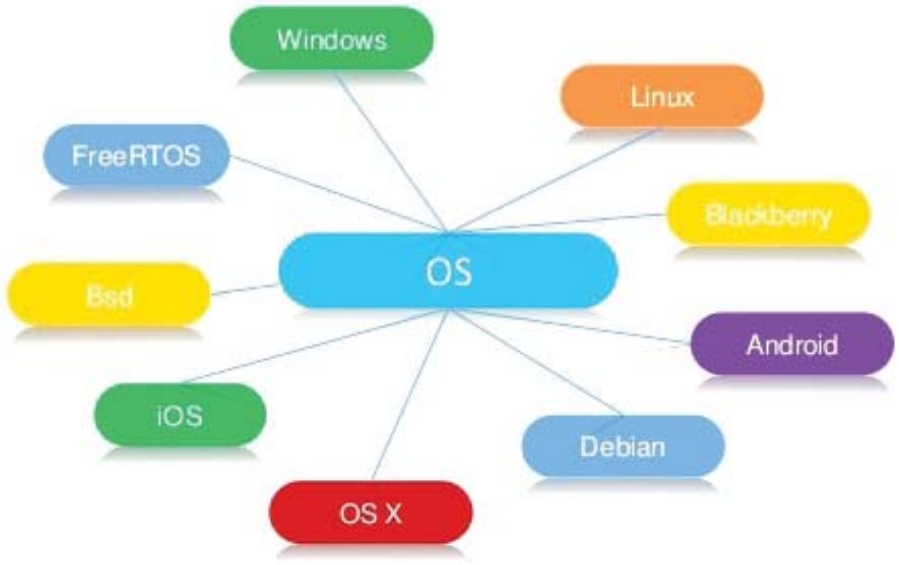
GATE | PSUs

COMPUTER SCIENCE & INFORMATION TECHNOLOGY

OPERATING SYSTEMS

Volume - I : Study Material with Classroom Practice Questions



ACE Engineering Academy
(Leading institute for ESE/GATE/PSUs)

HYDERABAD | DELHI | BHOPAL | PUNE | BHUBANESWAR | LUCKNOW | PATNA | BENGALURU | CHENNAI | VIJAYAWADA | VIZAG | TIRUPATHI | KUKATPALLY | KOLKATA

Operating Systems

(Classroom Practice Booklet Solutions)

1. Process Management – I

01. Ans: (c)

02. Ans: (c)

03. Ans: (a)

Sol: Software Interrupt is generated as a result of execution of a privileged instruction. This would change the mode from user to Kernel & vice-versa. System calls, state changes can be caused by program commands, which are referred to as system calls and are implemented using software interrupts or automatically following certain events.

04. Ans: (d)

Sol: Loader is frequently required system software.

05. Ans: (b)

Sol: Process going from running to ready state is always preemptive.

06. Ans: (c)

07. Ans: (c)

Sol: Scheduler process is meant to decide which ready process next should run on CPU.

08. Ans: (b)

Sol: In general, if no file name is specified in a command, the shell takes as input that you type on your keyboard.

09. Ans: (a)

Sol: Have multiple processes in ready to run.

10. Ans: (a)

Sol: S.J.F is the optimal non-preemptive CPU scheduling algorithm.

Explanation: SJF is the optimal non-preemptive CPU scheduling algorithm. So, in order to produce the optimal Solution here it considers the shortest job first. The optimal sequence is {j3, j2, j1}.

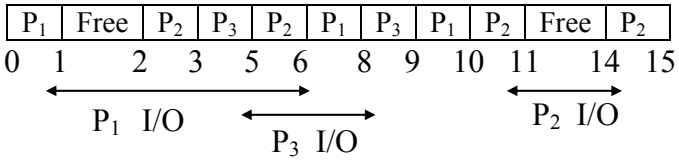
Since, Burst time(j3) <Burst time(j2)< Burst time(j1)

But to start j3 CPU Should wait for 1.0 units of time as its arrival time is 1.0.

11. Ans: (b)

Sol: Each process runs for q period and if there are n process p_1 p_2 p_3..... p_n p_1 p_2... then p_1 turn comes again when it has completed time quanta for remaining process p_2 to p_n i.e., it would take at most (n–1)q time. So Each process in round robin gets its turn after <= (n–1)q time when we don't consider

overheads, but if we consider overhead(s) then it would be ns+(n-1)q So, ns+(n-1)q <= t overhead will be reduced when time quantum is maximum allowable i.e., q >= (t–ns)/(n–1).



12. Ans: (b)

Sol: $P_1 \rightarrow$ Finishing time = 10

$P_2 \rightarrow$ Finishing time = 15

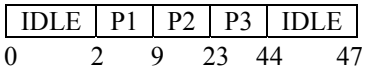
$P_3 \rightarrow$ Finishing time = 9

13. Ans: (b)

Sol: Draw the Gantt Chart and observe the CPU idle times.

Process	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

Draw the Gantt chart and observe the CPU idle time.



% OF TIME CPU IS IDLE = 5/47 × 100

= 10.6

14. Ans: (c)

Sol: Hint: Draw the Gantt- Chart.

15. Ans: (d) **16. Ans: (a)**

17. Ans: (b)

Sol: Consider the each statement:

(i) Statement is false because there is no connection between kernel supported threads and context switch.

(ii) Statement is true and it is drawback of user-level threads.

(iii) Statement is true because kernel-supported threads having own memory area and scheduled independently by the OS.

(iv) Statement is false because kernel is unaware about user level threads and there is no kernel support to user-level threads.

18. Ans: (b)

Sol: The critical distinction between them is data sharing processes do not share address space without explicit assistance. Threads within a process share address space.

19. Ans: (c)

Sol: RR is a pre-emptive scheduler, which is designed especially for time-sharing systems. In other words, it does not wait for a process to finish or give up control. In RR, each process is given a time slot to run. If the process does not finish, it will "get back in line" and receive another time slot until it has completed.

20. Ans: (b)

Sol: Total number of jobs executed per unit time (also called throughput)is maximum in SJF. Since shorter jobs are executed first and More number of jobs will be executed per unit time.

21. Ans: (c)

Sol: The number of child processes with n-fork() is $2^n - 1$.

22. Ans: (b)

23. Ans: (d)

Sol: Due to convoy effect.

24. Ans: (d)

25. Ans: (d)

Sol: Block initiated by process and Ready by OS.

26. Ans: (b) **27. Ans: (a)**

28.

Sol: The process will have increases it priority because of two pointers pointing same process. The advantage is, it will be given more time but shorter processes will suffer.

29.

Sol: CPU efficiency

= CPU useful time/ total CPU time.

(a) T/T+S
(b) T/T+S
} Because Q is more than T, AftertimeT process blocks for I/O

(c) Since Q < T, Each run of T requires T/Q switches. The resultant overhead is ST/Q.

$$\text{The efficiency} = T\left(T + \frac{ST}{Q}\right)$$

$$= \frac{Q}{Q+S}$$

(d) Since Q = S from (c) replace Q by S which is 50%.

(e) The efficiency becomes 0 as Q is ready 0

30.

Sol: (i) The SJF has the highest priority

(ii) The lowest level of Multilevel Feed back queue is FCFS.

(iii)FCFS gives highest priority to the job having been existence the longest time.

(iv) None

31.

Sol: Run in the order

x, 3, 5, 6, if x < 3

else 3, x, 5, 6, if x > 3 & x < 5

else 3, 5, x, 6 if x > 5 & x < 6

else 3, 5, 6, x if x > 6

by applying SJF; as it generates the least any waiting time.

There are only two main choices FCFS and SJF as no priority and Quantum size etc. has been given. Moreover x may take values 1,2,3,4,5, and more than that, it is assumed that the job responses immediately after getting the CPU. Then response time will be same as waiting time. SJF calculations are shown by taking various values of x. SJF is considered because it produces minimum response time.

Case:1 x= 1

Average waiting time = (0+1+4+9)/4

= 14/4 = 3.5

x	3	5	6

0 1 4 9 15

Case: 2 x = 2

This job will again be the first job to be executed.

Average waiting time = (0+2+5+10) / 4

= 17 / 4 = 4.25

Case: 3 x = 3.

Assuming x will again be the first job to be executed.

Average waiting time = (0+3+6+11)/ 4 = 5

Case: 4 x = 4.

3	x	5	6

0 3 7 12 18

Average waiting time

$$= \frac{0+3+7+12}{4} = \frac{11}{2} = 5.5$$

and so on.

Thus, the order of execution depends on the value of x and average response time is minimum x = 1.

32.

Sol: **Hint:**Refer Process State Transition Diagram

33.

Sol: One is Zero and the other is the pid of the child process.

34.

Sol: Creates processes incessantly.

2. Process Management – II

01. Ans: (c)

Sol: Critical section implies usage of shared resources.

02. Ans: (b)

Sol: Since semaphore value S=10

6P $\Rightarrow$ S = S – 6 = 4

4V $\Rightarrow$ S = S + 4 = 8

03. Ans: (b)

Sol: The semaphore value S = 7.

P operation will decrement the value of the semaphore by 1.

V operation will increment the value of the semaphore by 1.

20P operations $\Rightarrow$ S = S – 20

= 7 – 20 = – 13

15V operations $\Rightarrow$ S = S + 15

= – 13+15 = 2

04. Ans: (d)

Sol: This is the algorithm of solution of consumer produces process with the help of semaphore.

So, K = P (empty) $\rightarrow$ P for wait

L = V(full) $\rightarrow$ V for signal

M = P(full)

N = V(empty)

05. Ans: (d)

Sol: S_x and S_y are two binary semaphore if assume P means wait or V means signal then for two process P1 and P2 we take alternate then there is no chance of deadlock.

The code is as follows.

P_1	**P_2**
while true do{	while true do{
L_1 : P(S_x)	L_3 : P(S_x)
L_2 : P(S_y)	L_4 : P(S_y)
x = x + 1;	y = y + 1;
y = y – 1;	x = y – 1;
V(S_x);	V(S_y);
V(S_y);	V(S_x);
}	}

06. Ans: (a)

Sol: When one process executes critical section the other process waits up to shared variable Busy = False.

07. Ans: (c)

Sol: If P_1 access the variable critical_flag then it executes the critical section otherwise P_2 executes the critical section but both cannot and there is a possible deadlock.

08. Ans: (d)

Sol: Based on strict alternation.

09. Ans: (c)

Sol: (a) All philosophers gets one fork, each philosopher waits for the other fork held by other philosophers.

(b) There is a deadlock.

(c) This avoids deadlock.

10. Ans: (b)

Sol: If two barriers are invocated immediately then two goes into deadlock.

11. Ans: (b) **12. Ans: (c)**

13. Ans: (d)

Sol: Fetch-And_Add( ) is stated as atomic. Therefore, even if pre-emption takes place within the while loop, the other process will be denied the access to CS.

14. Ans: (d)

Sol: X, W reads x and increment x by 1

Y, Z reads x and decrement by 2

start with X. will perform P(S) then S=1, read.

x = 0, x = x + 1 = 1

Then Y will perform P(S) then S = 0, read

x = 0, x = x–2 = –2, then store x. V(S), S = 1

Then Z will perform P(S) then S = 0, read

x = –2, x = x–2 = –4, then store x,V(S),S = 1

Then x will store x, V(S), S = 2, x = 1

Then W will perform P(S), S = 1, read x =1

x = x + 1 = 2, store x, V(S), S = 2, x = 2

15. Ans: (a)

16. Ans: (a)

Sol: If context switching is disabled in P, and if the value of semaphore is 'one', then it goes into infinite loop.

17. Ans: (a)

18. Ans: (d)

Sol: As both processes can enter CS together hence ME is violated. After the entry of both processes into CS, on one process will continuously enter the CS infinite times and the other process would never be allowed to enter the CS and the situation is like a Livelock.[then option (d) is correct].

19. Ans: (a)

Sol: The Program can be proved to demonstrate that in any concurrent situation not more than one process is allowed to enter CS. Sometimes even no process is allowed after one process enters and comes out of CS. Hence at most one process in CS at any time. The remaining all conditions can be proved to be incorrect.

20. Ans: (a) **21. Ans: (c)**

22. Ans: (b) **23. Ans: 1 and 2**

24.

Sol: (a) Mutual exclusion is guaranteed

(b) Deadlock occurs; Allow both processes to get pre-empted after S_2 & Q_2 respectively.

(c) Now mutual exclusion is not guaranteed Deadlock is not possible, as both the processes would be executing the same code.

25. (a) Ans: 3

(b) Ans: 3

(c) Ans: 0

3. Deadlocks

01. Ans: (c)

Sol: Only c is not a valid deadlock prevention scheme.

02. Ans: (a)

Sol: Deadlock prevention deals only with preventing mutual exclusion, hold & wait, No Preemption, Circular wait.

03. Ans: (d)

04. Ans: (a)

Sol: This is dead lock free but cause starvation.

05. Ans: (b)

Sol: If there are 5 processes then at least one process will get two tape drives out of six tape drives. Hence the system is deadlock free.

06. Ans: (c)

Sol: As there are 3 processes and 4 resources then at least 1 process will get 2 resources. Therefore, Deadlock will not occur.

07. Ans: (a)

Sol: If there are 2 processes then each process will hold 3 tape drives as there are 6 tape drives for which the system is guaranteed to be deadlock free.

08. Ans: (d)

Sol: If there are 13 resources then deadlock will not occur as the peak demands is also 13.

09. Ans: (b)

Sol: Right now (n-2) processes are blocked, as available is zero. Only 'p' & 'q' processes can complete, upon which they would release their resources x_p, x_q. therefore if $x_p + x_q \geq \min y_{k\ k \neq p,q}$ this would guarantee that one out of the blocked processes can come out of the cycle. Right now (n–2) processes are blocked, as available is zero. Only 'p' &'q' processes can complete, upon which they would release their resources X_p, X_q therefore if $X_p + X_q >= \min Y_{k\ k\ != p,\ q}$ this would guarantee that one out of the blocked processes can come out of cycle.

10. Ans: (d) **11. Ans: (c)**

12. Ans: (a)

Sol: **Hint:** Draw the timing diagram for all three processes; one can observe that all the process requests could be satisfied at different intervals of time and eventually leading to successful completion of them.

13. Ans: (b)

Sol: Deadlock is not possible, because there 3 resources and 2 process, each needs a maximum of two resources.

14. Ans: Yes

15. (b) Ans: Need = Max-Allocation

(c) Ans: It is safe because the Needs of all processes are satisfiable with the available resources.

(d) Ans: Request should be granted only if the resulting state of the system is safe after granting the request. However if it is not then it is not granted and the process initiating the request is blocked.

16. Ans: Run Safety Algorithm and check that the needs of all processes are satisfiable with the available resources.

4. Memory Management

01. Ans: (b) **02. Ans: (b)**

03. Ans: (c)

Sol: Memory specification

= Number of words × width of word.

04. Ans: (c) **05. Ans: (a)**

06. Ans: (b)

Sol: Initialized early means it is FCFS.

If FIFO page replacement algorithm is used then a memory page containing a heavily used variable that was initialized very early and is in constant use is removed.

07. Ans: (c)

Sol: By definition of virtual memory.

08. Ans: (b)

Sol: Virtual Memory is implemented on secondary storage.

09. Ans: (c)

Sol: Single level paging has overhead of large page table sizes.

10. Ans: (c)

Sol: due to Belady's anomaly

Bélády's anomaly proves that it is possible to have more page faults when increasing the number of page frames while using the First in First Out (FIFO) page replacement algorithm.

11. Ans: (a)

12. Ans: (b)

Sol: **Bélády's anomaly** proves that it is possible to have more page faults when increasing the number of page frames while using the First in First Out (FIFO) page replacement algorithm.

locality of reference, also known as the **principle of locality**, is the phenomenon of the same value or related storage locations being frequently accessed.

13. Ans: (d)

14. Ans: (a)

Sol: Dirty bit indicates whether page is clean or modified

15. Ans: (a)

Sol: Segment table must be paged in order to be accommodated in one page.

In a paged segmented scheme of memory management, the segment table itself must have a page table because the segment table is often too large to fit in one page.

16. Ans: (b)

Sol: **locality of reference**, also known as the **principle of locality**, is the phenomenon of the same value or related storage locations being frequently accessed. There are two basic types of reference locality. Temporal locality refers to the reuse of specific data and/or resources within relatively small time durations. Spatial locality refers to the use of data elements within relatively close storage locations. Sequential locality, a special case of spatial locality, occurs when data elements are arranged and accessed linearly, e.g., traversing the elements in a one-dimensional array.

17. Ans: (d)

18. Ans: (b)

Sol: At any time required 14KB A, B, and C or D and E when you load it requires 14KB.

19. Ans: (c)

20. Ans: (a)&(b)

Sol: (a) Is false because memory references are dynamically translated into physical address at runtime.

(b) Is false, because all pieces of a process need not be loaded into main memory.

21. Ans: (d)

Sol: Absolute addresses will be assigned only by loader.

22. Ans: (d)

Sol: EMAT

= 0.96[1ns + 0.9 (1 ns) + 0.1(1 ns+ 10 ns)]
+ 0.04[1 ns + 10 ns + 10 ns + 0.9 (1ns) +0.1 (1ns+10ns)]

EMAT

=0.96[1ns+0.9(1ns)+0.1(1ns+10ns)]+0.04
[1ns +10ns+10ns+0.9(1ns)+0.1(1ns+10ns]

TLB is successfully 96% of total request & for remaining 4%.

RAM is accessed twice.

So average time taken.

= .96(1 +(0.9 #1) +0.1 #(1 +10))
+.04(21 +(.9 #.1)) +0.1 #(1 +10)

= .96(1 +.9 +1.1) +0.4(21 +.09 +1.1)

= .96 #3 +0.4 #23

= 2.88 +.92

= 3.80, 4 ns

23. Ans: (c)

Sol: First level page table size = 4 Kb in the second level, we require 3 pages of the inner page table. One for the code, another for data and last one for stack. A size of each page is 4 KB; outer page table is also 4 KB.

24. Ans: (b)

Sol: a: True

b: False

c: True

d: True

25. Ans: (b)

Sol: Virtual address = 32 bits

Physical address = 30 bits

Page size = 4 K byte

Page table entry = 32 bits

Number of pages = $2^{32}/2^{12} = \frac{VA}{Page\,size} = 2^{20}$

Minimum number of bits

$= 2^{30}/2^{12} = 2^{10} = 10$ bits

26. Ans: (c)

27. Ans: (c)

Sol: Find the reference string first the number of page faults = length of reference string in this case as the number of frames is one.

Find the reference String first:

It is 1,2,4,5,1,2,3.

The number of Page faults = length of reference string in this case as the number of frames is one.

28. Ans: (b) & (d)

29. Ans: (b)

Sol: Page frame number is most important and virtual page number may not be stored entirely.

30. Ans: (c)

31. Ans: (b)

Sol: Linking is done at run-time in Dynamic Linking.

32. Ans: (a) **33. Ans: (c)**

34. Ans: (d)

35. Ans(a)

Sol: The number of allocation units

$= 1GB/64KB = 2^{14}$.

1 bit is required for each allocation unit.

36. Ans: (c)

37. Ans: (b)

Sol: Effective Memory Access Time

= p * (page fault service time)

+ (1–p) *(Memory access time)

$= \left(\frac{1}{10^6}\right) * 10 * 10^6 \text{ ns} + \left(1-\left(\frac{1}{10^6}\right)\right) * 20 \text{ ns}$

= 30 ns (approximately)

38.

Sol: (a) 0, 4302 A: Error

(b) 1, 15 A: Error

(c) 2, 50 A: 90+50

(d) 3, 400 A: Error

(e) 4, 112 A: Error

39. Ans: (a) **40. Ans: (b)**

41. Ans: (c) **42. Ans: (a)**

43.

Sol: Physical address for the following logical addresses = Base + logical address

(a) 219 + 430 = 649

(b) 2300 + 10 = 2310

(c) 90 + 500 = 590 (Illegal reference).

(d) 1327 + 400 = 1767

(e) 1952+112 = 2064 (Illegal reference)

44.

Sol: D, E; will

45.

Sol: Good are A, C, E, F

46.

Sol: No. of entries equals no of pages. i.e., 32G

47.

Sol: 3 Levels using the conventional formula of page table size i.e N*e Bytes

48.

Sol: Using the equation EMAT = P*S+(1-P)*M Substituting the values from the statements we get roughly in micro seconds.

49.

Sol: Upon IO

50.

Sol: (a) 212 in 300 ; 417 in 500 ; 112 in 200 ; 350 in 600;

(b) Internal Frag. Is the difference of the Partition size and the accommodated process size.

(c) No External Fragmenetation in Fixed Partitoning.

51.

Sol: (a) 32 bits. P is 20 bits and D is 12 bits.

(b) 16

(c) 1M

52. Ans: 32768

Sol: 64 × 512 B = 32768

53. Ans: 1 Byte

Sol: Number of Partitions = 2^{24}/ 64K which is 256 and hence addressed with a Byte.

54. Ans: 384

Sol: Page Table Size = $\left(\frac{\text{Number of pages}}{\text{Page size}}\right) * \text{PTE}$

Number of Pages = 2^40B/16KB

PTE = 6B

$$\frac{2^{40}\text{B}}{16\,\text{KB}} \times 6\text{B} = 384\text{MB}$$

55.

Sol: (a) Minimum page size =128 Bytes.

Let page size = 2^k Bytes

Page table size = $2^{13} = (2^{13-k}) \times 2$ bytes

$= 2^{14-k}$ Bytes

By given condition $2^{14-k} = 2^k$, (k=7)

and hence page size = 2^7=128 Bytes.

56.

Sol: $\frac{2^{19} * 100\,\text{ns}}{100\,\text{ms}} = 51$ approx

57. Ans: (a) 0

(b) 2

(c) 1

(d) 0

58. Ans: (b)

Sol: LAS = 2^{48} B

PAS = 2^{32} B

Page Size PS = 8 K = 2^3.2 B = 2^{13} B

$$\text{Number of pages} = \frac{LAS}{PA} = \frac{2^{48}}{2^{13}} = 2^{35}$$

$$\text{Number of frames} = \frac{2^{32}}{2^{13}} = 2^{19}$$

59.

(a) **Ans:** 400 (2*mmat)

(b) **Ans:** 0.75*200 + 0.25*400 = 250 ns

60. Ans: 12 bits

Sol: $\frac{2^{32}}{2^{20}} = 12$ bits

61. Ans: (d) & (e)

5. File System & I/O Management

01. Ans: (b)

Sol: Nearest cylinder next is also known as shortest seek time first which is the optimal algorithm.

02. Ans: (b)

Sol: To facilitate/minimize the Disk access time.

03. Ans: (d)

Sol: cache, memory and registers are not devices.

04. Ans: (a)

Sol: Large Block size results in more internal fragmentation and reading a larger block results in reading more data and hence higher throughput.

05. Ans: (a)

Sol: Disk capacity = $16 \times 128 \times 256 \times 512$ B = 256Mb

Sector address = 4 + 7 + 8 = 19 bits.

06. Ans: (a) **07. Ans: (c)**

08. Ans: (d) **09. Ans: (c)**

10. Ans: (a) Transfer time is 204 ms

(b) Access time is 4 + 20 + 2014 ms

(c) Rot. Time is 40 ms

(d) Time to read a sector 1/100 ms

(e) Time to read a track is 40 ms

11. Ans: 3

Sol:

0	1	2	3	4	5	6	7	8							
1	0	1	0	0	1	0	1	1	0	0	0	0	1	1	In use
0	0	0	1	1	0	0	0	0	1	0	1	1	0	1	Free
	error					error				error					

12.

Sol: Data Transfer rate is measured in Bytes per second. It is the rate at which the number of bytes that can be transferred in one second. It is calculated with reference to track size and rpm.

To read all 8 sectors in a double interleaved disk two full rotations and latency of 0.5 is needed.

13.

Sol: 64TB which is the sum of 12 direct pointers and size accessible with single double and triple indirect pointers. Ignoring the sizes in front of triple indirect pointer the file size roughly is 64TB.