~~1~~

	P1	P3	P5	P7	P8
23	Blocked	Blocked	Ready ~~Suspended~~	Blocked	Ready
37	Ready ~~Suspended~~ ~~Suspended~~ ~~Ready~~	Read ~~Suspended~~ ~~Ready~~	~~Exited~~ Suspended	Blocked	Ready
47	Ready	Ready	Ready	Blocked	exited

1

12

	P1	P3	P5	P7	P8
23	Blocked ✓ read from disk unit 3	Blocked read from disk unit 2 ✓	Ready ✓	Blocked ~~write~~ write disk 3 ✓	Ready ✓
37	Ready ✓	Ready ✓	Suspended ✓	Blocked write disk 3 ✓	Ready ✓
47	Ready ✓	Ready ✓	Ready ✓	Blocked write disk 3 ✓	Exited ✓

Assuming that only one process can execute at a time and for the unknown times (like 23, 37 & 47) we don't know if the process is executing, so we assume it to be in ready queue. Also, I assumed that a process is swapped therefore it was not being accessed frequently i.e. it was suspended.

(2) a) LRU

Rank - 4. (A worst algo can be designed by looking at future, so 5 not given)

b) FIFO

Rank - 2 , Belady's anomaly observed.

c) Optimal replacement

Rank - 1

d) Second - chance replacement

Rank - 3

Assuming, as rank increase page fault rate increases

(3) a) LRU

No of Frames	No of faults
1	20 ✓ (All page faults)
2	18 ✓
3	16 ✗ 15
4	9 ✗ 10
5	8 ✓
6	8 ✗ 7
7	7 ✓

b) FIFO

~~No.~~ No of Frames	No of faults
1	20
2	18
3	16
4	12 × 14
5	10
6	10
7	7

10

c) OPT

No of frame	No of faults
1	20
2	14 × 15
3	11
4	8
5	7
6	~~8~~ 7
7	7

5 a) FCFS (Fist come first serve)

Head movements =

(26-26) + (37-26) + (100-37) + (100-14) + (88-14) + (88-33) + (99-33) + (99-12)

= 442

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

b) SSTF (Shoortest seek time First)

head movements

$= (26-26) + (33-26) + (37-33) + (37-14)$
$+ (14-12) + (88-12) + (99-88) + (100-99)$
$= 124$ (26 → 33 → 37 → 14 → 12 → 88 → 99 → 100)

---

c) SCAN (Assuming 100 is last cylinder)

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

head movements

$= (26-26) + (33-26) + (37-33) + (88-37)$
$+ (99-88) + (100-99) + (100-14) + (14-12)$
$= 162$ (Assuming 100 is last & 0 is first cylinder)

---

d) C-SCAN

12

26 → 33 → 37 → 88 → 99 → 100 → 0 → 12 → 14

$= (26-26) + (33-26) + (37-33) + (88-37)$
$+ (99-88) + (100-99) + (100-0) + (12-0)$
$+ (14-12)$
$= 188$

(6) rotational speed (ω) = 15,000 rpm.

# of bytes per sector (n_b) = ~~400~~ 512 bytes.

sectors per track (n_s) = 400

tracks = 1000 (n_t)

avg seek time = 4 ms (V)

File size (S) = 1 ~~M~~B = 1,048,576 Bytes

a) No of sectors required (n_r)

$$= \frac{1,048,576}{512} = 2048 \text{ sectors.}$$

Transfer time

= Rotational delay + Total seek time

$$= \cancel{\left(\frac{2048}{400} \times \frac{1}{\omega}\right) min + \left(\left[\frac{2048}{400}\right] \times\right.}$$

$$= \left(\frac{2048}{400} \times \frac{1}{\omega}\right) min + \left(\left[\frac{2048}{400}\right] \times V\right) ms$$

$$= (5.12 \times (15,000)^{-1}) min + (5 \times 4) ms$$

$$= (0.0205)\, s + (0.02)\, s$$

$$= 0.0405 \text{ seconds}$$

b) Average access time is the total seek time = 0.02 seconds

c) Rotational delay = 0.~~0~~205 seconds

d) Total time to read 1 sector

$= \left(\frac{1}{400} \times \frac{1}{w}\right)$ min.

$= \left(\frac{1000}{400} \times \frac{60}{15,000}\right)$ ms

3

$= 0.01$ ms

(Assuming disk read head was positioned at the sector, only rotational latency is involved)

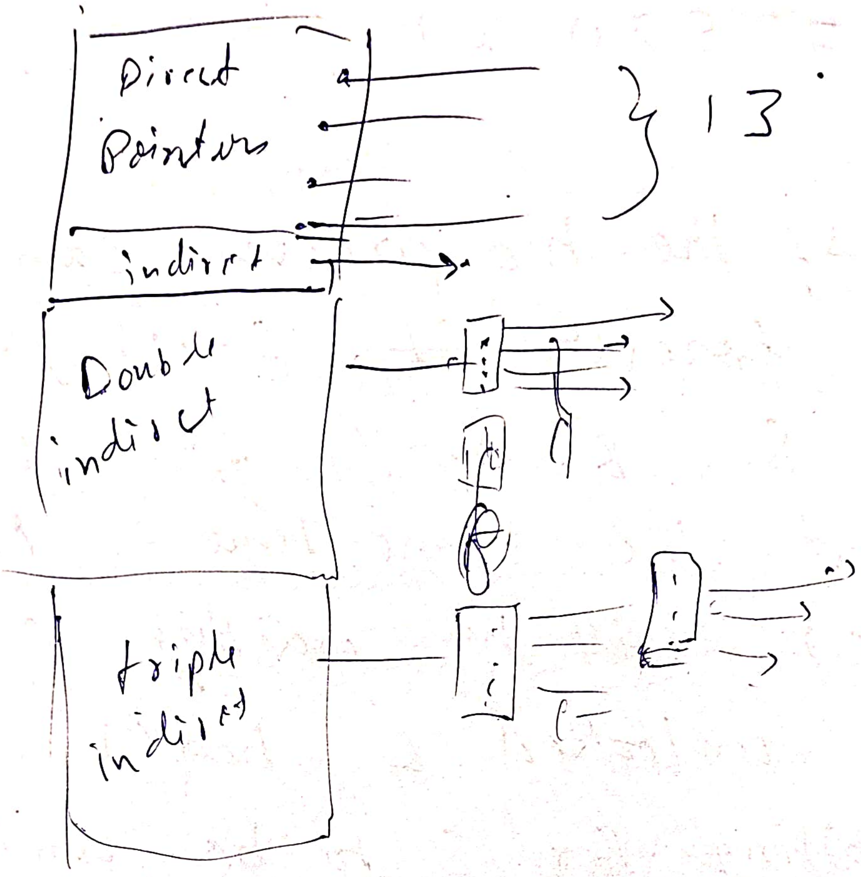
(e) Total time to read one track.

$= \left(\frac{1}{\omega}\right)$ min

$= \left(\frac{1000 \times 60}{15,000}\right)$ ms

$= 4$ ms

(7) i-node UNIX



One block = 8 KB (s)

Total size of file the i-node can handle

= (Due to direct pointers) + (Due to indirect pointer × Direct pointer)

+ (Due to double indirect × Due to indirect × Due to direct pointers)

+ (Due to triple indirect × Due to double indirect × indirect × Direct pointers)

$= 13s + (13 \times 13)s + (13 \times 13 \times 13)s + (13)^4 s$

$= 13(1 + 13 + 13^2 + 13^3)s = 30{,}940\ s$

= 30,990 × 8 KB
= 247,520 KB

(4) (a) Yes, the two process can be blocked forever if foo() gains lock on S & bar() gets lock on R at the same time. Now, foo() will be waiting for R to be unlocked & bar() will be waiting for S to be unlocked forver. Neither of them would get ~~any~~ lock on both S & R & thus block each other forever.

12

(b) No, ~~if~~ a single process will not be postponed while other is running. Because if one process is running this means it has lock to both semaphore S & R & ~~the~~ other process is waiting. When the locks are released. The other process starts executing. So, they ~~could~~ may both get blocked, but, one process can't indefinitely postpone the other process.

(8) As described in reflection attack an attacker takes advantage of a challenge-response authentication and makes the target to answer his own challenge & sends the target response of correct authentication.

To guard against such an attack, we have to use a better authentication mechanism like using public & private keys. The attacker can't simply use ~~us to aut~~ the target to authenticate them. Also, the function used to process the keys are very complex & non-invertible, so it will provide more secure authentication.

(6) (8)

Other simple method can be that users will not accept a challenge that ~~it~~ the user has sent itself. Requiring need of a large number of challenges.

This is impractical & can lead to DoS

Yet, another protection strategy can be to use different protocols at sender & receiver side. So, ~~same~~ sender side's response will not be accepted on the receiver side.

(2) a) LRU (Least recently used)
It is not the worst because we can devise an algo which results in max page faults by looking at future.
~~It can have rank either 2~~
Rank = 4

b) FIFO (First in first out)
It is a frequently used algo, but, not as good as OPT
Rank = 2

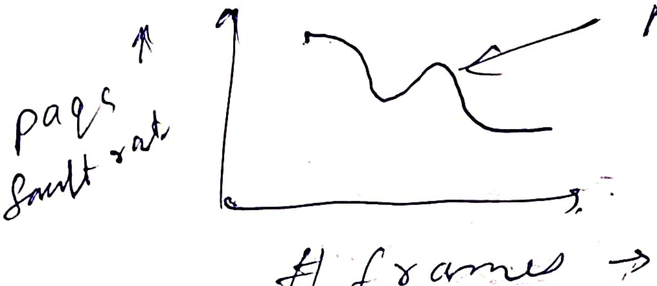
c) OPT
It is the best algo which looks at future of page requests & makes optimal page replacement.
Rank = 1

d) Second-change

It is an improved version of LRU

∴ Rank = 3

Belady's anomaly



peaks even though # frames increased.

4

Belady's anomaly is observed in FIFO.

Note :- FIFO can also perform worse than LRU in certain senarios, since the performance of these algo (except OPT) is heavily dependent on the actual sequence of requests.

How about others ??
Don't they have Belady's anomaly ??