OS Tutorial Assignment

Page No.
Date: / /

Name: Atharva Salitri Div: CSAI - B
Roll no: 37 PRN: 12310120

Q1. Know your Operating system
→ An operating system (OS) is an interface between a computer user and computer hardware. An operating system is a software which performs all the basic tasks like file management, memory management, handling input and output, and controlling peripheral devices such as disk drivers and printers. The software that contains the core components of the operating system is called the kernal. Some popular OS include Linux, Windows and MacOS.

Q2. Elaborate about the different types of OS
→ 1. Batch OS : a type of OS that does not interact with the computer directly. There is an operator who takes similar jobs having the same requirements and groups them into batches.
2. Time-Sharing OS : a type of OS that allows many users to share computer resources for the maximum utilization of the resources.

3. Distributed OS : a type of OS that manages a group of different computers and makes them appear to be a single computer. It is designed to operate on a network of computers.

4. Network OS : a type of OS that runs on a server and provides the capability to manage data, users, groups, security and other networking functions.

5. Multiprocessing OS : these systems are used to boost the performance of multiple CPUs within a single computer system.

6. Multiprogramming OS : allows multiple programs to run simultaneously on a single processor. While one program waits for I/O operations, the CPU can switch to another ready-to-run program.

Q3. Explain different architectures of OS

→ 1. Monolithic Architecture : each component of the OS is contained in the kernel i.e it is working in kernal space, communicate with each other using function calls. Since, all components are independent, when one of them fails the entire system fails.

2. Layered Architecture : components with similar functionalities are grouped to form a layer and in this way, total n+1 layers are constructed and counted from 0 to n where each layer has a different set of functionalities and services. Eg. OS / 360 and OS / 390 from IBM

3. MicroKernel Architecture : components like process management, networking, file system interaction, and device management are executed outside the kernal while memory management and synchronisation are executed inside the kernel.

4. Hybrid - kernel : combination of monolithic and microkernel, and gives a more advance and helpful approach. Implements speed and design of monolithic, modularity and stability of microkernel.

5. Exo - Kernal : developed at MIT to provide application - level management of hardware resources. By seperating resource management from protection, the exokernal architecture aims to enable application - specific customization.

Q4 Write the solutions for following problems:

i. Producer Consumer: The problem challenge lies in ensuring that producers do not overwrite data that has not yet been consumed and that consumers do not attempt to consume data that is not available.

To solve, we use semaphores that can be used to control access to shared resources.

1. Semaphore 'empty': counts the number of empty slots in the buffer.
2. Semaphore 'full': counts the number of filled slots in the buffer.
3. 'Mutex': ensures mutual exclusion when accessing the buffer.

```
Producer () {
   while true {
       item = produce Item ()

       wait (empty)
       wait (mutex)

       buffer.append (item)

       signal (mutex)
}      signal (full)
```

page No

Date: / /

```
Consumer () {
    while (true) {
        wait (full)
        wait (mutex).

        item = buffer.remove

        signal (mutex)
        Signal (empty)

        Cosnusume Item (item)
}
```

ii. Reader - Writer : The challenge is to allow concurrent access for multiple readers while ensuring that writers have enclusive access to prevent data inconsistency. To Solve we can use semaphores:

1. Semaphore 'mutex': ensures mutual enclusion when updating the count of readers
2. Semaphore 'wrt': controls access for writers, ensuring that only one writer can write at a time.
3. Integer 'read Count': keeps track of the number of active readers.

```
Reader () {
  while (true) {
      wait (mutex);
      readCount = readCount + 1;
      if (readCount == 1)
          wait (wrt);
      Signal (mutex);
      read.data()
      wait (mutex);      // enter critical
      readCount --;
      if (readCount == 0)
          signal (wrt);
      Signal (mutex);
  }.
}

Writer () {
    while (true) {
        wait (wrt);

        writeData ();

        Signal (wrt);
    }
}
```

page No
Date: / /

iii. Dining Philosopher: The challenge is to ensure that no philosopher starves i.e everyone gets a chance to eat and that deadlock is avoided.

To solve this, we can use semaphores and mutexes to manage access to the forks.

1. Forks as Semaphores: Each fork is represented as a semaphore initialised to 1, indicating that the fork is available
2. Philosopher States: Each philosopher can be in one of three states: thinking, hungry or eating.
3. Mutex for Access Control: used to ensure mutual exclusion when they pick up or put down forks.

```
Philosopher (id) {
    while (true) {
        think ();
        wait (mutex);
        wait (fork [id]);
        wait ( fork [(id+1) % 5]);
        Signal (mutex);
        eat ();
        Signal (fork [id]);
        Signal (fork [(id+1) %]);
} }
```

Q5

i. First Come First Serve

Process ID	Burst Time
P1	24
P2	3
P3	3

Gantt Chart :

P1	P2	P3
0	24	27	30

Waiting Time : (ms)

$P1 = 0$, $P2 = 24$, $P3 = 27$

Avg. Waiting Time : (ms)

$= (0 + 24 + 27)/3 = 17$

Turnaround Time : (ms)

$P1 = 0 + 24 = 24$

$P2 = 24 + 3 = 27$

$P3 = 27 + 3 = 30$

Scanned with ACE Scanner

ii. Shortest Job First (SJF) :

Process ID	Arrival Time	Burst Time
A	0	3
B	2	6
C	4	4
D	6	5
E	8	2

Gantt Chart :

A	B	E	C	D

0 3 9 11 15 20

Waiting time :-

$A = 0$

$B = 3 - 2 = 1$

$C = 11 - 4 = 7$

$D = 15 - 6 = 9$

$E = 9 - 8 - 1$

Avg. waiting time $= \frac{0+1+7+9+1}{5} = \cancel{5.4}\ 3.6$

Page No.
Date: / /

iii Round Robin Scheduling Q = 100 ms

Process	A.T	Execution Time
P_0	0	250
P_1	50	170
P_2	130	75
P_3	190	100
P_4	210	130
P_5	350	50

Gantt Chart:

P_0	P_1	P_2	P_3	P_4	P_5	P_0	P_1	P_4	P_1

0 100 200 275 375 475 525 625 655 725 775

Waiting Time:

$P_0 = (0-0) + (525-100) = 425$

$P_1 = 100 + (625-200) + (725-625) = 625$

$P_2 = 200$

$P_3 = 275$

$P_4 = 375 + (655-475) = 655$

$P_5 = 475$

Avg. W.T = 442.5

Turnaround Time:

$P_0 = 625$, $P_1 = 775$, $P_2 = 275$, $P_3 = 375$,

$P_4 = 725$, $P_5 = 525$

Avg. T.T = 550

Q6. Need Banker's Algorithm

a. Need = Max - Allocation

$$= \begin{bmatrix} 7 & 4 & 3 \\ 1 & 2 & 2 \\ 6 & 0 & 0 \\ 0 & 1 & 1 \\ 4 & 3 & 1 \end{bmatrix}$$

~~P1 Need ≤ available = [1 2 2] < [3 3 2]~~

For P_0: Need > Available ⇒ wait
[7 4 3] [3 3 2]

For P_1: Need <= Available ⇒ Safe
[1 2 2] <= [3 3 2]

Available = Available + Allocate
= [3 3 2] + [2 0 0] = [5 3 2]

For P_2: Need ≥ Available ⇒ wait
[6 0 0] ≥ [5 3 2]

For P_3: Need < Available ⇒ safe
[0 1 1] [5 3 2]

Available = [5 3 2] + [2 1 1] = [7 4 3]

For P_4: Need < Available ⇒ safe
[4 3 1] < [7 4 3]

Available = [7 4 3] + [0 0 2] = [7 4 5]

For P_0 : Need < work ⇒ Safe

Avail. = [7 4 5] + [0 1 0] = [7 5 5]

For P_2: Need < work ⇒ safe

Avail. = [7 5 5] + [3 0 2] = [10 5 7]

∴ Safe Sequence : P_1, P_3, P_4, P_0, P_2

b) Banker's Algorithm :

work = Available = [000]

For P_0: Need = [000] <= Work => safe

work = work + Allocation

= [000] + [010] = [0 10]

For P_2: [202] > [010] => wait

For P_2: Need < work => Safe seq.

work = [0 10] + [303] = [313]

For P_3: Need < work => Safe Seq.

work = [313] + [211] = [524]

For P_4: Need < work => Safe Seq.

work = [524] + [002] = [526]

For P_1: Need < work => Safe seq.

work = [526] + [200]

= [726]

∴ Safe Sequence : P_0, P_2, P_3, P_4, P_1

∴ No deadlock

page No.
Date: / /

Q7. $P_1 = 212$ BLOCKS: 100, 500,
a. $P_2 = 417$ (Kb) 200, 300,
$P_3 = 112$ 600
$P_4 = 426$
(Kb)

→ First Fit :

	100
P_1 (212)	500
P_3 (112)	200
	300
P_2 (417)	600

P_4 (426) cannot get memory

Best Fit :

	100
P_2 (417)	500
P_3 (112)	200
P_1 (212)	300
P_4 (426)	600

→ Worst Fit :

	100
P_2 (417)	500
	200
P_3 (112)	300
P_1 (212)	600

P_4 (426) cannot get memory

Next Fit :

	100
P_1 (212)	500
P_3 (112)	200
	300
P_2 (417)	600

P_4 (426) cannot get memory

II String : 7, 1, 0, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0, 1

a First In First Out (FIFO)

Page	Frame	Fault
7	[7, -, -]	Yes
1	[7, 1, -]	Yes
0	[7, 1, 0]	Yes
2	[2, 1, 0]	Yes
0	—	—
3	[2, 3, 0]	Yes
0	~~[2, 3, 0~~	—
4	[4, 3, 0]	Yes
2	[4, 2, 0]	Yes
3	[4, 2, 3]	Yes
0	[0, 2, 3]	Yes
3	—	—
2	—	—
1	[0, 1, 3]	Yes
2	[0, 1, 2]	Yes
0	—	—
1	—	—
7	[7, 1, 2]	Yes
0	[7, 0, 2]	Yes
1	[7, 0, 1]	Yes

14 page faults

page No.

Date: / /

5. Optimal

String 7, 1, 0, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0, 1

Page	Frames	Fault
7	[7, -, -]	Yes
1	[7, 1, -]	Yes
0	[7, 1, 0]	Yes
2	[2, 1, 0]	Yes
0	—	—
3	[2, 3, 0]	Yes
0	—	—
4	(2, 3, 4)	Yes
2	—	—
3	—	—
0	[2, 3, 0]	Yes
3	—	—
2	—	—
1	[2, 1, 0]	Yes
2	—	—
0	—	—
1	—	—
7	[7, 1, 0]	Yes
0	—	—
1	—	—

9 page faults

Page No.
Date: / /

II. Frames : 3

String : 7,1,0,2,0,3,0,4,2,3,0,3,2,1,2,0,1,7,0,1

~~Least Recently Used LRU~~

So: ~~First In First Out~~ ~~(Fifo)~~

Page	Frames	Fault
7	[7,-,-]	Yes
1	[7,1,-]	Yes
0	[7,1,0]	Yes
2	[2,1,0]	Yes
0	[2,1,0]	—
3	[2,3,0]	Yes
0	[2,3,0]	—
4	[4,3,0]	Yes
2	[4,2,0]	Yes
3	[4,2,3]	Yes
0	[0,2,3]	Yes
3	[0,2,3]	—
2	[0,2,3]	—
1	[1,2,3]	Yes
2	[1,2,3]	—
0	[1,2,0]	Yes
1	[1,2,0]	—
7	[1,7,0]	Yes
0	[1,7,0]	—
1	[1,7,0]	—

12 faults

d. Second Chance (Clock)

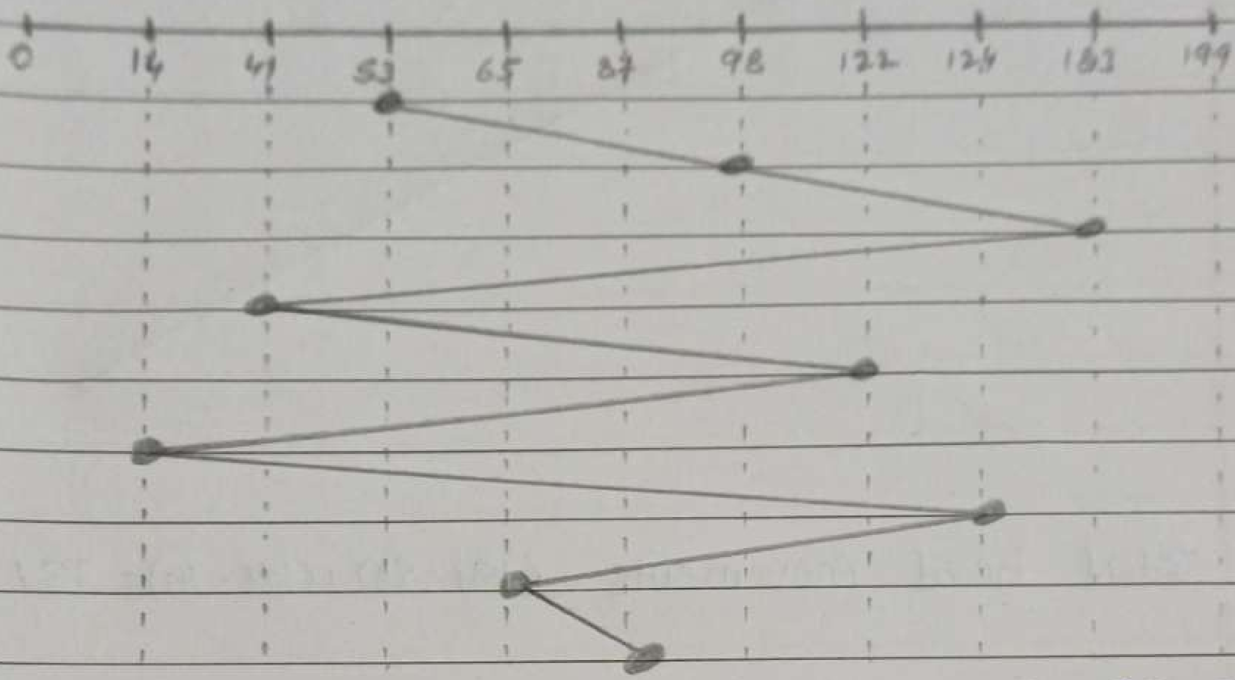
String: 7,1,0,2,0,3,0,4,2,3,0,3,2,1,2,0,1,7,0,1

Page	Frames	Fault
7	[7^0, -, -]	Yes
1	[7^0, 1^0, -]	Yes
0	[7^0, 1^0, 0]	Yes
2	[2^0, 1^0, 0^0]	Yes
0	[2^0, 1^0, 0^1]	—
3	[2^0, 3^0, 0^1]	Yes
0	[2^0, 3^0, 0^1]	—
4	[4^0, 3^0, 0^0]	Yes
2	[4^0, 3^0, 2^0]	Yes
3	[4^0, 3^1, 2^0]	—
0	[0^0, 3^0, 2^0]	Yes
3	[0^0, 3^1, 2^0]	—
2	[0^0, 3^1, 2^1]	—
1	[1^0, 3^0, 2^1]	Yes
2	[1^0, 3^0, 2^1]	—
0	[1^0, 0^0, 2^1]	Yes
1	[1^1, 0^0, 2^1]	~~Yes~~ —
7	[1^1, 7^0, 2^0]	Yes
0	[1^1, 7^0, 0^0]	Yes
1	[1^1, 7^0, 0^0]	—

12 page faults

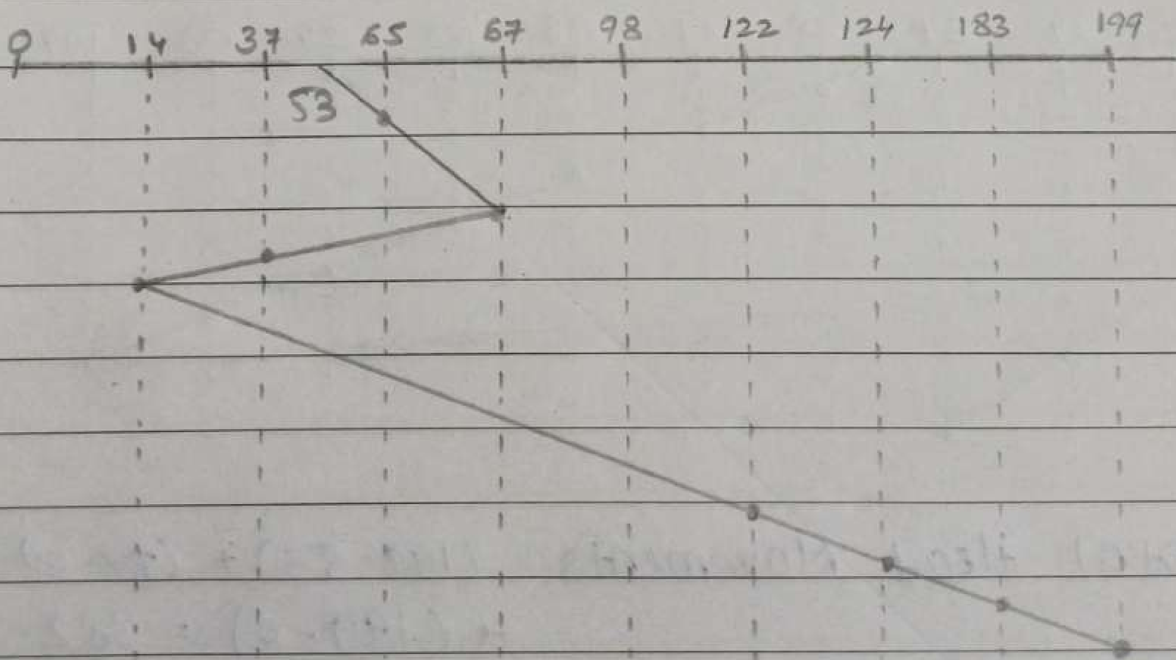
Q.8 Current position : 53

a. FCFS :



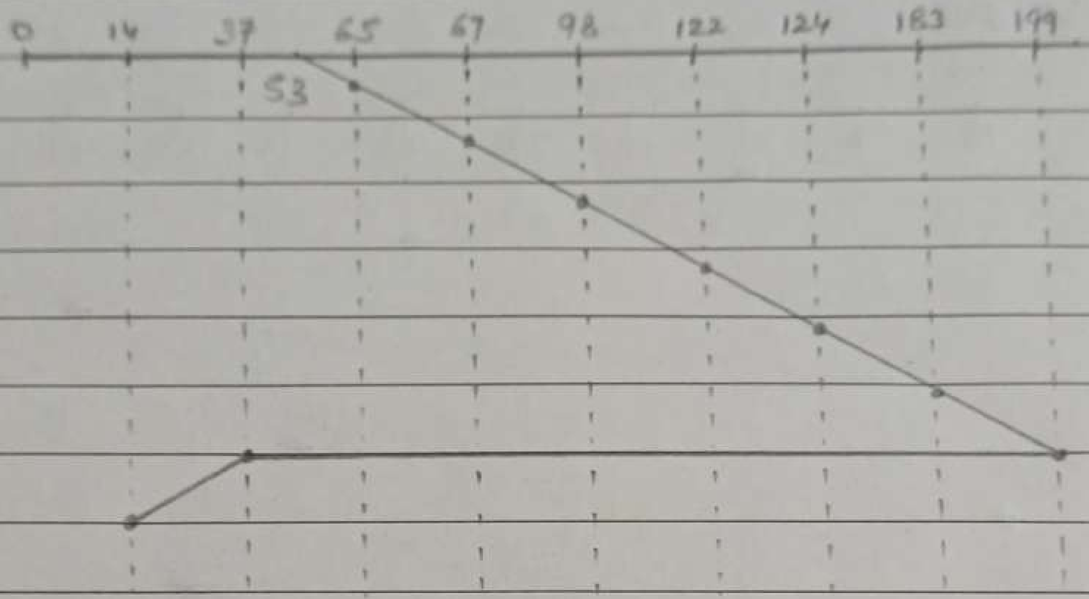
Total Head movements : 45+85+146+85+108+110+59+2 = 64

b. SSTF :



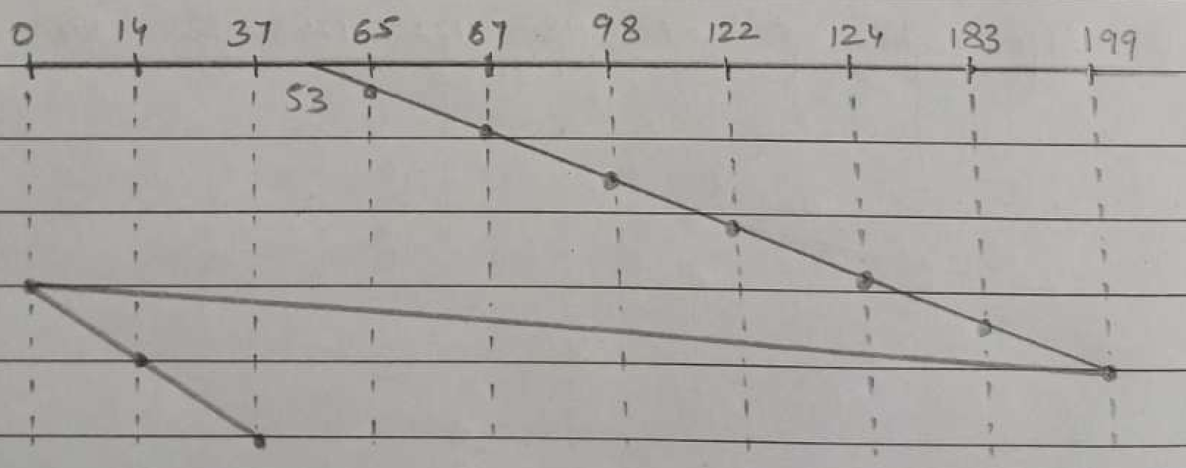
Total Head movements : (67 − 53) + (67 − 14)
+ (183 − 14) = 236

C. SCAN:



Total head movements : $(199-53)+(199-14) = 331$

D. C-SCAN:



Total Head Movements : $(199-53) + (37-0)$
$+ (199-0) = 382$

page No
Date: / /

Q9. I/O organisation :

→ Refers to how a computer system manages communication between its internal components and external devices. Its crucial for ensuring smooth data transfer between hardware components and peripheral devices.

Key components :

1) I/O devices :- external devices like keyboards, monitors, printers etc. that interact with the system, within the organisation.

2) I/O interfaces :- Acts as a bridge between CPU and peripheral devices, converting data into compatible formats eg. PCI, USB

3) I/O techniques :- Programmed I/O :- CPU controls all I/O operations by polling the device status, making it simple but inefficient due to CPU involvement in each data transfer. In Interrupt-driven, the device interrupts the CPU when its ready for data transfer, reducing CPU idle time. Direct memory access allows I/O devices to send or receive data directly to or from the main memory, bypassing the CPU for faster data transfer.

4) Memory mapped I/O uses same address space for memory I/O wheras Isolated I/O maintains seperate spaces, requiring special instructions.

Page No.
Date: / /

Q10. Record Blocking :

→ A method used in data storage and file systems to manage the way data records are organized and stored on physical storage media such as hard drives, tapes etc. It involves grouping individual records into larger blocks before writing them to storage, which can improve I/O performance and storage efficienty.

→ Types:
1) Fixed-length : all records in block are same size
2) variable-length : different size records in a block
3) spanned : Records can span across blocks.
4) Unspanned : Each records fits within one block.

→ Benefits :
- Improved efficiency : fewer I/O operations
- Reduced overhead : less frequent disk access
- Better Media utilisation : for sequential storage like tapes.

→ Considerations :
- Block size : needs to balance efficiency and performance
- Buffering : Blocks are often stored in memory before writing
- Data integrity : spanned blocks can risk data loss if one part is corrupted.