①

Time ↓	P_1	P_3	P_5	P_7	P_8
23	Waiting for IO to complete	Wating for IO to complete	Suspended	Blocked	Running
37	Ready	Ready	Swapped out	Waiting for IO to complete	Running
47	"	"	Swapped IN Ready	"	Terminated

- P_7 is blocked at 23 because disk unit 3 is locked as P_1 is reading.
- At 23 P_1 & P_3 are wating due to IO (Reading & Writing) And at 37 IO is completed so they are ready for CPU
- At 47 P_5 is swapped In and IO (writing is completed) so it is Ready for CPU.
- At 37 disc unit is not locked so P_7 is now waiting for IO (writing is going on)

(2) 5 = bad
1 = perfect

a LRU - Least Reacently Used

If a page is brought long time ago and not currently being used will be replaced with ~~otto~~ new page.

~~LRU is~~ k ~~cop~~ 6
LRU is k- competitive algo.
i.e. if if LRU has k page ~~fault~~ fault, so on an ang, atleast one page fault will ~~occur~~ occur in optimal algorithm.
where k is size of page table.

∴ LRU will be ranked '2'

b FIFO - First in First Out.

The one which has come earliest in ~~table~~ table will be removed.

It is also k-competitive.

But if a page came earliest and still being used will be remand if a ~~page~~ fault occurs.

∴ LRU will be ranked '3'

(b) Optimal Replacement:

It will be the best Replacement Algo. As no other Algo can perform better than Algo

∴ It will be ranked '1'

(d) Second chance replacement.

We will wait until a page is used 2 ~~times~~ times.

If a page is used twice then ~~then~~ it can be replaced if page fault occurs.

But it is not better than LRU

∴ It will be ranked "3"

So,

(1) Optimal Replacement

(2) LRU

(3) FIFO + Second Page Replacement.

(3) 1 2 3 4 2 1 5 6 2 1 2 3 7

6 3 2 1 2 3 6

(a) LRU

Cases No. of Frames = 1 — 20 PF

Frame (2) = 2 — 18 PF

Frame = 3 — 15 PF

Frame = 4 — 10 PF

Frame = 5 — 8 PF

Frame = 6 — 7 PF

Frame = 7 — 7 PF

(b) FIFO

No. of Frame = 1 = 20 PF

= 2 = 18 PF

= 3 = 15 PF

= 4 = 14 PF

= 5 = 10 PF

= 6 = 10 PF

= 7 = 7 PF

(c) Optimal Replacement

No. of Years:

1	→	1 PF
2	→	15 PF
3	→	11 PF
4	→	8 PF
5	→	7 PF
6	→	7 PF
7	→	7 PF

(4)

(a) Yes it can happen.

In foo
semWait (S)
when it will go for R it will be locked by bar

Simultaneously

In bar
SemWait (R)
when it will go for S it will be locked by foo

It will wait for bar to release R

It will wait for foo to release S

∴ both will wait for eachother

∴ Both being blocked forever.

(b) NO.

It foo executes semWait (S);
SemWait (R);

faster than bar executes
semWait (R);

Then foo will be finished
release S & R after doing this
Then bar can start executing.

No indefinite postponment.

(Assumig disc cylinder from 1 to 100)

(5) 26 37 100 14 88 33 99 12

↑
head

(Sorted) 12 14 26 33 37 88 99 100

12 7

(a) FCFS

26 → 37 → 100 → 14 → 88 → 33 → 99 → 12

(11) (63) (86) (74) (55) (66) (87)

Total head movement = 11 + 63 + 86 + 74 + 55 + 66 + 87

= 442

(b) SSTF

26 → 33 → 37 → 14 → 12 → 88 → 99 → 100

(7) (4) (23) (2) (76) (11) (1)

Total head movement = 7 + 4 + 23 + 2 + 76 + 11 + 1

= ~~124~~ = 124

(c) SCAN (going up)

21 → 33 → 37 → 88 → 99 → 100 → 14 → 12

(7) (4) (51) (11) (1) (81) (2)

Total = 7 + 4 + 51 + 11 + 1 + 86 + 2 = 162

(d) C-SCAN (going up)

26 → 33 → 37 → 88 → 99 → 100 → 1 → 12

(7) (4) (51) (11) (1) (99) (11) (2)

14

Total = 7 + 4 + 51 + 11 + 1 + 99 + 11 + 2

= 186

(6) RPM = 15000 rpm

512 byte per sector

400 sector per track

1000 tracks

any seek time is 4 ms

file size = 1MB

(a) Time take in one rotation = $\frac{60 \times 1000}{15000} = 4$ ms

Rotational latency = $\frac{1}{2}$ rotation time = 2ms

400 × 512 Byte per track

No. of track = $\frac{1048576}{400 \times 512}$

(Assuming transfer time includes seek time & rotational latency)

Total time = 4 ms + $\frac{1}{4} \times \frac{1048576}{400 \times 512}$ ms. + 2ms

= 4 + 0.4 × 5.12 + 2ms

= 4 + 20.48 + 2ms

~~= 24.48 ms~~

= 26.48 ms

(b) Avg access time for a file = Seek time + Rotational latency

= 4 ms + 2ms = 6ms

(c) Rotational delay = 2ms :

(d) (a) Time to read 1 sector = 4ms $\times \frac{1}{\text{no. of sector}}$

$= 4 \times \frac{1}{400}$

$= \frac{1}{100}$ ms,

(e) Total time to read one track = 4 ms

(7)

13 direct pointers

1 indirect "

1 double indirect "

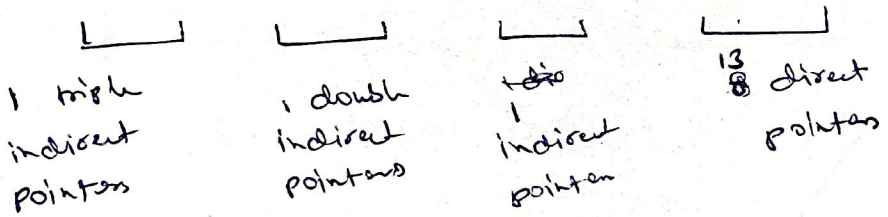
1 triple indirect "

32-bit pointer identifies one block of 8KB

Then,

Assuming direct pointers are 32-bit

13 direct pointers will identify $13 \times 8 = 104$ KB



∴ This can handle 104 KB file

P.T.O

(8) In Reflection attack attacker is using same challenge again.

To avoid this type of attack we need to make sure that same challenge can not be used more than once.

To do so we can add unique id & time stamp to the challenge.

Every challenge should be cross checked.

If that challenge has been used earlier then the attacker is trying to attack.

Now the authenticator will know when someone is trying to attack. So the connection can be closed.

This new protocol will be secured from Reflection attack.

P. T. O.

Another Soln

whenever a session is established between two parties an unique session id will also be generated. And each challenge will be generated using the session id. So when the attacker will send some challenge on different session then the session id will not match with the challenge. So that session will be closed and the attacker will get no response from that challenge.

∴ Reflection Attack will not be possible.