CDAC MUMBAI

Concepts of Operating System

Assignment 2

Part A

1. **echo "hello world**" :- this command will print hello world string on the terminal it is a shell command to print a string if we want to add a variable in between we can use $ to do it.
2. **name = "Product"** :- this saves a variable in shell name and stores Product string in it
3. **touch file.txt** :- this command creates a file with .txt format in the current directory (in current system we need permission to execute this command if we do it outside our home directory)
4. **ls -a**:- this command shows all the files and directories and hidden files present in the current directory
5. **rm file.txt** :- removes the file.txt file from the current directory
6. **cp file1.txt file2.txt:-** this command is used to copy all the contents of file1.txt to file2.txt and if you do not have file2.txt it will create and copy into it
7. **mv file.txt /path/to/directory** :- moves the file from current directory to the directory of which path is mentioned
8. **chmod 755 script.sh**:- in chmod command to specify permission we can use binary representation of numbers so chmod num1 num2 num3 filename here num1 :- is for owner num2:- for group num3 :- for others and each number's binary representation is used to give read write execute permission for ex 000 would give none while 001=1 will give execute permission so chmod 755 wile give wrx persimmon to owner and read and execute permission to group and others.
9. **grep "pattern" file.txt**:- the command is used to find the pattern inside the file.txt and display the line.
10. **kill PID** :- as we know each process has a PID to kill PID will kill the process who's PID is mentioned
11. **mkdir mydir && cd mydir && touch file.txt && echo "Helo world" > file.txt && cat file.txt**:- in this command && is used to execute one command after another it first creates mydir directory then navigates to mydir then creates a file echo hello world then directs the output to file.txt using '>' then using cat command prints the content of file,txt
12. **ls -l | grep ".txt"** :- we are using piping concept | allows us to take stdout of one command and transfer it to stdout of another command so ls -l list all the files and directories in current directory which is then used by grep to find files with .txt extension in them
13. **cat file1.txt file2.txt | sort | uniq** :- this commands concatenates file1 and file2 then sort them in ascending order and then prints each line once from both the files
14. **ls -l | grep "^d"** : this command first gives all the list of files and directories as a list to grep and then prints the directories as in ls -l ^d pattern is given to each directory as a representation that it is a directory
15. **grep -r "pattern" /path/to/directory** :- this command recursively tries to search all the files in the given path directory that matches the pattern given and display their names
16. **cat file1.txt file2.txt | sort | uniq -d** : this command will concatenate file1 and file2 then sort them and then print duplicate lines present in these both files as -d option is used to print duplicate lines
17. **chmod 664 file.txt** :- this command will give read and write permission to owner and group and will give write permission to others
18. **cp -r source d to destination d**:- will copy source directory to destination directory recursively
19. **find path/to/search -name "*.txt"** :- this command will search the given directory in path and print out names of file matching .txt
20. **chmod u+x file.txt** :- this command will give execution permission to the owner of the file

21. **echo $PATH** :- this command prints all the content of path variable where it searches for different commands and executables in the system these are directories separated by :

Identify True or False:

1. **ls** is used to list files and directories in a directory.:- True
2. **mv** is used to move files and directories.:- True
3. **cd** is used to copy files and directories.:-False
4. **pwd** stands for "print working directory" and displays the current directory.:-True
5. **grep** is used to search for patterns in files.:True

6. **chmod 755 file.txt** gives read, write, and execute permissions to the owner, and read and execute permissions to group and others.-True
7. **mkdir -p directory1/directory2** creates nested directories, creating directory2 inside director1 if directory1 does not exist.:-True
8. **rm -rf file.txt** deletes a file forcefully without confirmation.:True

Identify the Incorrect Commands:

1. **chmodx** is used to change file permissions.:- chmod is the correct command for the task mentioned
2. **cpy** is used to copy files and directories.:- cp is the proper command for the task
3. **mkfile** is used to create a new file.:- touch filename is the correct command to create a new file
4. **catx** is used to concatenate files:- cat command is used to concatenate files like cat file1 file2 .
5. **rn** is used to rename files.:- rn is not a command we can use mv file1 file2 command to rename or can download rename by sudo apt install rename

---

Part C

Question 1: Write a shell script that prints "Hello, World!" to the terminal.
Sol :-

```
  GNU nano 6.2
#!/bin/bash
echo "Hello,World!";
```

Question 2: Declare a variable named "name" and assign the value "CDAC Mumbai" to it. Print the value of the variable.
Sol:-

```
#!/bin/bash
echo "enter the string";
read x;
echo $x;
```

```
ayush@DRAGO:~/LinuxAssignment/shellPrograming$ nano var.sh
ayush@DRAGO:~/LinuxAssignment/shellPrograming$ bash var.sh
enter the string
CDAC Mumbai
CDAC Mumbai
```

Question 3: Write a shell script that takes a number as input from the user and prints it.

Sol:- the above script will work the same way

Question 4: Write a shell script that performs addition of two numbers (e.g., 5 and 3) and prints the result.

sol:-

```
  GNU nano 6.2
#!/bin/bash
x=$((3+4));
echo $x;
```

```
ayush@DRAGO:~/LinuxAssignment/shellPrograming$ bash ad.sh
7
ayush@DRAGO:~/LinuxAssignment/shellPrograming$
```

Question 5: Write a shell script that takes a number as input and prints "Even" if it is even, otherwise prints "Odd".

Sol:-

```
  GNU nano 6.2
#!/bin/bash
echo "enter the number";
read x;
if [ $(( $x%2 )) == 0 ]
then
        echo $x " is even";
else
         echo $x " is odd";
fi
```

```
ayush@DRAGO:~/LinuxAssignment/shellPrograming$ nano evenodd.sh
ayush@DRAGO:~/LinuxAssignment/shellPrograming$ bash evenodd.sh
enter the number
4
4  is even
ayush@DRAGO:~/LinuxAssignment/shellPrograming$ bash evenodd.sh
enter the number
5
5  is odd
ayush@DRAGO:~/LinuxAssignment/shellPrograming$
```

Question 6: Write a shell script that uses a for loop to print numbers from 1 to 5.
sol:-

```
  GNU nano 6.2
#!/bin/bash
for i in 1 2 3 4 5
do
        echo $i;
done;
```

```
ayush@DRAGO:~/LinuxAssignment/shellPrograming$ nano loops.sh
ayush@DRAGO:~/LinuxAssignment/shellPrograming$ bash loops.sh
1
2
3
4
5
ayush@DRAGO:~/LinuxAssignment/shellPrograming$
```

Question 7: Write a shell script that uses a while loop to print numbers from 1 to 5.

sol:-

```
  GNU nano 6.2
#!/bin/bash
i=1;
while [[ i -le 5 ]] ;do
        echo $i;
        i=$(( $i+1 ));
done
```

```
ayush@DRAGO:~/LinuxAssignment/shellPrograming$ nano while.sh
ayush@DRAGO:~/LinuxAssignment/shellPrograming$ bash while.sh
1
2
3
4
5
ayush@DRAGO:~/LinuxAssignment/shellPrograming$
```

Question 8: Write a shell script that checks if a file named "file.txt" exists in the current directory. If it does, print "File exists", otherwise, print "File does not exist".
sol:-

```
  GNU nano 6.2
#!/bin/bash
read x
if [ -f $x ]
then
        echo "File is present";
else
        echo "File is not present";
fi
```

```
ayush@DRAGO:~/LinuxAssignment/shellPrograming$ nano com.sh
ayush@DRAGO:~/LinuxAssignment/shellPrograming$ bash com.sh
ad.sh
File is present
ayush@DRAGO:~/LinuxAssignment/shellPrograming$
```

Question 9: Write a shell script that uses the if statement to check if a number is greater than 10 and prints a message accordingly.

sol :-

```
  GNU nano 6.2
#!/bin/bash
read x
if [ $x -gt 10 ]
then
        echo $x " is greater than 10";
else
        echo $x " is less than 10";
fi
```

```
ayush@DRAGO:~/LinuxAssignment/shellPrograming$ nano greater.sh
ayush@DRAGO:~/LinuxAssignment/shellPrograming$ bash greater.sh
5
5  is less than 10
```

Question 10: Write a shell script that uses nested for loops to print a multiplication table for numbers from 1 to 5. The output should be formatted nicely, with each row representing a number and each column representing the multiplication result for that number.

sol:-

```
  GNU nano 6.2
#!/bin/bash
for i in 1 2 3 4 5
do
        for j in 1 2 3 4 5 6 7 8 9 10 11 12
        do
                result=$(( $i * $j ));
                printf " $result  "
        done
        echo ""
done
```

```
ayush@DRAGO:~/LinuxAssignment/shellPrograming$ nano multi.sh
ayush@DRAGO:~/LinuxAssignment/shellPrograming$ bash multi.sh
 1   2   3   4   5   6   7   8   9   10   11   12
 2   4   6   8   10   12   14   16   18   20   22   24
 3   6   9   12   15   18   21   24   27   30   33   36
 4   8   12   16   20   24   28   32   36   40   44   48
 5   10   15   20   25   30   35   40   45   50   55   60
```

Question 11: Write a shell script that uses a while loop to read numbers from the user until the user enters a negative number. For each positive number entered, print its square. Use the **break** statement to exit the loop when a negative number is entered.

sol :-

```
  GNU nano 6.2
#!/bin/bash
x=1;
while [[ x -gt 0 ]]; do
        read x;
        if [ $x -lt 0 ]
        then
                break;
        fi
        echo $(( $x * $x ));
done;
```

```
ayush@DRAGO:~/LinuxAssignment/shellPrograming$ nano cond.sh
ayush@DRAGO:~/LinuxAssignment/shellPrograming$ bash cond.sh
2
4
3
9
9
81
-2
```

Part D

Common Interview Questions (Must know)

1. What is an operating system, and what are its primary functions?
2. Explain the difference between process and thread.
3. What is virtual memory, and how does it work?
4. Describe the difference between multiprogramming, multitasking, and multiprocessing.
5. What is a file system, and what are its components?
6. What is a deadlock, and how can it be prevented?
7. Explain the difference between a kernel and a shell.
8. What is CPU scheduling, and why is it important?
9. How does a system call work?

10. What is the purpose of device drivers in an operating system?
11. Explain the role of the page table in virtual memory management.
12. What is thrashing, and how can it be avoided?
13. Describe the concept of a semaphore and its use in synchronization.
14. How does an operating system handle process synchronization?
15. What is the purpose of an interrupt in operating systems?
16. Explain the concept of a file descriptor.
17. How does a system recover from a system crash?
18. Describe the difference between a monolithic kernel and a microkernel.
19. What is the difference between internal and external fragmentation?
20. How does an operating system manage I/O operations?
21. Explain the difference between preemptive and non-preemptive scheduling.
22. What is round-robin scheduling, and how does it work?
23. Describe the priority scheduling algorithm. How is priority assigned to processes?
24. What is the shortest job next (SJN) scheduling algorithm, and when is it used?
25. Explain the concept of multilevel queue scheduling.
26. What is a process control block (PCB), and what information does it contain?
27. Describe the process state diagram and the transitions between different process states.
28. How does a process communicate with another process in an operating system?
29. What is process synchronization, and why is it important?
30. Explain the concept of a zombie process and how it is created.
31. Describe the difference between internal fragmentation and external fragmentation.
32. What is demand paging, and how does it improve memory management efficiency?
33. Explain the role of the page table in virtual memory management.
34. How does a memory management unit (MMU) work?
35. What is thrashing, and how can it be avoided in virtual memory systems?
36. What is a system call, and how does it facilitate communication between user programs and the operating system?
37. Describe the difference between a monolithic kernel and a microkernel.
38. How does an operating system handle I/O operations?
39. Explain the concept of a race condition and how it can be prevented.

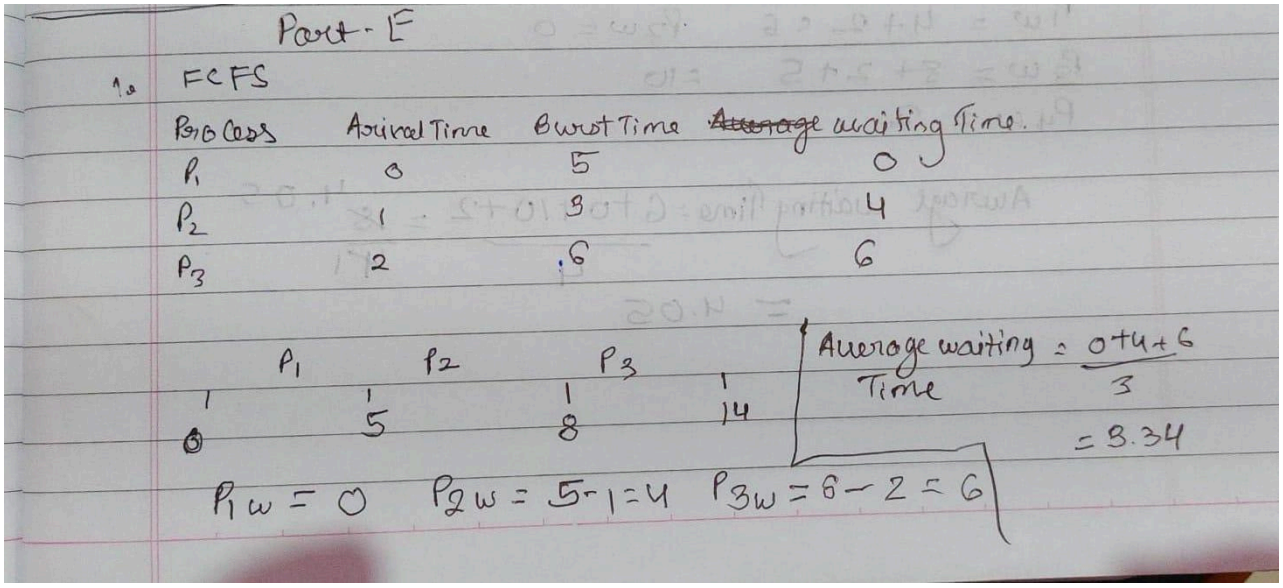
40. Describe the role of device drivers in an operating system.
41. What is a zombie process, and how does it occur? How can a zombie process be prevented?
42. Explain the concept of an orphan process. How does an operating system handle orphan processes?
43. What is the relationship between a parent process and a child process in the context of process management?
44. How does the fork() system call work in creating a new process in Unix-like operating systems?
45. Describe how a parent process can wait for a child process to finish execution.
46. What is the significance of the exit status of a child process in the wait() system call?
47. How can a parent process terminate a child process in Unix-like operating systems?
48. Explain the difference between a process group and a session in Unix-like operating systems.
49. Describe how the exec() family of functions is used to replace the current process image with a new one.
50. What is the purpose of the waitpid() system call in process management? How does it differ from wait()?
51. How does process termination occur in Unix-like operating systems?
52. What is the role of the long-term scheduler in the process scheduling hierarchy? How does it influence the degree of multiprogramming in an operating system?
53. How does the short-term scheduler differ from the long-term and medium-term schedulers in terms of frequency of execution and the scope of its decisions?
54. Describe a scenario where the medium-term scheduler would be invoked and explain how it helps manage system resources more efficiently.

Part E

1. Consider the following processes with arrival times and burst times:

Process	Arrival Time	Burst Time
P1	0	5
P2	1	3
P3	2	6

Calculate the average waiting time using First-Come, First-Served (FCFS) scheduling
sol :-



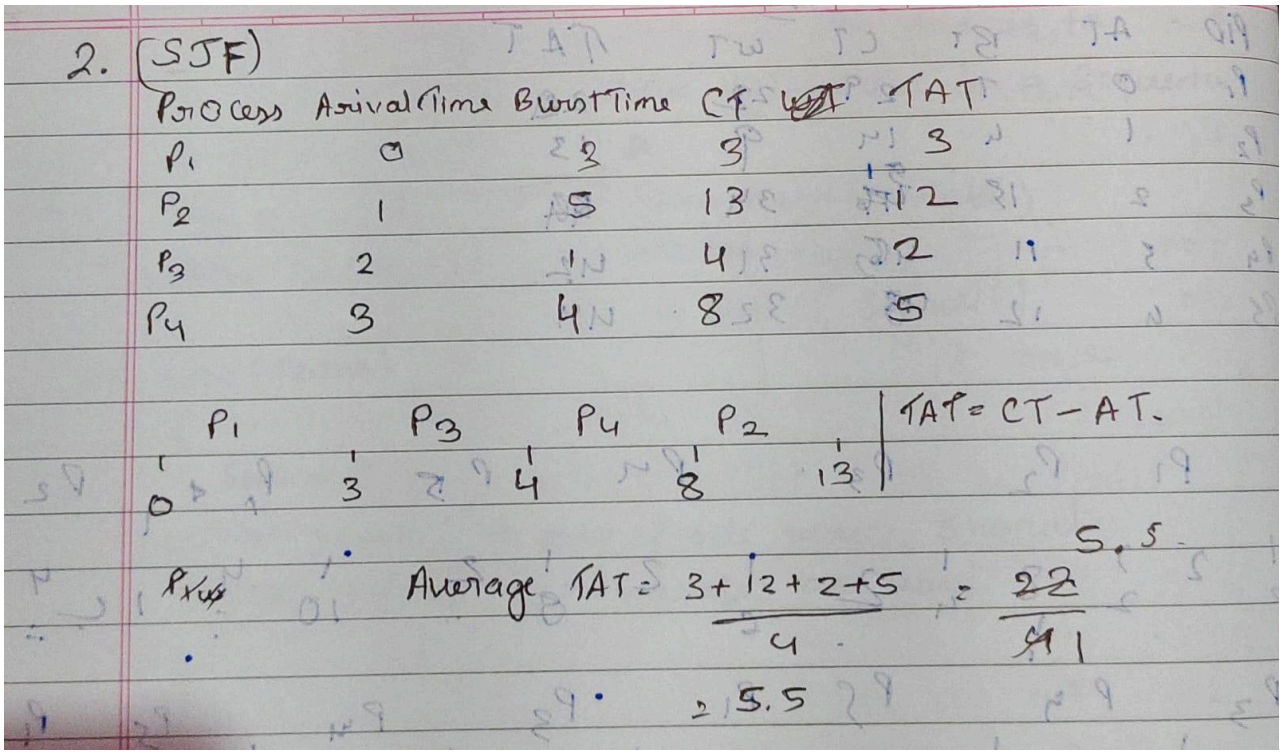
.

2. Consider the following processes with arrival times and burst times:

Process	Arrival Time	Burst Time
P1	0	3
P2	1	5
P3	2	1
P4	3	4

Calculate the average turnaround time using Shortest Job First (SJF) scheduling.

sol :-

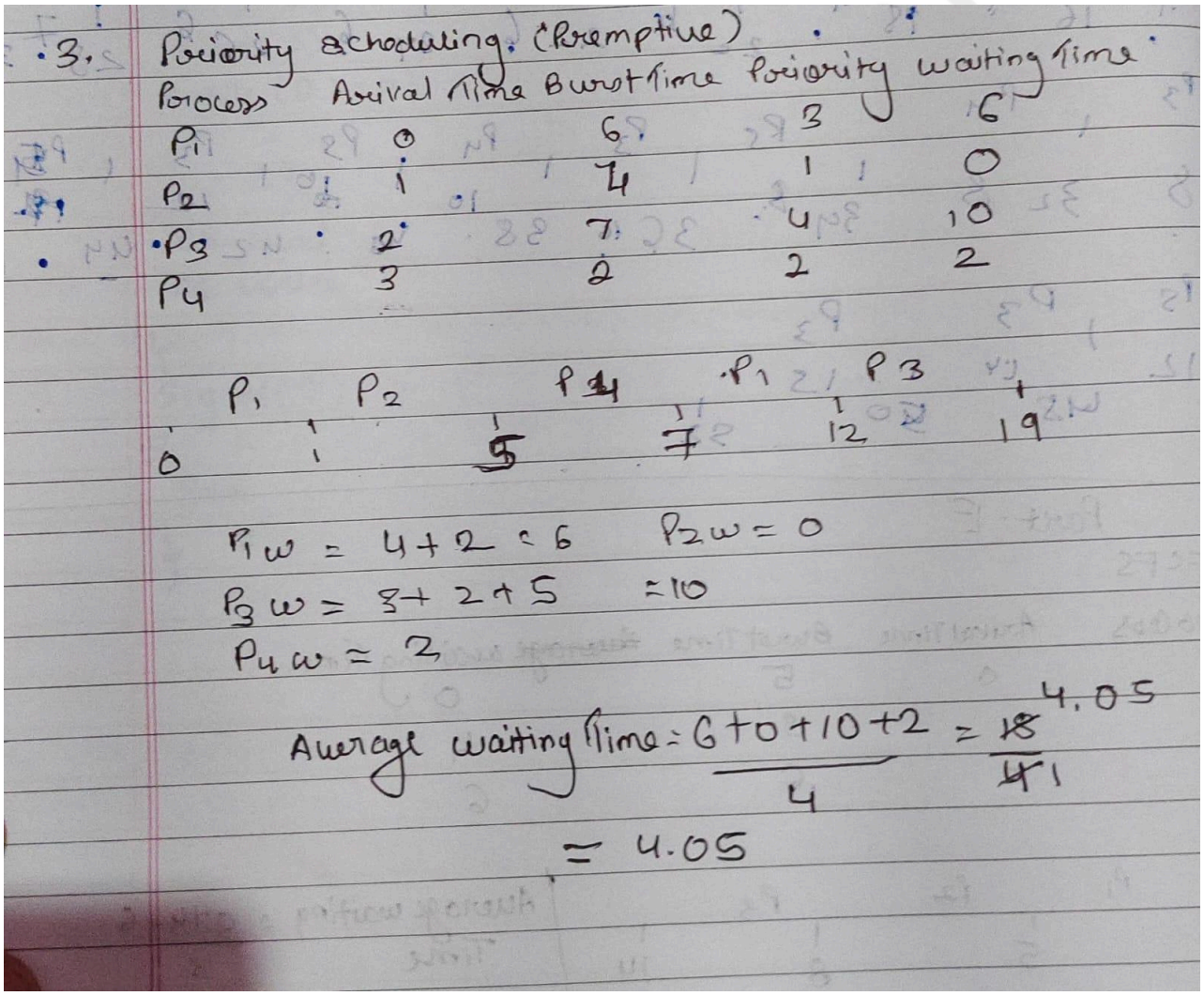


3. Consider the following processes with arrival times, burst times, and priorities (lower number indicates higher priority):

Process	Arrival Time	Burst Time	Priority
P1	0	6	3
P2	1	4	1
P3	2	7	4
P4	3	2	2

Calculate the average waiting time using Priority Scheduling.

sol:-

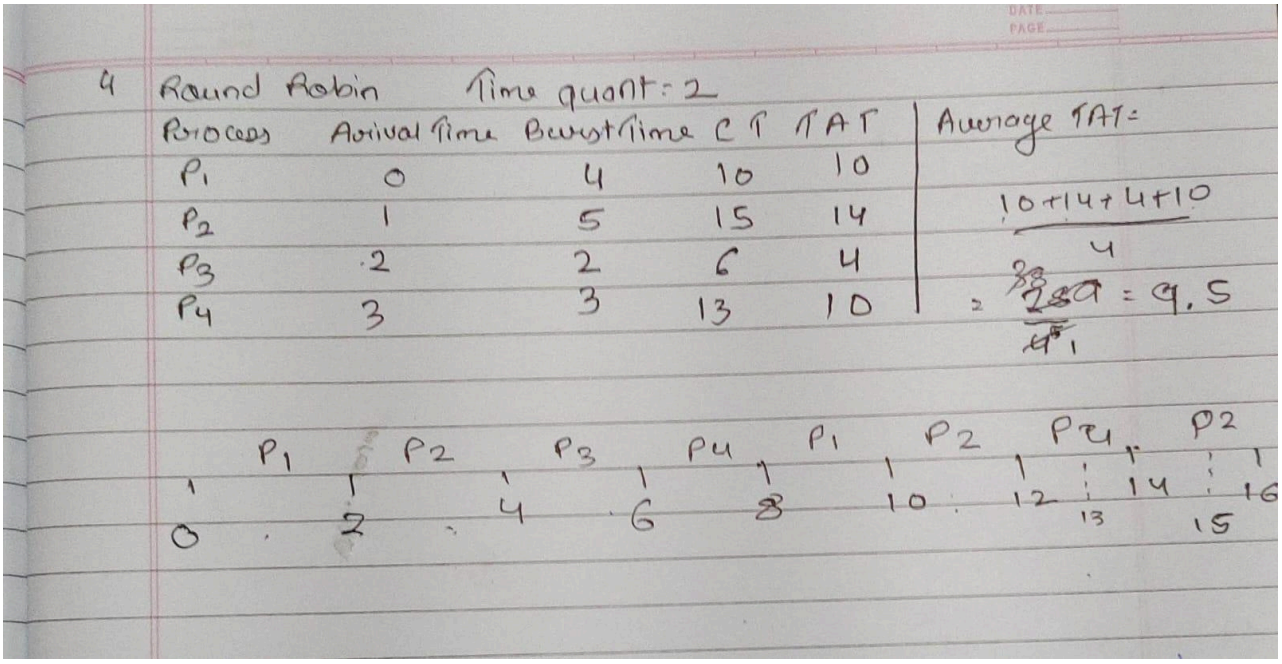


4. Consider the following processes with arrival times and burst times, and the time quantum for Round Robin scheduling is 2 units:

Process	Arrival Time	Burst Time
P1	0	4
P2	1	5
P3	2	2
P4	3	3

Calculate the average turnaround time using Round Robin scheduling.

sol:-



4 Round Robin Time quant: 2

Process	Arrival Time	Burst Time	CT	TAT
P_1	0	4	10	10
P_2	1	5	15	14
P_3	2	2	6	4
P_4	3	3	13	10

Average TAT = $\frac{10+14+4+10}{4} = \frac{38}{4} = 9.5$

P_1 P_2 P_3 P_4 P_1 P_2 P_4 P_2

0 2 4 6 8 10 12 13 14 15 16

5. Consider a program that uses the **fork()** system call to create a child process. Initially, the parent process has a variable **x** with a value of 5. After forking, both the parent and child processes increment the value of **x** by 1.
   sol :-
   What will be the final values of **x** in the parent and child processes after the **fork()** call?
   Sol:- the child and parent process are created using clone(); the both have different address the child has a copy of all the global and local variable of the parent but they don't share or communicate so no matter if one execute before another the initial value for both will be x=5 and final value be x=6 also note fork()
   for reference :-
   Child Process Initial Value :: localVar = 0, globalVar = 0
   Address of malloced privateMem in child = 0x7f245277d000 and value is 0
   Address of malloced sharedMem in child = 0x7f2452750000 and value is 0
   Updated child Process :: localVar = 1, globalVar = 2
   lets change the value of privateMem variable created by malloc in child
   Address of malloced privateMem in child = 0x7f245277d000 and value is 50
   lets change the value of sharedMem variable created my malloc in child
   Address of malloced sharedMem in child = 0x7f2452750000 and value is 100

   Parent Process Initial Value :: localVar = 0, globalVar = 0
   Address of malloced privateMem in parent = 0x7f245277d000 and value is 0
   Address of malloced sharedMem in parent = 0x7f2452750000 and value is 100
   Updated parent process :: localVar = 10, globalVar = 20
   lets change the value of privateMem variable created by malloc in parent
   Address of malloced privateMem in parent= 0x7f245277d000 and value is 100
   lets change the value of sharedMem variable created my malloc in parent
   Address of malloced sharedMem in parent = 0x7f2452750000 and value is 400

Submission Guidelines:

- Document each step of your solution and any challenges faced.
- Upload it on your GitHub repository

Additional Tips:

- Experiment with different options and parameters of each command to explore their functionalities.
- This assignment is tailored to align with interview expectations, CCEE standards, and industry demands.
- If you complete this then your preparation will be skyrocketed.