q.119- Q.120 are given:
On a system using a disk cache, the mean access time is dependent on the mean cache access time, the mean disk access time, and the hit rate. For the following, what is the mean access time?

Q.119 Cache:1ms; disk;100ms; hit rate: 25%
a) 50 b) 70
c) 75.25 d) None

Q.120 Cache:1ms;disk:100ms;
hit rate:50%
a) 30 b) 40
c) 50.50 d) None.

Q.121 On a simple paging system with 2^{24} bytes of physical memory, 256 pages of logical address space, and a page size of 2^{10} bytes, No. of bits are in a logical address ?
a) 10 b) 12
c) 16 d) 18.

Q.122 On a simple paging system with 2^{24} bytes of physical memory, 256 pages of logical address space, and a page size of 2^{10} bytes, No. of bytes are in a page frame?
a) 2^6 bytes b) 2^8 bytes
c) 2^{10} bytes d) None.

Q.123 On a simple paging system with 24^{24} bytes of physical memory, 256 pages of logical address space, and a page size of 2^{10} bytes, No. of bits in the physical address specify the page frame.
a) 10 b) 12
c) 14 d) None.

Q.124 On a simple paging system with 2^{24} bytes of physical memory, 256 pages of logical address space, and a page size of 2^{10} bytes, No. of entries are in the page table (how long is the page table)?
a) 156 entries b) 186 entries
c) 226 entries d) 256 entries.

Q.125 On a simple paging system with 2^{24} bytes of physical memory, 256 pages of logical address space, and a page size of 2^{10} bytes, No. of bits are needed to store an entry in the page table (how wide is the page table)?
a) 4 b) 7
c) 9 d) 14.

Q.126 At a particular time of computation, the value of a counting semaphore is 7. Then 20 P operations and 'x'. V operations were completed on this semaphore. If the final value of the semaphore is 5, x will be
a) 15 b) 22
c) 18 d) 13.

Q.127 A computer system has 6 tape drives, with 'n' processes competing for them. Each process may need 3 tape drives. The maximum, value of 'n' for which the system is guaranteed to be deadlock free is
a) 2 b) 3
c) 4 d) 1.

Q.128 In a paged memory, the page hit ratio is 0,35. The time required to access a page in secondary memory is equal to 100 ns. The time required to access a page in primary memory is 10 ns. The average time required to access a page is
a) 3.0 ns b) 68.0 ns
c) 68.5 ns d) 78.5 ns.

Q.129 In a multiuser operating system, 20 requests are made to use a particular resource per hour, on an average. The probability that no requests are made in 45 minutes is

a) e^{-15} b) e^{-5}
c) $1-e^{-5}$ d) $1-e^{-10}$

Data of Questions Q.130-134 are given:
For the processes listed in Table, the average turnaround time is

Process Scheduling Data

Process	Arrival time	Processing time
A	0.000	3
B	1.001	6
C	4.001	4
D	6.001	2

Q.130 In first-come first-served is
a) 5.25 b) 6.25
c) 7.25 d) None.

Q.131 In shortest job first is
a) 5.25 b) 6.75
c) 7.25 d) None.

Q.132 In shortest remaining time is
a) 5.25 b) 6.25
c) 6.75 d) None.

Q.133 In round robin (quantum=2) is
a) 5.25 b) 6.25
c) 7.25 d) 8.25.

Q.134 In round robin (quantum = 1) is
a) 4.50 b) 5.50
c) 6.50 d) 8.50.

Data of Questions Q.135- Q.139 are given :
For the processes listed above, wait time using:

Q.135 First-come first-served is
a) 0, 2, 3, 7 b) 0, 2, 5, 7
c) 0, 2, 5, 9 d) None.

Q.136 Shortest job first is
a) 0, 2, 3,7 b) 0, 2, 7,3
c) 0, 2, 3, 5 d) None.

Q.137 Shortest remaining time is
a) 0, 8, 8, 0 b) 0, 8, 0, 2
c) 0, 8, 3, 2 d) None.

Q.138 Round robin (quantum = 2) is
a) 1, 6, 7, 3 b) 2, 6, 7, 3
c) 2, 6, 3, 7 d) None.

Q.139 Round robin (quantum = 1) is
a) 1, 8, 2, 4 b) 1, 8, 6, 4
c) 2, 8, 3, 4 d) None.

Data of Questions Q.140- Q.142 are given :
For the processes listed in Table, turn around time is

Process Scheduling Data

Process	Arrival time	Processing time
A	0.000	4
B	2.001	7
C	3.001	2
D	3.001	2

Q.140 In first-come first-served
a) 3, 4, 5, 7 b) 3, 5, 7, 9
c) 4, 9, 10, 12 d) None.

Q.141 In shortest job first
a) 2, 13, 3, 5 b) 4, 13, 3, 5
c) 3, 13, 3, 15 d) None

Q.142 In shortest remaining time
a) 3, 4, 3, 7 b) 4, 13, 3, 5
c) 1, 3, 5, 7 d) None

Q.143 In round robin (quantum = 2)
a) 3, 10, 5, 7 b) 3, 10, 7, 4
c) 4, 13, 5, 7 d) None

Q.144 In round robin (quantum = 1)
a) 3, 10, 17,3 b) 5, 13, 6, 7
c) 5, 10, 3, 7 d) None

Data of Questions. Q.145-Q.150 are given :
For the processes listed above, average wait time is

Q.145 In first-come first-served
a) 2 b) 3
c) 4 d) 5.

Q.146 In shortest job first
a) 1.5 b) 2
c) 2.5 d) None.

Q.147 In shortest remaining time
a) 1.5 b) 2
c) 2.5 d) None.

Q.148 In round robin (quantum = 2)
a) 1.50 b) 2
c) 2.50 d) 3.50.

Q.149 In round robin (quantum = 1)
a) 2 b) 3
c) 4 d) 5.

Q.150 On a system using a disk cache, the mean access time is 41.2 ms, the mean cache access time is 2 ms, the mean disk access time is 100 ms, and the system has 8 Mb of cache memory. For each doubling of the amount of memory, the miss rate is halved. How much memory must be added to reduce the mean access time to 20 ms?
a) 10.8 b) 11.8
c) 12.8 d) None.

Data of Questions. Q.151- Q.155 are given :
On a disk with 1000 cylinders, numbers 0 to 999, compute the number of tracks the disk arm must move to satisfy all the requests in the disk queue. Assume the last request serviced was at track 345 and the head is moving toward track 0. The queue in FIFO order contains requests for the following tracks : 123, 874, 692, 475, 105, 376. Perform the computation for the following scheduling algorithms :

Q.151 FIFO
a) 1013 b) 1713
c) 2013 d) None.

Q.152 SSTF
a) 1198 b) 1298
c) 1398 d) None.

Q.153 SCAN
a) 1019 b) 1119
c) 1219 d) None.

Q.154 LOOK
a) 809 b) 909
c) 1009 d) None.

Q.155 C-SCAN
a) 1267 b) 1567
c) 1967 d) None.

Q.156 C-LOOK
a) 1107 b) 1207
c) 1507 d) None.

Data of Questions. Q.157- Q.160 are given :
A page size of 2000 bytes, and the following page table in given

In/out	Frame
in	20
out	22
in	200
in	150
out	30
out	50
in	120
in	101

Which of the following virtual addresses generate a page fault ?

Q.157 10451
a) 5421 b) Fault
c) 3421 d) None.

Q.158 5421
a) 301, 321 b) 401, 421
c) 501, 521 d) None.

Q.159 14123
a) 202, 123 b) Fault
c) 102, 113 d) None.

Q.160 9156
a) 101, 104 c) 103, 105
b) Fault d) None.

Data of Questions. Q.161 Q.165 are given :

Consider a set of 5 processes whose arrival time, CPU time needed and the priority are given below :

Process priority	Arrival time (in ms)	CPU time needed(in ms)	Priority
P1	0	10	5
P2	0	5	2
P3	2	3	1
P4	5	20	4
P5	10	2	3

Answer the questions based on the above information (smaller the number, higher the priority) :

Q.161 If the CPU scheduling policy is FCFS, the average waiting time will be
a) 12.8 ms
b) 8 ms
c) 16 ms
d) None of the above.

Q.162 If the CPU scheduling policy is SJF, the average waiting time (without pre-emption) will be
a) 12.8 ms
b) 6.8 ms
c) 17 ms
d) None of the above.

Q.163 If the CPU scheduling policy is SW with pre-emption, the average waiting time will be
a) 8 ms b) 14 ms
c) 5.6 ms d) None of the above.

Q.164 If the CPU scheduling policy is priority scheduling without pre-emption, the average waiting time will be
a) 12.8 ms b) 11.8 ms
c) 10.8 ms d) None of the above.

Q.165 If the CPU scheduling policy is priority scheduling with pre-emption, the average waiting time will be
a) 19 ms b) 7.6 ms
c) 8 ms d) None of the above.

Q.166 Disks with geometries exceeding the following maximums could not be handled by early DOS systems :
Cylinders 1024
Heads 16
Sectors per track 63
What is the maximum size disk these systems could use?
a) 328 MB b) 428 B
c) 528 B d) None

Data of Questions. Q.167- Q.168 are given :
A disk has 8 sectors per track and spins at 600 rpm. It takes the controller 10 ms from the end of one 1/0 operation before it can issue a subsequent one. How long does it take to read all 8 sectors using the following interleaving systems ?

Q.167 No interleaving
a) 400 ms b) 500 ms
c) 600 ms d) 800 ms.

Q.168 Single interleaving
a) 50 ms b) 100 ms
c) 150 ms d) 200 ms.

Q.169 Double interleaving
a) 75 ms b) 175 ms
c) 275 ms d) None.

Q.170 A disk has 19456 cylinders, 16 heads and 63 sectors per track. The disk spins at 5400 rpm. Seek time between adjacent tracks is 2 ms. Assuming the read/write head is already positioned at track 0, how long does it take to read the entire disk?
a) 28 min b) 38 min
c) 48 min d) 58 min.

Q.171 If the average seek time for the disk in the previous problem is 10 ms, what is the average time to read a sector?
a) 12.73 ms b) 13.73 ms
c) 14.73 ms d) 15.73 ms.

Q.172 On a simple paging system with a page table containing 64 entries of 11 bits (including valid/invalid bit)

each, and a page size of 512 bytes, how many bits in the logical address specify the page number?
a) 2 b) 4
c) 6 d) None.

Q.173 On a simple paging system with a page table containing 64 entries of 11 bits (including valid/invalid bit) each, and a page size of 512 bytes, how many bits in the logical address specify the offset within the page ?
a) 3 b) 5
c) 7 d) 9

Q.174 On a simple paging system with a page table containing 64 entries of 11 bits (including valid/invalid bit) each, and a page size of 512 bytes, how many bits are in a logical address?
a) 9 b) 11
c) 13 d) 15.

Q.175 On a simple paging system with a page table containing 64 entries of 11 bits (including valid/invalid bit) each, and a page size of 512 bytes, what is the size of the logical address space ?
a) 2^7 b) 2^9
c) 2^{11} d) 2^{15}

Q.176 On a capability-based system, each file is associated with a unique 16 bit number. For each file, each user may have the read or write capability. How many bytes are needed to store each user's access data?
a) 65536 b) 16384
c) 26384 d) None.

Q.177 On a system using fixed blocking with 20-byte records and 50 byte blocks, how much space will be wasted in each block?
a) 5 bytes b) 10 bytes
c) 15 bytes d) None.

Q.178 Given a system using unspanned blocking and 100 byte blocks. A file contains records of 20, 50, 35, 70, 40, 20. What percentage of space will be wasted in the blocks allocated for the file?
a) 31.25% b) 41.25%
c) 51.25% d) None.

Q.179 Consider a system having 'm' resources of the same type. These resources are shared by 3 processes A, B, C, which have peak time demands of 3, 4, 6 respectively. The minimum value of 'm' that ensures that deadlock will never occur is
a) 11 b) 12
c) 13 d) 14.

Q.180 A system has 3 processes sharing 4 resources. If each process needs a maximum of 2 units then
a) deadlock can never occur
b) deadlock may occur
c) deadlock has to occur
d) none of the above.

Q.181 'm' processes share 'n' resources of the same type. The maximum need of each process doesn't exceed 'n' and the sum all their maximum needs is always less than m + n. In this set-up,
a) deadlock can never occur
b) deadlock may occur
c) deadlock has to occur
d) none of the above.

Q.182 A process refers to 5 pages, A, B, C, D and E in the following order A;B;C;D;A; B;E;A;B;C;D;E. If the page replacement algorithm is FIFO, the number of page transfer with an empty internal store of 3 frames is
a) 8 b) 10
c) 9 d) 7

Q.183 Consider a system in which a directory entry can store upto 16

disk block addresses. For files no larger than blocks, the 16 addresses serve as the file's index table. For files larger than 16 blocks, the addresses point to indirect blocks which in turn point to 256 file blocks each. A block is 1024 bytes. How big can a file be?

a) 27 b) 2"
c) 222 d) None.

Data of Questions Q.184-Q.185 are given: How many bits would be needed to store the free list under the following conditions if a bit map were used to implement the free list ?

Q.184 For 16 bits per disk address ; 500,000 blocks total ; 200,000 free blocks

a) 200,000 b) 300,000
c) 400,000 d) 500,000

Q.185 For 16 bits per disk address; 500,000 blocks total ; 0 free blocks

a) 200,000 b) 300,000
c) 500,000 d) None.

Q.186 On a system with D bits per disk address, B blocks total, and F free blocks, under what conditions will the free list array use less space than the bit map?

a) $D \times F = B$ b) $D \times F > B$
c) $D \times F < B$ d) None.

Q.187 For D having the value of 32 bits, what is the fraction of the disk space that must be free?

a) $< \frac{1}{20}$ b) $< \frac{1}{30}$
c) $< \frac{1}{40}$ d) None.

Q.188 If the number of available page frames is increased to 4 then

a) the number of page transfers decreases
b) the number of page transfers increases
c) the number of page transfers remains the same
d) none of the above.

Q.189 If there are 32 segments, each of size 1 k byte, then the logical address should have

a) 13 bits b) 14 bits
c) 15 bits d) 16 bits.

Q.190 Disk requests come to a disk driver for cylinders 10, 22, 20, 2, 40, 6 and 38, in that order at a time when the disk drive is reading from cylinder 20. The seek time is 6 ms per cylinder. The total seek time, I the disk arm scheduling algorithm is first-come- first-served is

a) 360 ms b) 850 ms
c) 900 ms d) None of the above.

Q.191 A certain moving arm disk storage with one head has following specifications :
Number of tracks/recording surface = 200
Disk rotation speed = 2400 rpm
Track storage capacity = 62500 bits

ANSWER KEY:

1	2	3	4	5	6	7	8	9	10	11	12	13	14
(c)	(c)	(c)	(c)	(c)	(c)	(b)	(a)	(a)	(d)	(d)	(b)	(a)	(b)
15	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**
(b)	(b)	(c)	(d)	(c)	(b)	(b)	(c)	(b)	(b)	(c)	(a)	(b)	(c)
29	**30**	**31**	**32**	**33**	**34**	**35**	**36**	**37**	**38**	**39**	**40**	**41**	**42**
(d)	(c)	(b)	(a)	(c)	(b)	(c)	(a)	(a)	(d)	(c)	(b)	(c)	(a)
43	**44**	**45**	**46**	**47**	**48**	**49**	**50**	**51**	**52**	**53**	**54**	**55**	**56**
(d)	(d)	(b)	(d)	(b)	(a)	(d)	(c)	(c)	(a)	(b)	(c)	(c)	(b)
57	**58**	**59**	**60**	**61**	**62**	**63**	**64**	**65**	**66**	**67**	**68**	**69**	**70**
(c)	(c)	(b)	(a)	(a)	(a)	(c)	(b)	(c)	(c)	(b)	(d)	(c)	(d)
71	**72**	**73**	**74**	**75**	**76**	**77**	**78**	**79**	**80**	**81**	**82**	**83**	**84**
(a)	(a)	(a)	(b)	(a)	(b)	(c)	(b)	(c)	(d)	(a)	(b)	(b)	(b)
85	**86**	**87**	**88**	**89**	**90**	**91**	**92**	**93**	**94**	**95**	**96**	**97**	**98**
(c)	(d)	(c)	(c)	(b)	(c)	(c)	(a)	(c)	(d)	(c)	(b)	(d)	(d)
99	**100**	**101**	**102**	**103**	**104**	**105**	**106**	**107**	**108**	**109**	**110**	**111**	**112**
(c)	(d)	(c)	(b)	(c)	(c)	(c)	(c)	(a)	(b)	(c)	(c)	(c)	(b)
113	**114**	**115**	**116**	**117**	**118**	**119**	**120**	**121**	**122**	**123**	**124**	**125**	**126**
(a)	(c)	(b)	(b)	(d)	(b)	(c)	(c)	(d)	(c)	(c)	(d)	(d)	(c)
127	**128**	**129**	**130**	**131**	**132**	**133**	**134**	**135**	**136**	**137**	**138**	**139**	**140**
(a)	(c)	(a)	(c)	(b)	(b)	(d)	(d)	(b)	(b)	(b)	(b)	(b)	(c)
141	**142**	**143**	**144**	**145**	**146**	**147**	**148**	**149**	**150**	**151**	**152**	**153**	**154**
(b)	(b)	(c)	(b)	(d)	(c)	(c)	(d)	(c)	(b)	(c)	(b)	(c)	(c)
155	**156**	**157**	**158**	**159**	**160**	**161**	**162**	**163**	**164**	**165**	**166**	**167**	**168**
(c)	(c)	(b)	(b)	(a)	(b)	(a)	(b)	(c)	(c)	(b)	(c)	(d)	(d)
169	**170**	**171**	**172**	**173**	**174**	**175**	**176**	**177**	**178**	**179**	**180**	**181**	**182**
(c)	(d)	(d)	(c)	(d)	(d)	(d)	(b)	(b)	(b)	(a)	(a)	(a)	(c)
183	**184**	**185**	**186**	**187**	**188**	**189**	**190**	**191**					
(c)	(d)	(c)	(c)	(b)	(b)	(c)	(d)	(a)					

EXPLANATIONS

Q.15 (b)
Deadlock occurs when each other of the 3 user processes hold one resource and make simultaneous demand for another. If there are 4 resources one of the 3 user processes will get the fourth instance of the resource and relinquish one or both of the resource(s) it is currently holding after

Q.25 (c)
Each p operation will decrease the Semaphore value by 1 and V operations increase it by 1. If x is 18, then 7P operations will make semaphore value 0. If this is followed by 7V operations the value comes back to 7. So after 18 P and 18 V operations, the value of the semaphore will be 7. The remaining 2P operations result in the semaphore value 5.

Q.29 (d)
Even in a non-multiprogramming system, memory protection may be used, when for example, spooling is being used.

Q.32 (a)
2 processes can never lead to deadlock as the peak time demand of (3+3) tape drives can be satisfied. But 3 processes can lead to deadlock if each holds 2 drives and then demand one more.

Q.41 (c)
$0.35\times10+(1-0.35)\times100= 68.5$ ns

Q.43 (d)
Size of Virtual Memory depends on Size of Hard Disk. It doesn't depend on Address Bus, since we have many addressing modes.

Q.44 (d)
The arrival pattern is a Poisson distribution.
P (K requests) $= e^{-\mu T}(\mu T)^k / k!$
Here k = 0, μ= 20, T = 3/4.

Q.52 (a)
It is 0+10+(15–2)+(18–5)+(38–10) divided by 5, i.e., 12.8 ms.

Q.53 (b)
It is 8+0+3+15+8 divided by 5, i.e., 6.8 ms.

Q.54 (c)
It is 10+3+0+15+0 divided by 5, i.e., 5.6 ms.

Q.55 (c)
It is 30 + 0 + 3 + 3 + 18 divided by 5, i.e., 10.8 ms.

Q.56 (b)
It is 30 + 3 + 0 + 5 + 0 divided by 5, i.e., 7.6 ms.

Q.60 (a)
Having 11 resources ensure that at least one process will have no pending request. This process after using will release the resources and so Deadlock can never occur.

Q.61 (a)
At least one process will be holding 2 resources in case of a Simultaneous demand from all the processes. That process will release the 2 resources, thereby avoiding any possible deadlock.

Q.62 (a)
Using Banker's algorithm, one can show that one process has to enquire all its needed resources. This process after completing its task, will release all its resource, thereby avoiding any possible deadlocks.

Q.63 (c)
The first 3 references A, B, C fills the internal storage with A, B, C in 3 page transfer. Now the next reference D results in a page fault. So, page A is downloaded and D takes its place after a page transfer. So, the internal store has D, B and C. The next reference is A- results in a page fault. So, a page transfer takes place and swaps B and A. Continuing this way, we find totally 9 page transfers are necessary.

Q.67 (b)
Refer Qn 63, Applying the same logic, we find the required number of page transfer is 10. So, increasing the number of pages need not necessarily reduce the number of page faults. It is the actual sequence of references that decides.

Q.69 (c)
To specify a particular segment, 5 bits are required (since 2^5=32). Having selected page, to select a particular byte one needs 10 bits (since 2^{10}=1 K byte).So, totally 5 + 10 = 15 bits are needed.

Q.70 (d)
The disk drive has to traverse totally 146 cylinders (verify). So, Seek time is 6 × 146 = 876 ms

Q.72 (a)
To cover 2400×62500 bits, 60 s are needed. Average latency time is the needed to traverse 100 tracks i.e., 100 ×62500 bits, which are 2.5 S.

Q.78 (b)
As this scheduling policy has a better average response time for the interactive users.

Q.79 (c)
When it tries to access 0100, it results in a page fault as the memory is empty tight now. So, it loads the second page (which has the address 100-199), Typing to access 200 will result in a page fault, as it not in memory right now. So the third page with the address from 200 to 299 will replace the second page in memory. Trying to access 430 will result in another page fault. Proceeding this way, we find trying to access the address 0510, 0120, 0220 and 0320 will all result in page faults. So, all together 7 page faults.

Q.80 (d)
There are 64 words in a block. So the 4K cache has (4×1024)/64= 64 blocks. Since 1 set has 4 blocks, there are 16 sets. 16 sets need 4 bits for representation. In a set there are 4 blocks, which needs 2 bits. Each block has 64 words. So, the word field has 6 bits.

Q.81 (a)
The memory configuration after the arrival of the jobs 1, 2, 3 and the termination of job 1 can be depicted as:

FREE-200	JOB2-350	JOB3-300	FREE-150

First fit algorithm will allocate the FREE-200 slot for job 4.But best fit

algorithm will allocate the FREE-150 slot for job 4. The memory configuration for the first fit and best fit will be

JOB4-120	FREE-80	JOB2-350	JOB3-300	FREE-150

And

FREE-200	JOB2-350	JOB3-300	JOB4-120	FREE-30

respectively. When job 5 arrives, it will be allotted the FREE-150 slot by the first fit algorithm and the FREE-200 slot by the first fit algorithm. The memory allocation table for the first fit and best fit will be

JOB4-120	FREE-80	JOB2-350	JOB3-300	JOB5-150

and

JOB5-150	FREE-50	JOB2-350	JOB3-300	FREE-30

When Job 6 arrives, it will be allotted the FREE-80 slot by the first fit algorithm. The best fit algorithm will find no room to store Job 5 as the needed 80 K, is not available contiguously. So, it has to wait till a job terminates. So, the first fit algorithm performs better in the case.

Q.89 (b)

Locality of reference is based on the fact that a page that is referenced is likely to be referenced again in the near future.

Q.93 (c)

Let the mutex be initialized to 1. Any one of the 9 processes p_i, i = 1, 2, 3 ..., 9 can get into the critical section after the execution P (mutex) which decrements the mutex value to 0. At this time P_{10} can enter into the critical section as it uses V (mutex) instant of P (mutex) to get into the critical section. As a result of this, mutex will be incremented by 1. Now any one of the 9 processes P_i, i = 1, 2, 3, ..., 9 (expecting the one that is already inside the critical section) can get into the critical section after decrementing the mutex to 0. None of the remaining processes can get into the critical section. If the mutex is initialized to 0, only 2 processes can get into the critical section. So the largest number of processes is 3.

Q.95 (c)

There is no limit to the number or processes that can be in the ready and blocked states. At most, n processes may be in the run state, since a process in the run state must be allocated to a CPU, and there are only n CPUs.

Q.96 (b)

This is combinatories problem of how many ways n objects can be ordered. When selecting the first object, one has n possible choices. For the second object, n-1. Each subsequent selection involves one less choice. The total number of possibilities 'is computed by multiplying all the possibilities at each point, making the answer n !

Q.97 (d)

65,536 is 2^{16}. Dividing the number of bytes of total memory by the size of each partition yield the number of partitions, $2^{24}/2^{16} = 2^8$. Eight bits are needed to store one of 2^8 partition numbers.

Q.98 (d)

The total amount of memory on, the system is irrelevant. Since 65,536 is 2^{16}, the largest valid address is 2^{16}-1. In binary, 2^{16}-1 is 16 ones. The limit register must have 16 bits.

Q.99 (c)

Since the system has only one limit register, it must be large enough to accommodate addresses in any partition. The largest possible partition is 2^{32} Therefore. The limit register must have 32 bits.

Q.103 (c)

Record 5 is preceded by 4 records, each of 15 bytes. The fifth record will start at byte (4*15)+1=61.

Q.104 (c)

The logical layout of the records is the same for both access methods.

Q.105 (c)

Record N will start at byte ((N-1)*S)+1.

Q.106 (c)

Six. Sequential access typically is one way, from start to finish. To go back to the sixth record, the input would have to start over from record one. Typically, the input stream would have to be reopened or rewound to accomplish this.

Q.107 (a)

The maximum number of holes is created by having alternating free and used blocks. Thus, the maximum number of holes is (K+1)/2 (where 'I' represents integer division). The minimum number of holes is zero, which occurs when all of memory is in use.

Q.109 (c)

Processes will run until they block. For each cycle, s units of overhead will be needed to accomplish r units of useful work. CPU efficiency is $\frac{r}{(r+s)}$

Q.110 (c)

Since processes will still run until they block, the answer is the same as for part **(a)**.

Q.111 (c)

The number of switches required will be r/q, making the time wasted on switches sr/q. CPU efficiency is $\frac{r}{(r+sr/q)} = \frac{r}{(q+s)}$

Q.112 (b)

Same answer as above except with q=s, the equation evaluates to $\frac{1}{2}$.

Q.113 (a)

Using the equation in part **(c)**, as q goes to 0. CPU efficiency goes to 0.

Q.114 (c)

The process will be interrupted at time 2 in the first queue, time 2+7 in the second queue, time 9+12 the third queue, time 21+17 the fourth queue, and will be running in the fifth queue when it terminates. There were four interrupts.

Q.115 (b)

Q.116 (b)

Given the bus is 32 bits wide, the controller can transfer 4 byte 10,000,000 times per second or 4 bytes every 100 ns. The controller will steal a cycle once every 10 instruction fetches for a slowdown of 10%.

Q.117 (d)

At most 2^{14} memory locations can be accessed. All addresses 2^{14} and higher refer to I/O devices.

Q.119 (c)

100×0.75 + 1×0.25 = 75.25.

Q.120 (c)

100×0.50 + 1×0.50 = 50.50.

Q.121 (d)

The logical address space contains 256= 2^8 pages of 2^{10} bytes, making the total logical address space $2^{10}\times2^{18}=2^{18}$ bytes. An 18 bit address is required to cover a 2^{18} byte address space.

Q.122 (c)

The page frame size is the same as the page size. Therefore, the page frame size is 2^{10} bytes.

Q.123 (c)

A 24 bit address is required to cover the 2^{24} byte physical address space. Since pages, and therefore page frames, are 2^{10} bytes, the last 10 bits in the physical address specify the page offset. The remaining 14 bits specify the page frame number.

Q.124 (d)

The page table must contain an entry for each page. Since there are 256 pages in the logical address space, the page table must be 256 entries long.

Q.125 (d)

Each entry contains 1 bit to indicate if the page is valid and 14 bits to specify the page frame number.

Q.126 (c)

Each P operation will decrease the semaphore value by 1 and V operation increase by 1. If x is 18, then 7P operations will make the semaphore value 0. If this is followed by 7V operations the value comes back to 7. So, after 10P and 18V operations, the value of the semaphore will be 7. The remaining 2 P operations result in the semaphore value 5.

Q.127 (a)

2 process can never lead to deadlock as the peak time demand of 6(3+3) tape drives can be satisfied. But 3 processes can lead to a deadlock if each hold 2 drives and then demand one more.

Q.128 (c)

0.35×10+(1-0.35) ×100=68.5ns.

Q.129 (a)

The arrival pattern is a Poisson distribution.

P (k requests)= $e^{-\mu T}(\mu T)^k/k$

Here k= 0, μ= 20, T = 3/4.

So required probability is e^{-15}.

Q.130 (c)

Turnaround time is computed by subtracting the time the process entered the system from the time it terminated.

$$\frac{((3-0)+(9-1)+(13-4)+(15-6))}{4}=7.25$$

Q.131 (b)

$$\frac{((3-0)+(9-1)+(15-4)+(11-6))}{4}=6.75$$

Q.132 (b)

$$\frac{((3-0)+(15-1)+(8-4)+(10-6))}{4}=6.25$$

Q.133 (d)

$$\frac{((5-0)+(13-1)+(15-4)+(11-6))}{4}=8.25$$

Q.134 (d)

$$\frac{((4-0)+(15-1)+(14-4)+(12-6))}{4}=8.50$$

Q.135 (b)

Wait time can be computed by subtracting a process's execution time from its turn round time.
A:(3-3)=0, B:(8-6)=2,
C:(9-4)=5, D:(9-2)=7.

Q.136 (b)
A:(3-3)= 0, B:(8-6)=2,
C:(11-4)=7, D:(5-2)=3

Q.137 (b)
A :(3-3)= 0, B:(14-6)= 8,
C:(4 -4)= 0, D:(4-2)=2.

Q.138 (b)
A:(5-3)=2, B:(12-6)=6,
C:(11-4)= 7, D:(5-2)=3.

Q.139 (b)
A:(4-3)= 1, B:(14-6)=8,
C:(10-4)= 6, D:(6-2)=4.

Q.140 (c)
A : 4, B : 9, C : 10, D : 12

Q.141 (b)
A : 4, B:13, C : 3, D:5

Q.142 (b)
A : 4, B : 13, C : 3, D : 5

Q.143 (c)
A : 4, B:13, C:5, D:7

Q.144 (b)
A:5, B:13, C:6, D:7

Q.145 (d)
5.00

Q.146 (c)
2.50

Q.147 (c)
2.50

Q.148 (d)
3.50

Q.149 (c)
4.00

Q.150 (b)
The hit ratio can be computed from the equation,
2×(1-m)+ 100 m=41.2
2-2 m+ 100 m=41.2
39.2 = 98 m 0.40 = m
Doubling memory to 16 Mb would lower the miss rate to 0.20.
2×0.80+100×0.20=21.6
An additional doubling to 32 Mb would be needed to lower the mean access time below 20 ms.
2×0.90+100×0.10= 11.8.

Q.151 (c)
The tracks travelled to will be 345, 123, 874, 692, 475, 105 and 376, making the total distance 222+ 751 +182 +217 +370 +271 =2013

Q.152 (b)
The tracks travelled to will be 345, 376, 475, 692, 874, 123and 105, making the total distance 529+769 = 1298.

Q.153 (c)
The tracks travelled to will be 345, 123, 105, 0, 376, 475, 692 & 874, making the total distance 345+874 = 1219.

Q.154 (c)
The tracks travelled to will be 345,123, 105, 376, 475, 692 and 874, making the total distance 240 + 769 = 1009.

Q.155 (c)
The tracks travelled to will be 345, 123, 105, 0, 999, 874, 692, 475 & 376, making the total distance 345+999+623=1967

Q.156 (c)

The tracks travelled to will be 345, 123, 105, 874, 692, 475 & 376, making the total distance 240+ 769+ 498 = 1507.

Q.157 (b)

Virtual address 10451 is offset 451 in page 5. Page 5 is "out" resulting in a fault.

Q.158 (b)

Virtual address 5421 is offset 1421 in page 2. Page 2 is "in" at frame 200 which is address 400,000. Adding the offset to the frame's starting address yields physical address 401, 421.

Q.159 (a)

Virtual address 14123 is offset 123 in page 7. Page 7 is "in" at frame 101 which is address 202,000. Adding the offset to the frame's starting address yields physical address 202, 123.

Q.160 (b)

Virtual address 9156 is offset 1156 in page 4. Page 4 is "out" resulting in a fault.

Q.161 (a)

It is 0+10+(15-2)+(18-5)+(38-10) divided by 5, i.e., 12.8 ms.

Q.162 (b)

It is 8+0+3+15+8 divided by 5, i.e. 6.8 ms.

Q.163 (c)

It is 10 + 3 + 0 + 15 + 0 divided by 5, i.e., 5.6 ms.

Q.164 (c)

It is 30 + 0 + 3 + 3 + 18 divided by 5, i.e., 10.8 ms.

Q.165 (b)

It is 30 + 3 + 0 + 5 + 0 divided by 5, i.e., 7.6 ms.

Q.166 (c)

1024 x 16 x 63 x 512 = 528,482,304 = 528 Mb.

Q.167 (d)

The disk makes 10 revolutions per second or one revolution in 100 ms. In 10 ms, less than one sector will have passed under the read/write head. The next sector cannot be read until the disk makes almost a complete revolution. It will require 8 revolutions to read all 8 sectors. At 100 ms per revolution, it will take 800 ms.

Q.168 (d)

The next sector will spin under the read/write head almost as soon as the next I/O operation is issued. Two revolutions will be needed to read all 8 sectors, making the total read time 200 ms.

Q.169 (c)

A total of 2.75 revolutions will be needed to read all 8 sectors, for a total time of 275 ms.

Q.170 (d)

Each track can be read in one revolution, or 11.11ms. To read all 19456×16 tracks requires approximately 3459 seconds. Seek time is (19456-1)×2=39s. Total time is 3498s= 58 min.

Q.171 (d)

Seek time will be 10 ms. Latency time will be 5.55 ms. Transfer time will be 0.18 ms. Total time is 15.73 ms.

Q.172 (c)

Since there are 64= 26 pages, 6 bits of the logical address are required to specify the page number.

Q.173 (d)

Since a page is 512 = e bytes long, 9 bits are required to specify the offset within the page. -

Q.174 (d)

The 6 most significant bits of the logical address would specify the page number, the 9 least significant bits would specify the page offset, making the total number of bits 15.

Q.175 (d)

In the previous problem, the logical address was determined to have 15 bits. A 15 bit address creates an address space of 32,768 =215.

Q.176 (b)

Two bits are needed for each of 65,536 (2^{16}) files. Thus, 131,072 bits or 16,384 bytes would be needed.

Q.178 (b)

The first block can contain the first two records, leaving 30 bytes wasted. The second block can only contain the third record, wasting 65 bytes. The third block can only contain the fourth record, wasting 30 bytes. The last two records will be allocated to the fourth block, which will contain 40 wasted bytes. Thus, of the 400 bytes in the four blocks, 30+ 65 + 30 + 40 = 165 bytes, or 41.25% is

Q.179 (a)

Having 11 resources ensures that at least 1 process will have no pending request. This process after using will release the resources and so deadlock can never occur.

Q.180 (a)

At least one process will be holding 2 resources in case of a simultaneous demand from all the processes. That process will release the 2 resources, thereby avoiding any possible deadlock.

Q.181 (a)

Using Banker's algorithm, one can show that one process has to acquire all its needed resources. This process, after completing its task, will release all its resources, thereby avoiding any possible deadlock.

Q.182 (c)

The first 3 references A, B, C fills the internal storage with A, B, C in 3 page transfers. Now the next reference D results in a page fault. So, page A is downloaded and D takes its place, after a page transfer. So, the internal store has D, B and C. The next reference is A results in a page fault. So, a page transfer takes place and swaps B and A. Continuing this way, we find totally 9 page transfers are necessary.

Q.183 (c)

The largest possible file will have $16=2^4$ indirect blocks. Since each indirect block can point to $256=2^8$ blocks, the largest file has $256=2^{8+4}$ blocks. Given a block size of $1024=2^{10}$, the maximum file size is $2^{4+8+10}= 2^{22}$

Q.184 (d)

In either case, the number of bits per address and the number of free blocks are irrelevant. The status of each of the 500,000 blocks is stored in a bit. 500,000 bits are needed.

Q.186 (c)

$D \times F < B$.

Q.187 (b)

$32 \times F < B$

$F < \frac{B}{32}$

Less than one thirty-second of the blocks can be free.

Q.188 (b)

Applying the same logic, we find the required number of page transfer is 10. So, increasing the number of pages need not necessarily reduce the number of page faults. It is the actual sequences of references that decides.

Q.189 (c)

To specify a particular segment, 5 bits are required (since$2^5=32$). Having selected a page, to select a particular byte one needs 10 bits (since 2^{10}= 1 k byte). So totally 5+10=15bits are needed.

Q.191 (a)

To cover 2400×62500 bits, 60 s are needed. Average latency time is the time needed to traverse 100 tracks i.e., 100×62500 bits, which is 2.5 s.