Assignment 2 – SHREYASH SANJAY AGHARKAR_KH

Part -A

1 echo "Hello, World!"

- Prints the string "Hello, World!" to the terminal.

2 name="Productive"

- Assigns the string "Productive" to the variable name.

3 touch file.txt

- Creates an empty file named file.txt in the current directory (if it doesn't already exist). If it already exists, it updates the file's timestamp.

4 ls -a

- Lists all files and directories, including hidden ones (those starting with a dot .), in the current directory.

5 rm file.txt

Removes (deletes) the file named file.txt from the current directory.

6 cp file1.txt file2.txt

- Copies the content of file1.txt into file2.txt. If file2.txt already exists, it will be overwritten.

7 mv file.txt /path/to/directory/

- Moves file.txt to the specified directory (/path/to/directory/). If the file is already there, it will overwrite it.

8 chmod 755 script.sh

- Changes the permissions of script.sh so that the owner has read, write, and execute permissions (7), while the group and others have read and execute permissions (5).

9 grep "pattern" file.txt

- Searches for the string "pattern" in file.txt and prints the matching lines.

10 kill PID

- Terminates the process with the specified Process ID (PID). It sends a signal to stop the process.

11.mkdir mydir && cd mydir && touch file.txt && echo "Hello, World!" > file.txt && cat file.txt

- Creates a directory mydir, enters it (cd mydir), creates a file file.txt, writes "Hello, World!" into it, and then displays the contents of file.txt.

12 ls -l | grep ".txt"

- Lists the details of all files in the current directory and filters the results to show only those that have .txt in their name.

13 cat file1.txt file2.txt | sort | uniq

- Concatenates the contents of file1.txt and file2.txt, sorts them, and removes any duplicate lines.

14 ls -l | grep "^d"

- Lists all files and directories in the current directory, showing detailed information, and filters to display only directories (lines starting with "d" represent directories).

15 grep -r "pattern" /path/to/directory/

- Recursively searches for the string "pattern" in all files within the specified directory and its subdirectories.

16 cat file1.txt file2.txt | sort | uniq -d

- Concatenates the contents of file1.txt and file2.txt, sorts the lines, and filters to show only the duplicate lines (lines that appear in both files).

17 chmod 644 file.txt

- Changes the permissions of file.txt so that the owner has read and write permissions (6), while the group and others have read-only permissions (4).

18 cp -r source_directory destination_directory

- Copies the entire source_directory (and its contents) recursively into the destination_directory.

19 find /path/to/search -name "*.txt"

- Searches for all files ending with .txt in the specified directory and its subdirectories.

20 chmod u+x file.txt

- Adds execute permissions for the user (owner) of file.txt.

21 echo $PATH

- Prints the value of the PATH environment variable, which lists directories where executable files are located.

Part-B

Identify True or False:

1 True —

ls is used to list files and directories in the current directory.

2 True —

mv is used to move (or rename) files and directories.

3 False —

cd is used to change the current directory, not to copy files or directories.

4 True —

pwd stands for "print working directory" and displays the full path of the current directory.

5 True —

grep is used to search for patterns within files.

6 True —

chmod 755 file.txt gives read, write, and execute permissions to the owner (7), nd read and execute permissions to group and others (5).

7 True —

mkdir -p directory1/directory2 creates both directory1 and directory2, ensuring that directory1 is created if it doesn’t already exist.

8 True — rm -rf file.txt forcefully deletes file.txt without asking for confirmation

Sub Question- Identify the Incorrect Commands:

1 Incorrect — chmodx is not a valid command. The correct command is chmod, used to change file permissions.

2 Incorrect — cpy is not a valid command. The correct command is cp, which is used to copy files and directories.

3 Incorrect — mkfile is not a standard command. The correct command is touch, which is used to create an empty file.

4 Incorrect — catx is not a valid command. The correct command is cat, which is used to concatenate and display file contents.

6 Incorrect — rn is not a valid command. The correct command is mv, which is used to rename files and directories.

Part -C

Question 1: Shell script to print "Hello, World!" to the terminal.

```
#! /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.

```
#! /bin/bash

name="CDAC Mumbai"

echo $name
```

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

```
#!/bin/bash

read -p "Enter a number: " number

echo "You entered: $number"
```

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

```
#!/bin/bash

num1=5

num2=3

result=$((num1 + num2))

echo "The sum is: $result"
```

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

```
#!/bin/bash

read -p "Enter a number: " number

if (( number % 2 == 0 )); then
```

```
  echo "Even"
else
  echo "Odd"
fi
```

Question 6: Shell script that uses a for loop to print numbers from 1 to 5.

```
#!/bin/bash
for i in {1..5}
do
  echo $i
```

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

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

Question 8: 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".

```
bash
#!/bin/bash
if [ -f "file.txt" ]; then
  echo "File exists"
else
  echo "File does not exist"
```

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

```
bash
#!/bin/bash
read -p "Enter a number: " number
if [ $number -gt 10 ]; then
```

```
  echo "The number is greater than 10"
else
  echo "The number is not greater than 10
```

Question 10: Shell script that uses nested for loops to print a multiplication table for numbers from 1 to 5.

Bash

```
#!/bin/bash
for i in {1..5}
do
  for j in {1..5}
  do
    result=$((i * j))
    echo -n "$result "
  echo
```

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.

#!/bin/bash

```
while true
do
  read -p "Enter a number: " number
  if [ $number -lt 0 ]; then
    break
  else
    square=$((number * number))
    echo "The square of $number is: $square"
  fi
done
```

echo "Negative number entered, exiting."

Part -E

Question number 1

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

Now, step 1

1. P1 starts at time 0 and has a burst time of 5, so it completes at time: CTP1=ArrivalTimeP1+BurstTimeP1=0+5=5CT_{P1} = Arrival Time_{P1} + Burst Time_{P1} = 0 + 5 = 5CTP1=ArrivalTimeP1+BurstTimeP1=0+5=5
2. P2 starts when P1 finishes, which is at time 5. P2 has a burst time of 3, so it completes at time: CTP2=StartTimeP2+BurstTimeP2=5+3=8CT_{P2} = Start Time_{P2} + Burst Time_{P2} = 5 + 3 = 8CTP2=StartTimeP2+BurstTimeP2=5+3=8
3. P3 starts when P2 finishes, which is at time 8. P3 has a burst time of 6, so it completes at time: CTP3=StartTimeP3+BurstTimeP3=8+6=14CT_{P3} = Start Time_{P3} + Burst Time_{P3} = 8 + 6 = 14CTP3=StartTimeP3+BurstTimeP3=8+6=14

Step 2: Calculate Turnaround Time (TAT)

Turnaround Time is the total time taken by the process from arrival to completion. It can be calculated as: TAT=CompletionTime−ArrivalTimeTAT = Completion Time - Arrival TimeTAT=CompletionTime−ArrivalTime

1. TAT_{P1} = CTP1−ArrivalTimeP1=5−0=5CT_{P1} - Arrival Time_{P1} = 5 - 0 = 5CTP1 −ArrivalTimeP1=5−0=5
2. TAT_{P2} = CTP2−ArrivalTimeP2=8−1=7CT_{P2} - Arrival Time_{P2} = 8 - 1 = 7CTP2 −ArrivalTimeP2=8−1=7
3. TAT_{P3} = CTP3−ArrivalTimeP3=14−2=12CT_{P3} - Arrival Time_{P3} = 14 - 2 = 12CTP3−ArrivalTimeP3=14−2=12

Step 3: Calculate Waiting Time (WT)

Waiting Time is the time a process spends waiting in the ready queue before it gets executed. It can be calculated as:

WT=TurnaroundTime−BurstTimeWT = Turnaround Time - Burst TimeWT=TurnaroundTime−BurstTime

1. WT_{P1} = TATP1−BurstTimeP1=5−5=0TAT_{P1} - Burst Time_{P1} = 5 - 5 = 0TATP1−BurstTimeP1=5−5=0
2. WT_{P2} = TATP2−BurstTimeP2=7−3=4TAT_{P2} - Burst Time_{P2} = 7 - 3 = 4TATP2−BurstTimeP2=7−3=4
3. WT_{P3} = TATP3−BurstTimeP3=12−6=6TAT_{P3} - Burst Time_{P3} = 12 - 6 = 6TATP3−BurstTimeP3=12−6=6

Step 4: Calculate Average Waiting Time

The average waiting time is the total waiting time of all processes divided by the number of processes.
Average Waiting Time=WTP1+WTP2+WTP33\text{Average Waiting Time} = \frac{\text{WT}_{P1} + \text{WT}_{P2} + \text{WT}_{P3}}{3}Average Waiting Time=3WTP1+WTP2+WTP3

Substitute the values:
Average Waiting Time=0+4+63=103=3.33\text{Average Waiting Time} = \frac{0 + 4 + 6}{3} = \frac{10}{3} = 3.33Average Waiting Time=30+4+6=310=3.33

Final Answer:

The average waiting time for the processes using FCFS scheduling is 3.33 units.

Question number 2 from pdf…

Step 2: Sort the processes by burst time and arrival time

At time 0:

- P1 is the only process, so it executes first.

At time 3 (after P1 finishes):

- The remaining processes are P2 (arrival time 1), P3 (arrival time 2), and P4 (arrival time 3).
- P3 has the shortest burst time (1), so it executes next.

At time 4 (after P3 finishes):

- Remaining processes are P4 (burst time 4) and P2 (burst time 5).
- P4 has the shorter burst time, so it executes next.

At time 8 (after P4 finishes):

- Only P2 remains, and it executes last.

Step 3: Calculate Completion Time (CT)

Completion time is the time at which a process finishes execution.

1. **P1: Starts at time 0 and finishes at time 3.**
   - **CTP1=3CT_{P1} = 3CTP1=3**
2. **P3: Starts at time 3 and finishes at time 4 (since its burst time is 1).**
   - **CTP3=4CT_{P3} = 4CTP3=4**
3. **P4: Starts at time 4 and finishes at time 8 (since its burst time is 4).**
   - **CTP4=8CT_{P4} = 8CTP4=8**
4. **P2: Starts at time 8 and finishes at time 13 (since its burst time is 5).**
   - **CTP2=13CT_{P2} = 13CTP2=13**

Step 4: Calculate Turnaround Time (TAT)

Turnaround Time is the total time a process takes from arrival to completion. It is calculated as:

TAT=CompletionTime−ArrivalTimeTAT = Completion Time - Arrival TimeTAT=CompletionTime−ArrivalTime

1. **TAT_{P1} = CTP1−ArrivalTimeP1=3−0=3CT_{P1} - Arrival Time_{P1} = 3 - 0 = 3CTP1 −ArrivalTimeP1=3−0=3**
2. **TAT_{P3} = CTP3−ArrivalTimeP3=4−2=2CT_{P3} - Arrival Time_{P3} = 4 - 2 = 2CTP3 −ArrivalTimeP3=4−2=2**
3. **TAT_{P4} = CTP4−ArrivalTimeP4=8−3=5CT_{P4} - Arrival Time_{P4} = 8 - 3 = 5CTP4 −ArrivalTimeP4=8−3=5**
4. **TAT_{P2} = CTP2−ArrivalTimeP2=13−1=12CT_{P2} - Arrival Time_{P2} = 13 - 1 = 12CTP2−ArrivalTimeP2=13−1=12**

Step 5: Calculate Average Turnaround Time

The average turnaround time is the total turnaround time of all processes divided by the number of processes. It is calculated as:

Average Turnaround Time=TATP1+TATP3+TATP4+TATP24\text{Average Turnaround Time} = \frac{TAT_{P1} + TAT_{P3} + TAT_{P4} + TAT_{P2}}{4}Average Turnaround Time=4TATP1 +TATP3+TATP4+TATP2

Substitute the values:

Average Turnaround Time=3+2+5+124=224=5.5\text{Average Turnaround Time} = \frac{3 + 2 + 5 + 12}{4} = \frac{22}{4} = 5.5Average Turnaround Time=43+2+5+12=422=5.5

Final Answer:

The average turnaround time using Shortest Job First (SJF) scheduling is 5.5 units.

Question number 3 from pdf

Step 1: Order the processes based on priority and arrival time

- P2 has the highest priority (priority 1), so it executes first.
- P4 has the second-highest priority (priority 2), so it executes next.
- P1 has the next priority (priority 3), so it executes after P4.
- P3 has the lowest priority (priority 4), so it executes last.

Step 2: Calculate Completion Time (CT)

We need to find out when each process completes execution. The process is executed according to its arrival time and priority.

1. P2 starts at time 1 (arrival time of P2) and runs for 4 units (burst time of P2). So, it completes at time:
    - CTP2=1+4=5CT_{P2} = 1 + 4 = 5CTP2=1+4=5
2. P4 starts at time 5 (since it has the next highest priority and is scheduled after P2) and runs for 2 units. So, it completes at time:
    - CTP4=5+2=7CT_{P4} = 5 + 2 = 7CTP4=5+2=7
3. P1 starts at time 7 and runs for 6 units (its burst time). So, it completes at time:
    - CTP1=7+6=13CT_{P1} = 7 + 6 = 13CTP1=7+6=13
4. P3 starts at time 13 (after P1 finishes) and runs for 7 units. So, it completes at time:
    - CTP3=13+7=20CT_{P3} = 13 + 7 = 20CTP3=13+7=20

Step 3: Calculate Turnaround Time (TAT)

Turnaround time is the total time a process takes from arrival to completion. It is calculated as:

TAT=CompletionTime−ArrivalTimeTAT = Completion Time - Arrival TimeTAT=CompletionTime−ArrivalTime

1. TAT_{P2} = CTP2−ArrivalTimeP2=5−1=4CT_{P2} - Arrival Time_{P2} = 5 - 1 = 4CTP2 −ArrivalTimeP2=5−1=4
2. TAT_{P4} = CTP4−ArrivalTimeP4=7−3=4CT_{P4} - Arrival Time_{P4} = 7 - 3 = 4CTP4 −ArrivalTimeP4=7−3=4
3. TAT_{P1} = CTP1−ArrivalTimeP1=13−0=13CT_{P1} - Arrival Time_{P1} = 13 - 0 = 13CTP1−ArrivalTimeP1=13−0=13

4. TAT_{P3} = CTP3−ArrivalTimeP3=20−2=18CT_{P3} - Arrival Time_{P3} = 20 - 2 = 18CTP3−ArrivalTimeP3=20−2=18

Step 4: Calculate Waiting Time (WT)

Waiting time is the time a process spends waiting in the ready queue before it starts executing. It is calculated as:

WT=TurnaroundTime−BurstTimeWT = Turnaround Time - Burst TimeWT=TurnaroundTime−BurstTime

1. WT_{P2} = TATP2−BurstTimeP2=4−4=0TAT_{P2} - Burst Time_{P2} = 4 - 4 = 0TATP2−BurstTimeP2=4−4=0
2. WT_{P4} = TATP4−BurstTimeP4=4−2=2TAT_{P4} - Burst Time_{P4} = 4 - 2 = 2TATP4−BurstTimeP4=4−2=2
3. WT_{P1} = TATP1−BurstTimeP1=13−6=7TAT_{P1} - Burst Time_{P1} = 13 - 6 = 7TATP1−BurstTimeP1=13−6=7
4. WT_{P3} = TATP3−BurstTimeP3=18−7=11TAT_{P3} - Burst Time_{P3} = 18 - 7 = 11TATP3−BurstTimeP3=18−7=11

Step 5: Calculate Average Waiting Time

The average waiting time is the total waiting time of all processes divided by the number of processes.

Average Waiting Time=WTP2+WTP4+WTP1+WTP34\text{Average Waiting Time} = \frac{WT_{P2} + WT_{P4} + WT_{P1} + WT_{P3}}{4}Average Waiting Time=4WTP2+WTP4+WTP1+WTP3

Substitute the values:

Average Waiting Time=0+2+7+114=204=5\text{Average Waiting Time} = \frac{0 + 2 + 7 + 11}{4} = \frac{20}{4} = 5Average Waiting Time=40+2+7+11=420=5

Answer for Question 3:

The average waiting time using Priority Scheduling is 5 units.

Question number 4 from pdf

Step 1: Execute the processes in Round Robin order

Let's simulate the Round Robin execution:

1. P1 starts at time 0 and gets 2 units of CPU time. So, after 2 units, P1 has 2 units remaining and the time is now 2.

2. P2 starts at time 2 and gets 2 units of CPU time. So, after 2 units, P2 has 3 units remaining and the time is now 4.

3. P3 starts at time 4 and gets 2 units of CPU time (since its burst time is 2). So, P3 finishes at time 6.

4. P4 starts at time 6 and gets 2 units of CPU time. So, after 2 units, P4 has 1 unit remaining and the time is now 8.

5. P1 resumes execution at time 8 and gets 2 more units, completing its burst time at time 10.

6. P2 resumes execution at time 10 and gets 2 more units, so it has 1 unit remaining. The time is now 12.

7. P4 resumes execution at time 12 and completes its remaining 1 unit of burst time. So, P4 finishes at time 13.

8. P2 resumes execution at time 13 and completes its remaining 1 unit of burst time. So, P2 finishes at time 14.

Step 2: Calculate Completion Time (CT)

The completion times for each process:

1. **P1 completes at time 10.**
2. **P2 completes at time 14.**
3. **P3 completes at time 6.**
4. **P4 completes at time 13.**

Step 3: Calculate Turnaround Time (TAT)

Turnaround time is the total time a process takes from arrival to completion. It is calculated as:

TAT=CompletionTime−ArrivalTimeTAT = Completion Time - Arrival TimeTAT=CompletionTime−ArrivalTime

1. **TAT_{P1} = CTP1−ArrivalTimeP1=10−0=10CT_{P1} - Arrival Time_{P1} = 10 - 0 = 10CTP1−ArrivalTimeP1=10−0=10**
2. **TAT_{P2} = CTP2−ArrivalTimeP2=14−1=13CT_{P2} - Arrival Time_{P2} = 14 - 1 = 13CTP2−ArrivalTimeP2=14−1=13**
3. **TAT_{P3} = CTP3−ArrivalTimeP3=6−2=4CT_{P3} - Arrival Time_{P3} = 6 - 2 = 4CTP3 −ArrivalTimeP3=6−2=4**

4. **TAT_{P4} = CTP4−ArrivalTimeP4=13−3=10CT_{P4} - Arrival Time_{P4} = 13 - 3 = 10CTP4−ArrivalTimeP4=13−3=10**

Step 4: Calculate Average Turnaround Time

The average turnaround time is the total turnaround time of all processes divided by the number of processes:

Average Turnaround Time=TATP1+TATP2+TATP3+TATP44\text{Average Turnaround Time} = \frac{TAT_{P1} + TAT_{P2} + TAT_{P3} + TAT_{P4}}{4}Average Turnaround Time=4TATP1 +TATP2+TATP3+TATP4

Substitute the values:

Average Turnaround Time=10+13+4+104=374=9.25\text{Average Turnaround Time} = \frac{10 + 13 + 4 + 10}{4} = \frac{37}{4} = 9.25Average Turnaround Time=410+13+4+10=437 =9.25

Answer for Question 4:

The average turnaround time using Round Robin scheduling with a time quantum of 2 units is 9.25 units.

Question number 5 from pdf...

In a program that uses the fork() system call, a child process is created as a copy of the parent process. However, both processes run independently after the fork() is called. Importantly, the parent and child processes do not share memory, meaning they have separate copies of variables in memory.

1. **Initially, the parent process has a variable x with a value of 5.**
2. **The fork() system call is executed. At this point:**
   - **The parent process continues executing with its own copy of the variable x, which is still 5.**
   - **The child process is created as a copy of the parent process, so it also has its own copy of the variable x, which is also 5 at this point (this is a copy-on-write behavior).**
3. **After forking, both the parent and the child process increment the value of x by 1. This happens independently in both processes:**
   - **The parent process increments its own x by 1, so now x in the parent process is 6.**
   - **The child process also increments its own x by 1, so now x in the child process is 6 as well.**

Final Values of x:

- **Parent process: x = 6**
- **Child process: x = 6**

both processes will have the final value of x equal to 6.