MANOJ MALIPEDDI

6th march

3 type of I/O's

- program driven I/O (during I/O cpu is unused)
- Interrupt driven I/O

↓ will change

(30 units to send request)

state from blocked/ready → run

States a program can take

- ready
- block
- run.

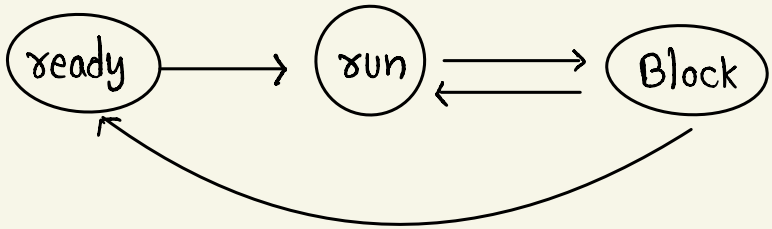
Suspend — suspend block.

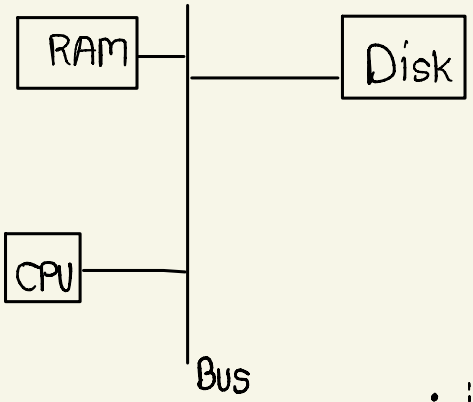
└ suspend ready.

Job	Service	Direct memory Access		Programmed I/O	
		Interval	State	interval	State
A	(40 run) (80 I/O) (90 run)	(0-40) (40-120) cpu disk (120-210) cpu	(0-40) run (40-120) blocked (120-210) run	0-210 cpu 40-120 disk	(0-40 run) (40-120 block) (120-210 run)
B	(60 run)	(40-100) cpu	(0-40) ready (40-100) run	210-270	(0-210 ready) (210-270 run) CPU 40-120 pooling

Job	Service	Interrupt driven I/O	
		interval	state
A	(40 run) (80 I/O) (90 run)	(0-70) cpu (70-120) disk (130-220) cpu	(0-40) run (40-120) blocked (120-130) ready (130-220) run
B	(60 run)	(70-130) cpu	(0-70) ready (70-130) run

State transition diagram



- 

- in DMA requests are written on to ram
- in DMA if disk, cpu want to use bus, disk uses bus in cycle stealing mode.
  (whenever cpu is not using bus, disk uses bus to carryout I/O).

• Total memory - 100, SRTN, swapin time 5, swapout - 7

Job	Arrival	Memory req.	service	interval
P	0	40	50	0-10, 30-70
Q	10	40	15	10-18, 23-30
R	18	40	5	18-23

(memory issues)

interval
0-10, 30-70
10 - 25
25 - 30

(swap-in, swap-out is avoided because there is no advantage)

Job	Arrival	Memory	Service	INTERVAL		
					out	in
P	0	40	50	(M0 - 39) 0-10, 35-75.	18-25	30-35
Q	10	40	16	(M40 - 79) 10-25, 30-31		
R	18	40	5	(M0 - 39) 25-30		

(25)
(5)
(7)
(37)

less average waiting time ⇒ (29)

0-10, 31-71	(21)
10-26	
26-31	(8)

Job	Arrival	Memory	Service	INTERVAL			wait
					out	in	
A	0	40	50	0-10, 18-25, 30-35, 43-71			8+5+8=21
B	10	60	16	10-18, 35-43	18-25	30-35	17
C	18	60	5	25-30			7

A: 0-10, 31-71 ; B: 10-26 ; C - 26-31 (29) (35)

– less average waiting time

Job	Arrival	Memory	Service	INTERVAL		
					out	in
U	0	60	50	0-12, 59-97	12-19	24-29
V	10	40	42	12-19, 24-59		
W	12	60	5	19-24		

(35)

Job	Arrival	Memory	Service	INTERVAL		
					out	in
P	0	80	50	0-18 65-97	18-25	60-65
Q	10	80	30	30-60		
R	18	80	5	25-30		

PRQP - service order

7th March

Page replacement algorithms:

- FIFO: first in first out
- LRU: Least recently used
- optimal
- Second chance (capacity = 4)
- Approximate LRU.

Need	FIFO	LRU	optimal	implement LRU	second chance	Approximate LRU
G	G f	f		G	G_r	G_r
M	GM f	f		GM	$G_r M_r$	$G_r M_r$
T	GMT f	GMT f		GMT	$G_r M_r T_r$	$G_r M_r T_r \rightarrow G_h M_h T_h$
M	GMT	GTM		GTM	$G_r M_r T_r$	$G_h M_r T_h$
H	GMTH f	GMTH f	GMTH	GTMH	$G_r M_r T_r H_r$	$G_h M_r T_h H_r \rightarrow M_r T_h H_r$
U	MTHU f	MTHU f	GMTU	TMHU	$M_r T_h H_r U_r$	$M_r T_h H_r U_r \rightarrow M_h T_h H_h U_h$
T				MHUT	$M_r T_r H_r U_r$	$M_h T_r H_h U_h$
G	THUG f	THUG		HUTG	$T_r H_r U_r G_r$	M, H or U victim.
M	HUGM f	TUGM		UTGM	$U_r G_r T_r M_r$	
K	UGMK f	TKGM		TGMK	$M_r U_r G_r K_r$	

Victim : removed element

Fault : Disk → Ram

New object brought

FIFO :

8 faults ⟶ capacity (4)

6 faults ⟶ capacity (5)

- Give a smallest sequence such that

first victim is

H in FIFO

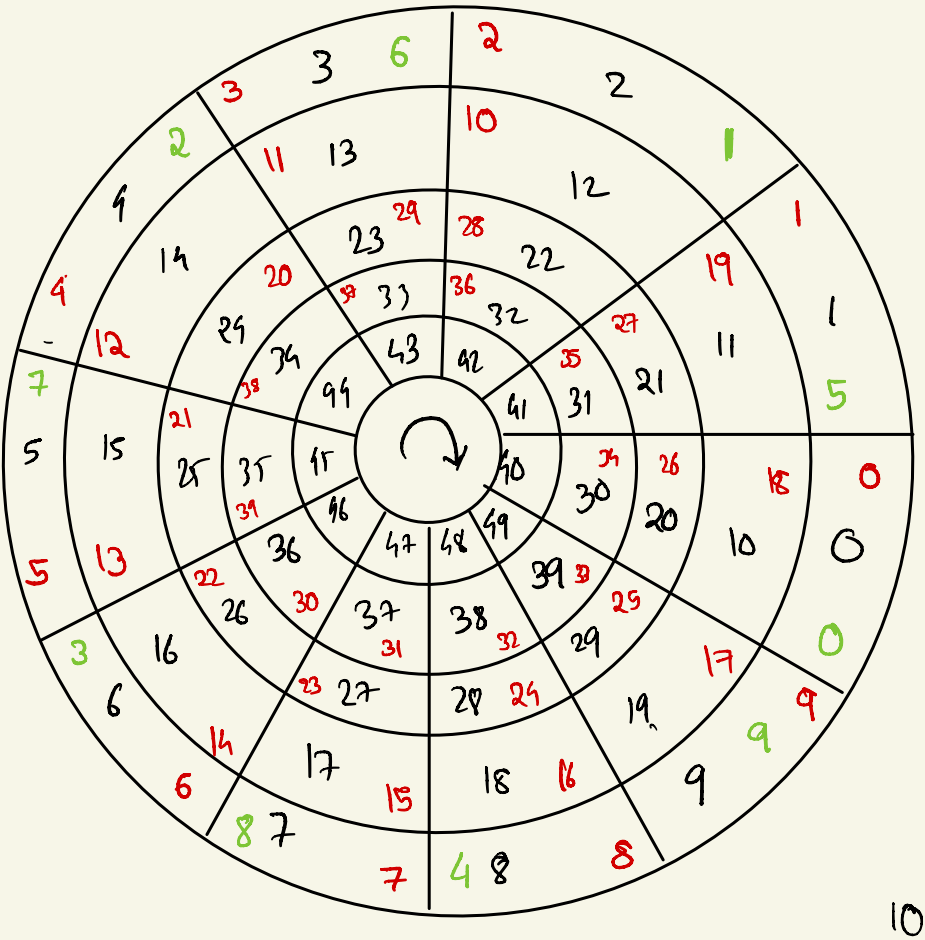
U in LRU

T in optimal

HUHTKHU

14th March.

- - hardware numbering
- - logical block number.



10
+6

10unit / revolution

3unit / track

2units : Disk → ram.

At t=0, head is at block-0.

- how and when block 6 can be accessed, → do nothing at t=6, head on block 6.

- how to access block 17

(radial head mov.) → at $t = 3$: head is on block 13
30

$t = 7$: head is on block 17.
(70)

↳ seek time.

- to access block 12 :

12 sec required.

Access 29, 10:

Move head 2 units left

at $t = 6$, wait for 3 → (head will be on 29) ($t = 9$)

Move head 1 unit right

at $t = 12$ head is on block 12

wait for 8

$t = 20$: head on 10

Access 10, 29

1 unit left

t=3, head on 13

wait 7

t=10 : on 10

1 unit left

t=13 : head on 23

wait for 6

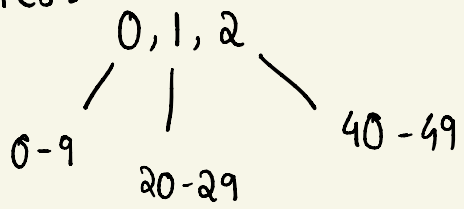
t=19: head on 29. (19)

Access 0-9, 12, 13, 14, 15. - 15

0 - 13 - 21

- – numbering is given after taking D→R time under consideration.

File transfer:



?

---

5 units D→R, 7 units R→D, 30 units on disk access.

Job	Service	
A (0)	(40 run) (h = 42 9) (30 run)	(0-40 cpu) (40-54 Disk) (54-84 disk operation) (84-119 CPU)
B (15)	(35 run)	(54-89 cpu)

(15-54) ready, (54-89) run

Job	Service	
P	(40 run) (Print (m)) (30 run)	(0-40 cpu), (40-47 → disk) (47-77 Disk operation) (82-87) Disk → ram, (87-117) run
Q	(35 run)	(47-82 run)

20th March,

Job P: output ABCD

Job Q: output GHIJ

Possible outputs:

AGHBICDJ

ABGCHIDJ

total number of possible outputs = $\frac{8!}{4! \cdot 4!}$

When Job P is executing P's critical section, Job Q should not execute Q's critical section.

Critical: BC HI

atomic		critical
not permitted	AGBHICDJ	– Not permitted
permitted	AGHIBCDJ	– permitted.
not permitted.	ABGCHIDJ	– permitted

UVW
KLM
$\frac{6!}{3!.3!} = 20$

if VW, KL → atomic - $\frac{4!}{2!.2!} = 6$

if VW, KL
→ critical

KLUVMW is allowed.

Solutions to deal with critical section:

Mutual exclusion:

	Time taken
Initial F is 0	
while (F == 1) wait;	5
F == 1;	3
critical section	
F = 0;	7

- P: A(22) B(17) C(14) D(9)

Q: G(14) H(40) I(6) J(42)

	0-22 A		0-14G	
while	22-27		14-19	while
while	27-32		19-22	F=1
	⋮		22-62	(H)
			62-68	(I)
while	72-77		68-75	F=0
while	77-82		75-117	(J)
F=1	82-85			
(B)	85-102			
(c)	102-116			
(D)	116-125.			

Flag should be 0 at the start of while loop to start critical section.

P: A(22) B(17) C(14) D(9)

Q: G(24) H(40) I(6) J(42)

	0-22 A	0-24 G
while	22-27	24-29 while
F=1	27-30	29-32 F=1

(Mutual execution is Failed)

0-22 A	0-24 G
22-32 TAS	~~24-34 TAS~~
	32-42

only one process can use TAS

Test and set

```
F=1
while(F==1) wait;
```

```
U = TAS(F)
CR Section
F=0
```

TAS - returns old flag.

mutual GHO

Job P	Job Q
P = 1	Q = 1
while (Q == 1) wait;	while (P = 1) wait;
Critical	Critical
P = 0	Q = 0

Job P: (A) P = 1 while (Q == 1) wait; (B) (C) P = 0 (D)

Job Q: (G) Q = 1 while (P == 1) wait; (H) (I) Q = 0 (J)

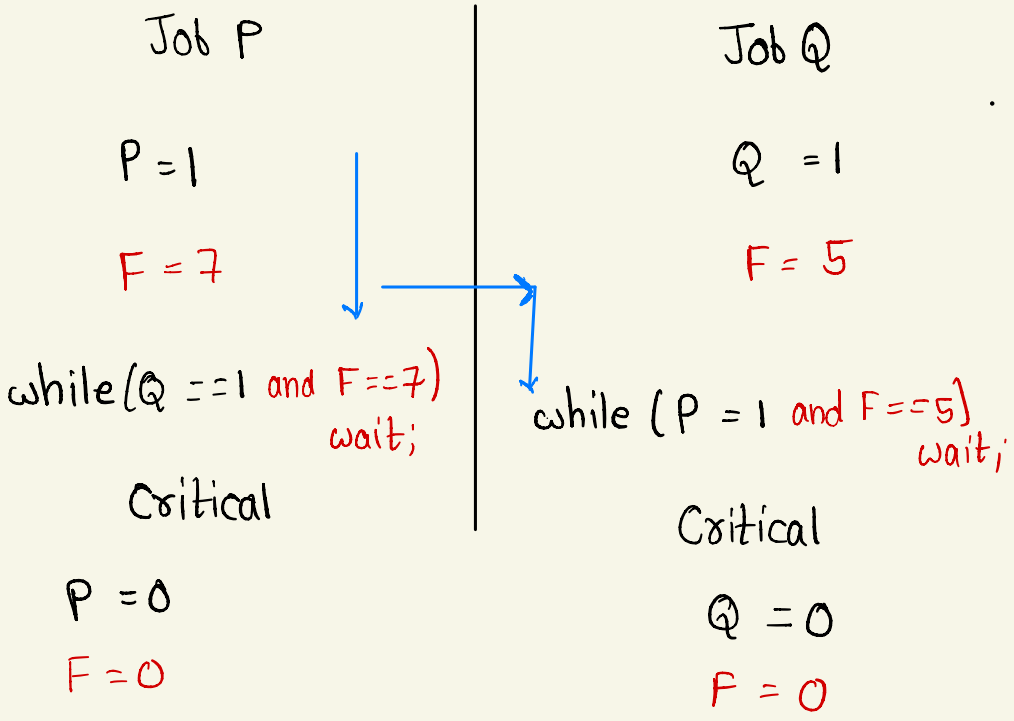
$t(\text{while } (P==1)\ \text{wait;}) > t(Q=1)$ > $t(\text{while } (Q==1)\ \text{wait;}) > t(P=1)$

always always

$t(\text{while } (P==1)\ \text{wait;}) > t(Q=1)$

(mutual exclusion with deadlock)

mutual GHO



without loss of generality : $t(F=5) > t(F=7)$

t = 0 : P = 1

t = 100 : F = 7

t = 200 : while (Q == 1) && F == 7 wait

.

.

.

t = 2000 F = 5

CR

t = 3000 while (P == 1) && F == 5 wait;

GENERAL

own = 1

F = self

while (opponent == 1) and

(F == self) wait;

Critical

F = 0

own = 0.

Job P	Job Q
F = 7	F = 5
P = 1	Q = 1
while (Q == 1) and (F == 7) wait;	while (P == 1) and (F == 5) wait;
Critical	Critical
P = 0	Q = 0

Prove mutual exclusion fails,

F = 7	
	F = 5
	Q = 1
	while (P == 1 and (F == 5)
P = 1	
while (Q == 1) and F == 7	critical
Critical	

mutual exclusion is failed in this case.

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
a[i]	23	78	42	64	37	95	81	36	43	79	26	31	47	23	87	35	29	71	36	49
b[i]	26	31	47	23	87	35	29	71	36	49		$h_B = 0$								
c[i]	49	09	89	87	24	30	20	07	79	28		$h_c = 0$								

Redundant assay of independent disk (RAID)

operation		Two disks RAID (0)	time	on hA=0	disk time
a[18]		hA=0, hB=8	8	18	18
a[9]		hA=4, hB=8	4	4	14
a[12]		hB=2	6	12	8

RAID (1)	Time	operation	
$h_B = 8$	8	a [18]	
$h_A = 4$	4	a [4]	
$h_A = 2$ $h_C = 2$	2	a [12]	

$(74+79+27+36) \% 100$

74, 79, 27, 36, 16

- we can correct error only if we know the point of error in this method.

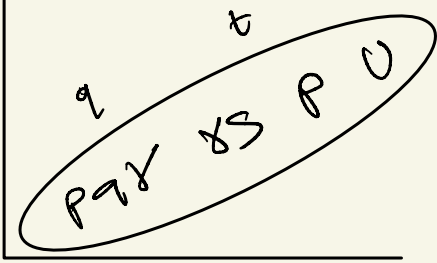
P	q	r	S	(P+q+r)	(P+q+S)	(P+r+S)	(q+r+S)
74,	79,	27,	36	80	89	37	42
74,	~~79~~ 32,	27,	36	W	W	C	

P	q	γ	S	pqγ	pqs	pγs	qγS
74	79	27	36	C	C	C	C
q wrong				W	W	C	
γ wrong				W	C	W	
S wrong				C	W	W	
P wrong				W	W	W	

since it is unique for different errors

we can detect error in 4 element array using 3 additional elements.

Simillarly 5 element array errors can be detected using 4 elements.



p q r s t u	p s t	p q u	q r s	r t u
p	ω	ω	c	c
q	c	ω	ω	c
r	c	c	ω	ω
s	ω	c	ω	c
t	ω	c	c	ω
u	c	ω	c	ω

27th march

Concurrency control:

Job A: [Arrive 5] [run 10] [P = 29] [run 20] [Q = 36] [run 20]

Job B: [Arrive 7] [run 5] [P = 32] [run 40] [Q = 42] [run 10]

[35 - 53 wait] [Q = 36] [53 - 73]

A: [5 - 15] [P = 29] [15 - 35] [Q = 36] [35 - 55]

B: [7 - 12] [P = 32] [12 - 52] [Q = 42] [52 - 62]

P = 29, Q = 36
Finally.

t = 12 : P = 32 BA B → A

t = 15 : P = 29

t = 35 : Q = 36

t = 42 : Q = 42

not permitted.

Final value P = 29
Q = 42

both P,Q values
should be
From same program

Job D: [Arrive 5] [run 9] [R=91] [run 10] [S=36] [run 20]

Job E: [Arrive 7] [run 10] [S=29] [run 5] [R=36] [run 10]

Job D: [5-14] [R=91] [14-24] [S=36] [24-41]

Job E: [7-17] [wait 17-25] [5-29]

t=14: D → E

at t=17

Job A: [Arrive 5] [run 10] [P=29] [run 20] [Q=36] [run 20]

Job B: [Arrive 7] [run 5] [P=32] [run 40] [Q=42] [run 10]

Job C: [Arrive 39] [run 10] [M=61] [run 10] [Q=96] [run 10]

[35 - 39 wait] [Q=36] [39 - 69]

A: [5 - 15] [P=29] [15 - 35]

B: [7 - 12] [P=32] [12 - 52] [Q=42] [52 - 62]

C: [39 - 49] [M=61] [49 - 59] [Q=96] [59 - 69]

instead of waiting upto 53 wait is upto 39 because new path is seen

ABC : P=32 , Q=96

BAC : P=29, Q=96

ACB : P=32, Q=42

t=12 P=32

t=15 P=29

t=39 Q=36

Job A: [Arrive 5] [run 10] [P=29] [run 20] [Q=36] [run 20]

Job B: [Arrive 7] [run 5] [P=32] [run 40] [Q=42] [run 10]

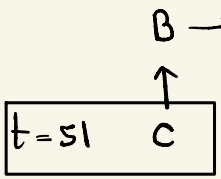
Job C: [Arrive 39] [run 10] [M=61] [run 2] [Q=96] [run 10]

A: [5-15] [P=29] [15-35] [35-39 wait] [Q=36] [39-59]

B: [7-12] [P=32] [12-52] [51-53 wait] [Q=96] [53-63]

C: [39-49] [M=61] [49-51] ~~[Q=96] [59-69]~~

at t=12



→ kept under wait

at t=15 BA

at t=51 last B attempt made.

Job A: [Arrive 5] [run 10] [P=29] [run 20] [Q=36] [run 20]

Job B: [Arrive 7] [run 5] [P=32] [run 40] [Q=42] [run 10]

Job C: [Arrive 39] [run 10] [M=61] [run 2] [P=96] [run 10]

A: [5-15] [P=29] [15-35] [wait 35-39] [Q=36]

B: [7-12] [P=12] [12-52] [Q=42]

C: [39-49] [M=61] [49-51] [P=96]

at $t=15$

B → A Fixed.

at $t=35$

A → B attempt made

at $t=39$

C → A

Job G : (arrive 5) (Y=61) (run 30) (Q=36)

Job H : (arrive 17) (M=Y) (run 10) (Q=49)

G : (t=5 Y=61) (5-35) [Q=36]

H : (t=17 M=61) (17-27) (27-36 wait) [Q=49]

at t=17
H after G Fixed.

Job G : (arrive 25) (Y=61) (run 30) (Q=36)

Job H : (arrive 17) (M=Y) (run 100) (Q=49)

G : (t=25)(Y=61) [25-55] [55-118 wait] (Q=36) at t=25,
H before G
Fixed.

H : (t=17 M=old) [17-117] (Q=49)

Job U : (arrive 25) (Z = 19) (run 30) (I = k)

Job V : (arrive 17) (J = Z) (run 100) (k = 36)

at t = 25 V → U

A : (run 5) [critical] [run 20] [No critical] [run 10]

B : (run 7) [critical] (run 40) [No critical] [run 5]

initial i = 0

P and V - semaphores

P: i = i+1; if (i>1) { Q[i] = id; wait }

V: if (i≠1) signal (Q[i]), i = i-1;

A : (0-5 run) (P) (5-25) V (25-35)

B : (0-7 run) (P) (25-65) (V)

at t=25 : signal Q[2]

at t=5 : i=1

at t=7 : i=2 Q[2]=B

at t=65 i=0

A : (run 5) (CR) (run 10) (NCR) (run 7)

B : (run 2) (CR) (run 20) (NCR) (run 5)

C : (run 30) (CR) (run 12) (NCR) (run 5)

D : (run 12) (CR) (run 15) (NCR) (run 10)

A: [0-5] (at t=5, i=2, Q[2]=A)

B: [0-2] (at t=2, i=1) (2-22) (at t=22 signal (Q(3)) i=2)

C: [0-30] (at t=30, i=3, Q[3]=C)

D: [0-12] (at t=12, i=3, Q[3]=D) (22-37 run)

at t=37 i=2, signal (Q(3))