

①

What is this "waiting" state??

Time ↓

⑧

	P ₁	P ₃	P ₅	P ₇	P ₈
23	Waiting for IO to complete	Waiting for IO to complete	Suspended	Blocked	Running
37	Ready	Ready	Swapped out	Waiting for IO to complete	Running
47	"	"	Swapped IN Ready	"	Terminated

* P₇ is blocked at 23 because
disk unit 3 is locked as
P₁ is reading.

* At 23 P₁ & P₃ are waiting
due to IO (Reading & Writing)

And at 37 IO is completed
so they are ready for CPU

* At 47 P₅ is swapped in and
IO (writing is completed) so it is
ready & ~~wait~~ for CPU.

* At 37 disk unit is not locked
so P₇ is now waiting for IO
(writing is going on)

(2)

5 = bad
1 = perfect

a LRU = Least Recently Used

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

~~LRU is k comp~~

LRU is k -competitive algo.

i.e. if LRU has k page ~~fault~~ fault, so on an avg, atleast one page fault will ~~occur~~ occur in optimal algorithm.

where k is size of page table.

\therefore 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 removed if a page fault occurs.

\therefore (LRU) ^{FIFO??} will be ranked '3'

⑥

Optimal Replacement:

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

∴ It will be ranked '1'

④

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'

⑥

So,

①

Optimal Replacement ✓

②

LRU ✓

③

FIFO

+

Second Page Replacement.

↓
worst!

How about Belady's
anomaly ??

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 ✓

Pages (2) = 2

18 PF ✓

Pages = 3

15 PF ✓

Pages = 4

10 PF ✓

Pages = 5

8 PF ✓

Pages = 6

7 PF ✓

Pages = 7

7 PF ✓

b FIFO

No. of Frames = 1

20 PF ✓

= 2

= 18 PF ✓

= 3

= 15 PF X 16

= 4

= 14 PF ✓

= 5

= 10 PF ✓

= 6

= 10 PF ✓

= 7

= 7 PF ✓

(c) optimal Repliment

No. of Rooms =

11

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 each other

∴ Both being blocked forever

(b) NO.

If foo executes semWait (S)!
semWait (R);

faster than bar executes

semWait (R);

Then foo will be ~~finished~~

release S & R after done it

Then ~~Then~~ bar can start executing.

No indefinite postponement. ✓

12

(Assuming disc cylinder for 1 to 100)

(5)

26



head

(sorted)

37

100

14

88

33

99

12

12

14

26

33

37

88

99

100

12

7

(6)

FCFS

26 → 37 → 100 → 14 → 88 → 33

↓ 99
12 ← 87

$$\begin{aligned} \text{Total head movement} &= 11 + 63 + 86 + 74 + 55 + \\ &\quad 66 + 87 \\ &= 442 \end{aligned}$$

(b)

SSTF

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

(7)

(4)

(23)

(2)

(76)

(11)

(1)

$$\begin{aligned} \text{Total head movement} &= 7 + 4 + 23 + 2 + 76 + 11 + 1 \\ &= 124 \end{aligned}$$

(c)

SCAN (going up)

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

(7)

(4)

(51)

(11)

(1)

(82)

(2)

$$\text{Total} = 7 + 4 + 51 + 11 + 1 + 86 + 2 = 162$$

(d) C-SCAN (going up)

26 → 33 → 37 → 88 → 99 → 100 → 1 → 12
(7) (4) (51) (11) (1) (99) ↓ (2)
14

$$\begin{aligned} \text{Total} &= \underline{7} + \underline{4} + \underline{51} + \underline{11} + \underline{1} + \underline{99} + \underline{11} + \underline{2} \\ &= \underline{186} \end{aligned}$$

(10)

⑥ RPM = 15000 rpm

512 bytes per sector

400 sectors per track

1000 tracks

avg seek time is 4 ms

file size = 1 MB

① Time taken in one rotation = $\frac{60 \times 1000}{15000} = 4 \text{ ms}$

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

400 x 512 Bytes per track =

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

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

= $4 + 4 \times 5.12 + 2 \text{ ms}$

= $4 + 20.48 + 2 \text{ ms}$

~~= 24.48 ms~~

= 26.48 ms

② Avg access time for a file = Seek time + Rotational Latency

= $4 \text{ ms} + 2 \text{ ms} = \underline{\underline{6 \text{ ms}}}$

Assuming transfer time includes seek time & rotational latency

(c) Rotational delay = 2ms ✓

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

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

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

4

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

⑦

13 direct pointers

1 indirect "

1 double indirect "




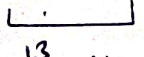
1 triple indirect "

32-bit pointer identifies one block of 8KB

Then,

Assuming direct pointers are 32-bit

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

			
1 triple indirect pointers	1 double indirect pointers	1 indirect pointers	$\frac{13}{8}$ direct pointers

\therefore This can handle 104 KB file

⑧ In Reflection attack ~~reflection attack~~

attacker is using same ~~attack~~ challenge again.

To avoid this type of ~~attack~~ attack

we need to make ~~sure~~ sure that - same

~~same~~ challenge can not be used ~~more~~ more than once.

✓ OK, but impractical

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

Every challenge should be cross checked.

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

Now the ~~attacker~~ authenticator will know when someone is trying ~~to~~ to attack. So ~~the~~ ~~can~~ ~~of~~ that ~~information~~ can be checked.

This new protocol will be secured from Reflection ~~attack~~ attack.

⑨

P. T. O,

Another Solⁿ

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

∴ Reflection Attack will not be possible.

