Name : Ananya Prasad
Reg No : 20BCE10093
EXAM : CSE3003 / TEE

Faculty : Dr Abha Trivedi Ma'am
Semester : Fall sem 2021-22
Batch / Date : B21+B22+B03 / 6 Jan, 2022

5

1. ①

Operating system is a system software that has access to all hardware devices and memory.

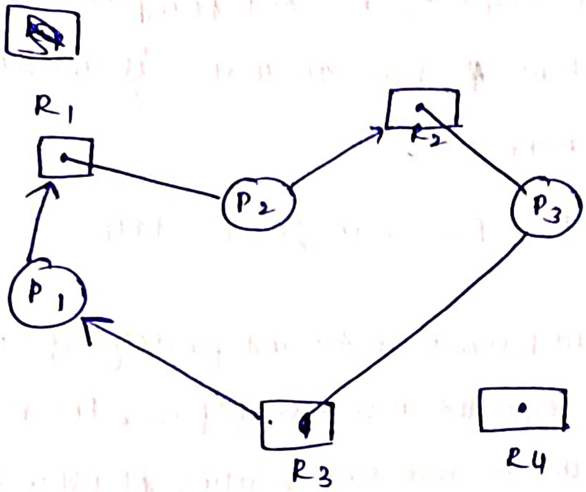
- It functions in such a way that it hides all the backend ~~raw~~ complex work with the hardware.
- It manages the function of hardware for many applications following some pre-decided policies.
- It make sures that the processes are different and dont merge up to hamper function

We can compare an operating system to that of functioning of a toy-shop as it directs the operations and manage the resources.
They also have a set of pre-defined policies for work
Different tasks are alloted differently and not merged up.
So, similarities →

~~OS~~ OS	Toy shop manager services
• Control of CPU, memory, hardware.	• control of workers, stock, machenary, logistics, ~~...~~ etc.
• Has policies ~~...~~ to access to data and communciation, operations on I/O, resource allocation etc.	• Has policies for worker safety, fair work etc
• Has the tasks for system calls to execute complex tasks	• Has the task to break operations and assign so that complex tasks simplify to increase shop performance.

2a

RAG cycle and deadlock



converting this to matrix →

Allocation Matrix

	R_1	R_2	R_3	R_4
P_1	0	0	1	0
P_2	1	0	0	0
P_3	0	1	0	0

Request Matrix

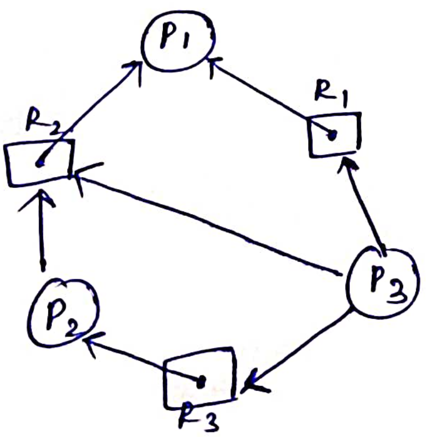
	R_1	R_2	R_3	R_4
P_1	1	0	0	0
P_2	0	1	0	0
P_3	0	0	1	0

→ [0 0 0 1] avail

avail * [1 1 1 1]

∴ It cannot process any processes. So it is in UNSAFE state.
This is a dead lock.

RAG cycle and NO deadlock



Converting to matrix,

Allocated

	R_1	R_2	R_3
P_1	1	1	0
P_2	0	0	1
P_3	0	0	0
	1	1	1

[1 1 1]
R_1 R_2 R_3

Request

	R_1	R_2	R_3
P_1	0	0	0
P_2	0	1	0
P_3	1	1	1

3

Here, P_1 has the hole of R1 and P_2, so it completes its process and release R_1 and R_2. P_2 now requests R_2 and completes the process and release R_2 and P_3. Now P_3 has all that it needs and executes. So, no deadlock occurs here.

Process sequence ⇒ P_1 P_2 P_3, in SAFE state.

Starvation : A thread would have to wait indefinitely if other threads keep coming and getting the requested resources. before. The resource is used by multiple threads but a thread will stop waiting if other threads stop coming in as the resource is then free.

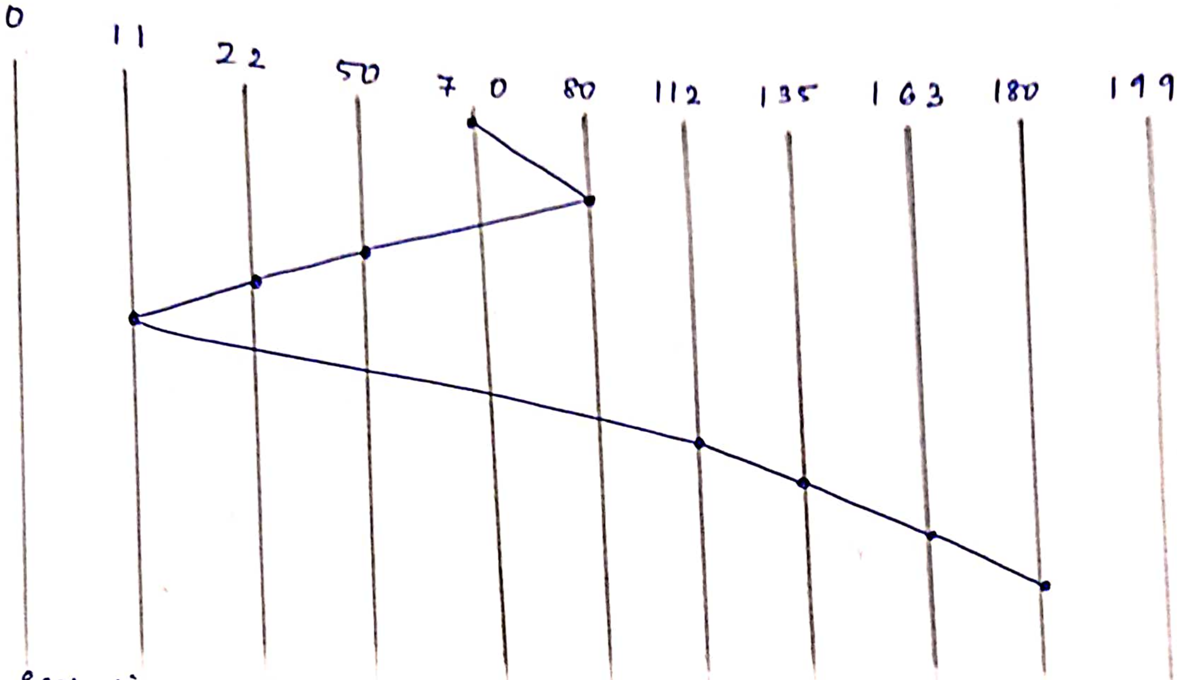
In Deadlock : A group of threads wait for resources held by others in the group. as no one is completing the task, ~~there~~ there are no movements at all. They become mutually exclusive and are waiting forever, held in a cycle of sorts. So deadlock is more critical than starvation.

80, 180, 22, 163, 112, 50, 11, 135

⇒ ascending order → 11, 22, 50, 80, 112, 135, 163, 180

Head → 70 , 0 - 199.

SSTF.

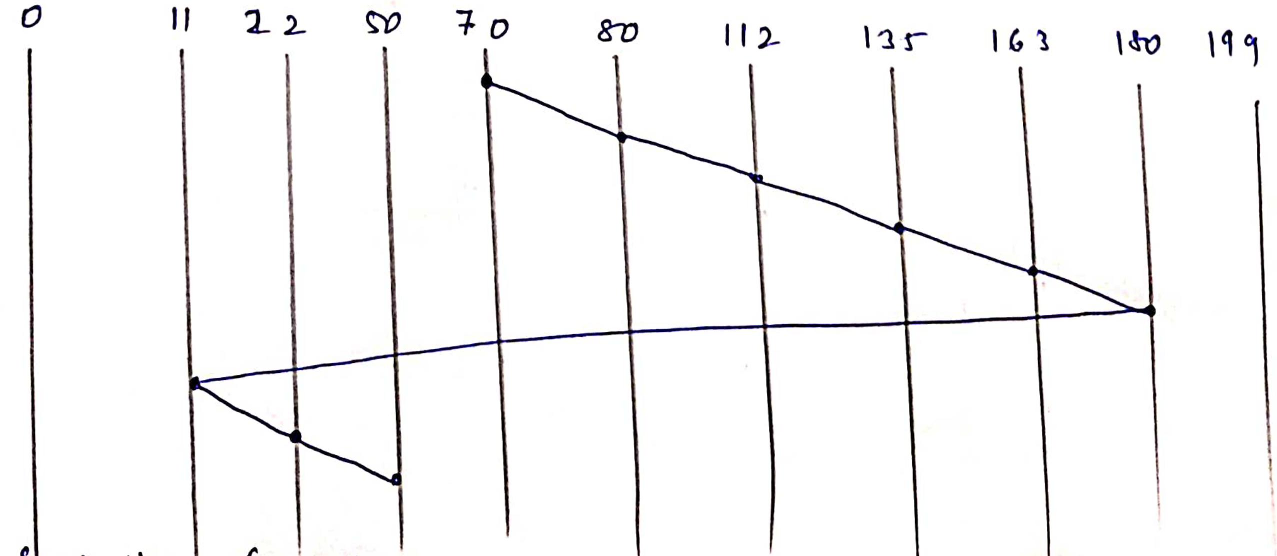


Seek time = (80 - 70) + (80 - 50) + (50 - 22) + (22 - 11) + (11 - 112) + (135 - 112) + (163 - 135) + (180 - 163)

= 10 + 30 + 28 + 11 + 101 + 23 + 28 + 17

= 248

C LOOK



Seek time = (80 - 70) + (112 - 80) + (135 - 112) + (163 - 135) + (180 - 163) + (180 - 11) + (22 - 11) + (50 - 22) = 10 + 32 + 23 + 28 + 17 + 169 + 11 + 28

= 318

Process	Max			Allocation			Need			Available		
	A	B	C	A	B	C	A	B	C	A	B	C
P_0	3	2	2	1	0	1	2	2	1	2 2	1 0	3 2
P_1	4	5	1	1	3	0	3	2	1	0 2	0 2	1 7
P_2	2	1	6	0	1	4	2	0	2	3	2	8
P_3	8	0	3	4	0	2	4	0	1	4	5	8
P_4	0	7	4	0	2	1	0	5	3	8	5	10
	Total consump.			6	6	8						

available = 2 1 3 → $\boxed{8 \quad 7 \quad 11}$ → (A) total avail

P_0 not possible, P_1 not possible

P_2 executes →
```
   2  1  3  → avail
 - 2  0  2
 ---------
   0  1  1
```

P_2 releases →
```
 + 2  1  6
 ---------
   2  2  7  → avail
```
~~granted~~

P_3 ×, P_4 ×

P_0 executes →
```
 - 2  2  1  → granted
 ---------
   0  0  6
```

P_0 releases →
```
 + 3  2  2
 ---------
   3  2  8  → avail
```

P_1 executes →
```
   3  2  1  → grant
 ---------
   0  0  7
```

P_1 releases →
```
 - 4  5  1
 ---------
   4  5  8  → avail
```

P_3 · takes place → 4 5 8 → avail

− 4 0 1 → grant

0 5 7

P_3 releases → + 8 0 3

8 5 10 → avail

P_4 takes place → − 0 5 3 → grant

8 0 7

P_4 releases → 0 7 4

8 7 11

(C) ∴ safe sequence = P_2 P_0 P_1, P_3, P_4

(A) A = 8, B = 7, C = 11

(B) NEED MATRIX →

→	A	B	C
P_0	2	2	1
P_1	3	2	1
P_2	2	0	2
P_3	4	0	1
P_4	0	5	3

5

⇒ 3 2 1 3 4 1 6 2 4 3 4 2 1 4 5

LRU ⇒ Replace a page which is not used for a long time.

Pages	3	2	1	3	4	1	6	2	4	3	4	2	1	4	5
f1	3	3	3	3	3	~~3~~	6	6	~~6~~	3	3	~~3~~	1	1	1
f2		2	2	~~2~~	4	4	~~4~~	2	2	2	2	2	2	~~2~~	5
f3			1	1	1	1	1	~~1~~	4	4	4	4	4	4	4
	X	X	X	✓	X	✓	X	X	X	X	✓	✓	X	✓	X

MM

* Page faults = 10
* Hit ratio $= \frac{5}{15} = \frac{1}{3}$
* Miss ratio $= \frac{10}{15} = \frac{2}{3}$

OPTIMAL ⇒ Replace the page not used in near future

Pages	3	2	1	3	4	1	6	2	4	3	4	2	1	4	5
f1	3	3	3	~~3~~	4	4	4	4	4	4	4	4	4	~~4~~	5
f2		2	2	2	2	2	2	2	2	2	2	~~2~~	1	1	1
f3			1	1	1	~~1~~	6	6	~~6~~	3	3	3	3	3	3
	X	X	X	✓	X	✓	X	✓	✓	X	✓	✓	X	✓	X

MM

Page faults = 8

Hit ratio $= \frac{7}{15}$

Miss ratio $= \frac{8}{15}$