RAMANUJAN COLLEGE

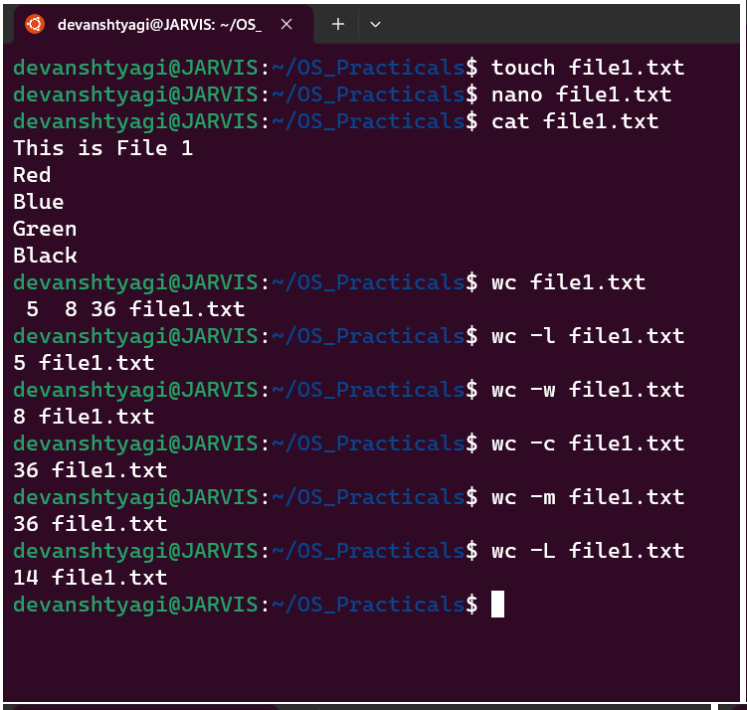
(University of Delhi)

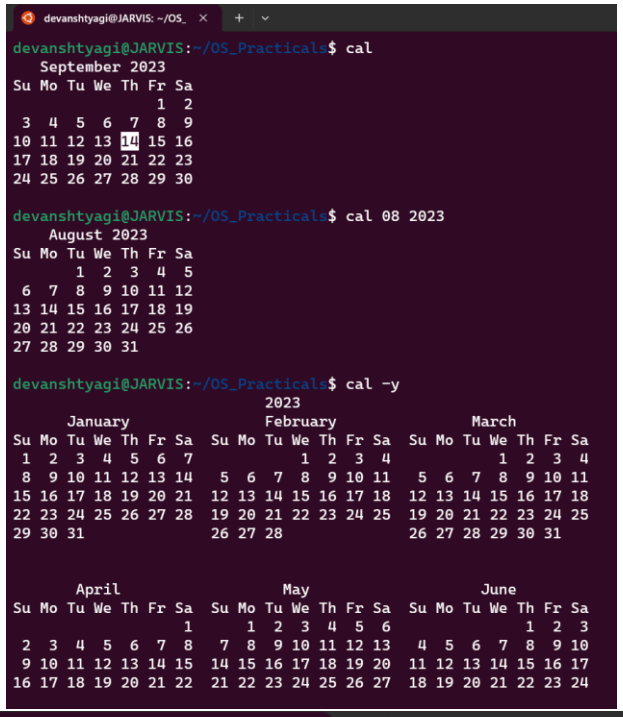
OPERATING SYSTEM (PRACTICAL FILE)

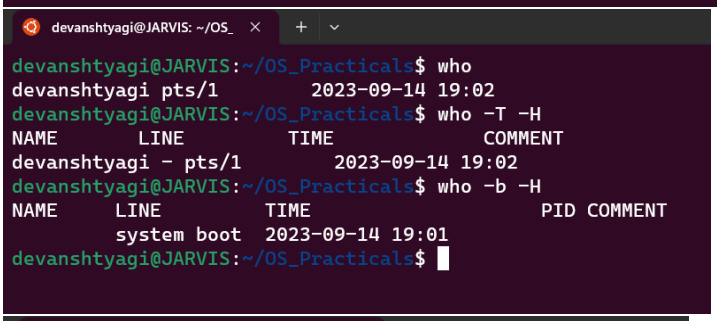
Name : Devansh Tyagi
College Roll No. : 20221416
University Roll No. : 22020570005
Course : B.Sc. (Hons.) Computer Science
Year : Second

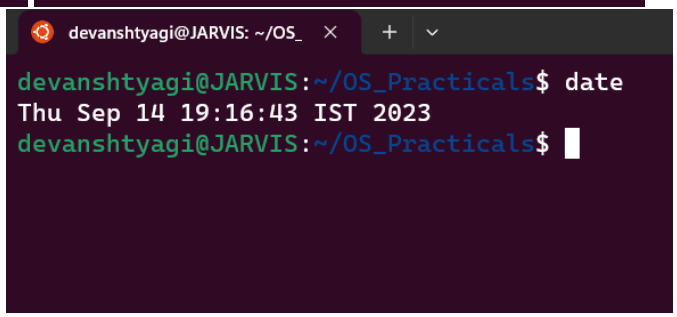
1. Execute various LINUX commands for:

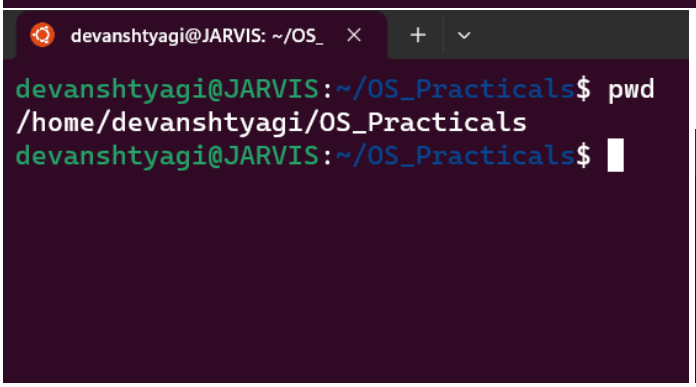
i. Information Maintenance: wc, clear, cal, who, date, pwd

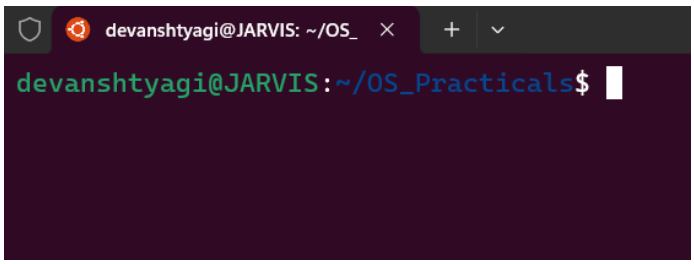












ii. File Management: cat, cp, rm, mv, cmp, comm, diff, find, grep, awk

```
devanshtyagi@JARVIS: ~/OS_
devanshtyagi@JARVIS:~/OS_Practicals$ cat file1.txt
This is File 1
Red
Blue
Green
Black
devanshtyagi@JARVIS:~/OS_Practicals$
```

```
devanshtyagi@JARVIS: ~/OS_
devanshtyagi@JARVIS:~/OS_Practicals$ cat file1.txt
This is File 1
Red
Blue
Green
Black
devanshtyagi@JARVIS:~/OS_Practicals$ cat file2.txt
devanshtyagi@JARVIS:~/OS_Practicals$ cp file1.txt file2.txt
devanshtyagi@JARVIS:~/OS_Practicals$ cat file2.txt
This is File 1
Red
Blue
Green
Black
devanshtyagi@JARVIS:~/OS_Practicals$
```

```
devanshtyagi@JARVIS: ~/OS_
devanshtyagi@JARVIS:~/OS_Practicals$ touch file3.txt
devanshtyagi@JARVIS:~/OS_Practicals$ nano file3.txt
devanshtyagi@JARVIS:~/OS_Practicals$ cat file3.txt
This file is going to be removed

devanshtyagi@JARVIS:~/OS_Practicals$ rm file3.txt
devanshtyagi@JARVIS:~/OS_Practicals$ cat file3.txt
cat: file3.txt: No such file or directory
devanshtyagi@JARVIS:~/OS_Practicals$
```

```
devanshtyagi@JARVIS: ~/OS_
devanshtyagi@JARVIS:~/OS_Practicals$ mkdir NewDir
devanshtyagi@JARVIS:~/OS_Practicals$ touch file3.txt
devanshtyagi@JARVIS:~/OS_Practicals$ nano file3.txt
devanshtyagi@JARVIS:~/OS_Practicals$ cat file3.txt
This file is going to be moved.

devanshtyagi@JARVIS:~/OS_Practicals$ mv file3.txt NewDir
devanshtyagi@JARVIS:~/OS_Practicals$ cat file3.txt
cat: file3.txt: No such file or directory
devanshtyagi@JARVIS:~/OS_Practicals$ cd NewDir
devanshtyagi@JARVIS:~/OS_Practicals/NewDir$ cat file3.txt
This file is going to be moved.

devanshtyagi@JARVIS:~/OS_Practicals/NewDir$
```

```
devanshtyagi@JARVIS: ~/OS_
devanshtyagi@JARVIS:~/OS_Practicals$ cat file1.txt
This is File 1
Red
Blue
Green
Black
devanshtyagi@JARVIS:~/OS_Practicals$ cat file2.txt
This is File 1
Red
Blue
Green
Black
devanshtyagi@JARVIS:~/OS_Practicals$ cat file3.txt
New File for Paractical
Green
Yellow
Red

devanshtyagi@JARVIS:~/OS_Practicals$ comm file1.txt file2.txt
                This is File 1
                Red
                Blue
                Green
                Black
devanshtyagi@JARVIS:~/OS_Practicals$ comm file1.txt file3.txt
        New File for Paractical
comm: file 2 is not in sorted order
        Green
This is File 1
comm: file 1 is not in sorted order
Red
Blue
Green
Black
        Yellow
```

```
devanshtyagi@JARVIS: ~/OS_
devanshtyagi@JARVIS:~/OS_Practicals$ touch file3.txt
devanshtyagi@JARVIS:~/OS_Practicals$ nano file3.txt
devanshtyagi@JARVIS:~/OS_Practicals$ cat file1.txt
This is File 1
Red
Blue
Green
Black
devanshtyagi@JARVIS:~/OS_Practicals$ cat file2.txt
This is File 1
Red
Blue
Green
Black
devanshtyagi@JARVIS:~/OS_Practicals$ cat file3.txt
New File for Paractical
Green
Yellow
Red

devanshtyagi@JARVIS:~/OS_Practicals$ cmp file1.txt file2.txt
devanshtyagi@JARVIS:~/OS_Practicals$ cmp file1.txt file3.txt
file1.txt file3.txt differ: byte 1, line 1
devanshtyagi@JARVIS:~/OS_Practicals$
```

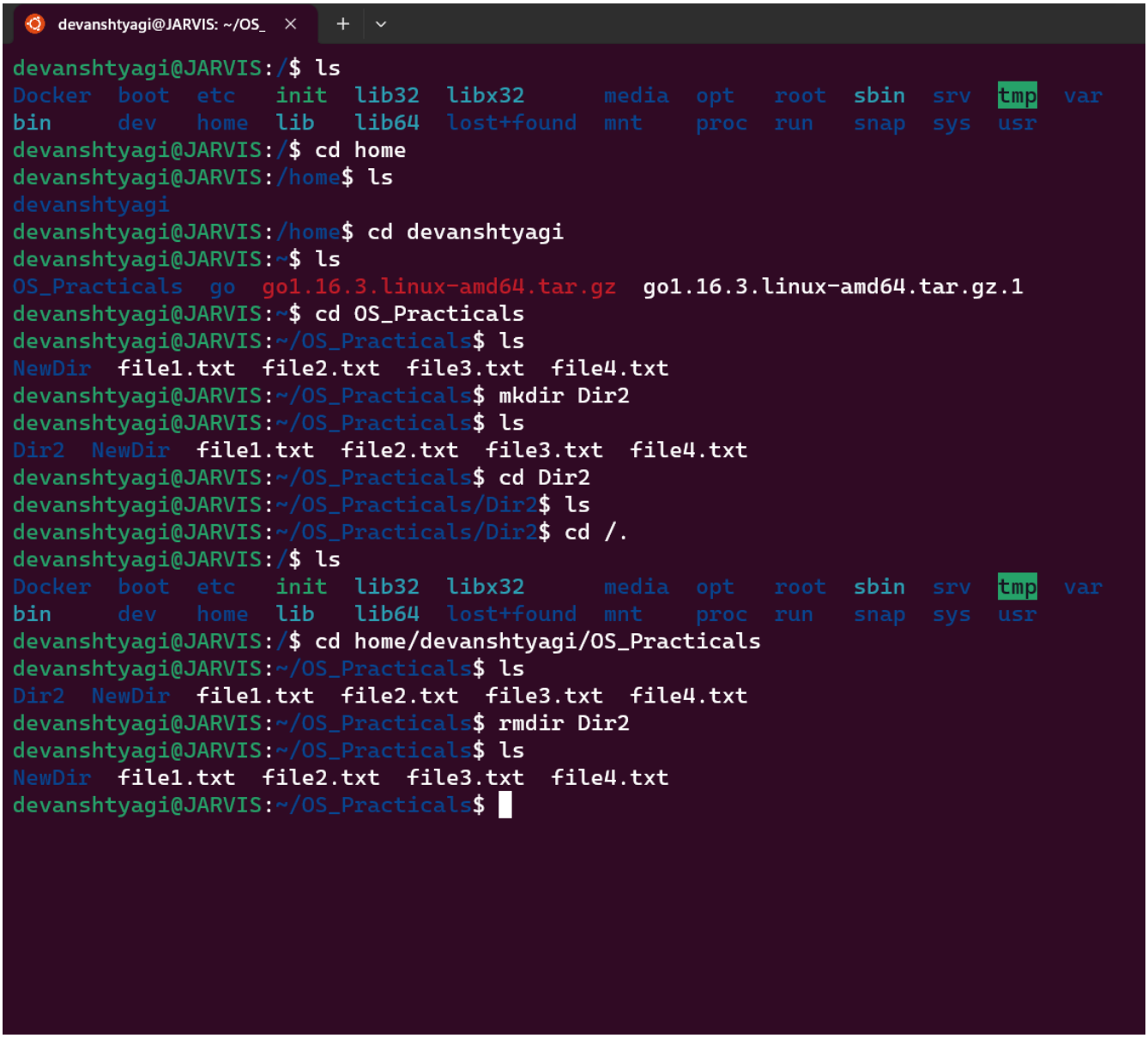
```
devanshtyagi@JARVIS: ~/OS_
devanshtyagi@JARVIS:~/OS_Practicals$ find . -name '*.txt'
./NewDir/file3.txt
./file1.txt
./file2.txt
./file3.txt
devanshtyagi@JARVIS:~/OS_Practicals$
```

```
devanshtyagi@JARVIS: ~/OS_
devanshtyagi@JARVIS:~/OS_Practicals$ cat file1.txt
This is File 1
Red
Blue
Green
Black
devanshtyagi@JARVIS:~/OS_Practicals$ cat file2.txt
This is File 1
Red
Blue
Green
Black
devanshtyagi@JARVIS:~/OS_Practicals$ cat file3.txt
New File for Paractical
Green
Yellow
Red

devanshtyagi@JARVIS:~/OS_Practicals$ diff file1.txt file2.txt
devanshtyagi@JARVIS:~/OS_Practicals$ diff file1.txt file3.txt
1,3c1
< This is File 1
< Red
< Blue
---
> New File for Paractical
5c3,5
< Black
---
> Yellow
> Red
>
devanshtyagi@JARVIS:~/OS_Practicals$
```

```
devanshtyagi@JARVIS: ~/OS_
devanshtyagi@JARVIS:~/OS_Practicals$ cat file1.txt
This is File 1
Red
Blue
Green
Black
devanshtyagi@JARVIS:~/OS_Practicals$ grep "Red" file1.txt
Red
devanshtyagi@JARVIS:~/OS_Practicals$ grep -r "Red" *
file1.txt:Red
file2.txt:Red
file3.txt:Red
devanshtyagi@JARVIS:~/OS_Practicals$
```

```
devanshtyagi@JARVIS: ~/OS_
devanshtyagi@JARVIS:~/OS_Practicals$ cat file4.txt
Car Color
BMW White
Nano Black
Audi Red

devanshtyagi@JARVIS:~/OS_Practicals$ awk '{print $1 "\t" $2}' file4.txt
Car     Color
BMW     White
Nano    Black
Audi    Red

devanshtyagi@JARVIS:~/OS_Practicals$
```

iii. Directory Management : cd, mkdir, rmdir, ls



2. Execute various LINUX commands for:

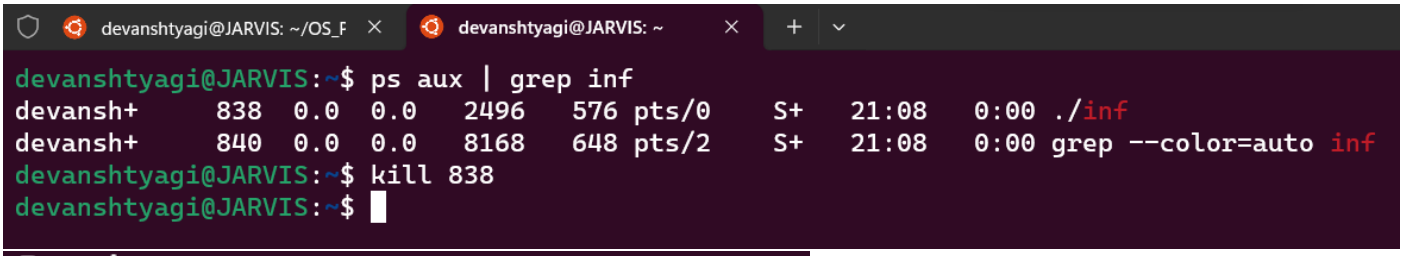
i. Process Control: fork, getpid, ps, kill, sleep

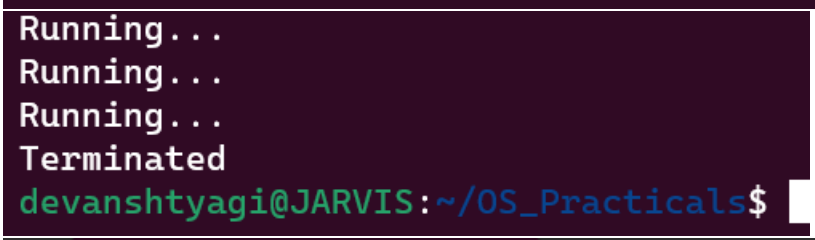
```
devanshtyagi@JARVIS: ~/OS_
devanshtyagi@JARVIS:~/OS_Practicals$ cat fork.c
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>

int main(){
    int x = 1;
    if(fork() == 0){
        printf("Hello from Child x = %d\n", ++x);
    }
    else{
        printf("Hello from Parent x = %d\n", --x);
    }
    return 0;
}
devanshtyagi@JARVIS:~/OS_Practicals$ gcc fork.c -o fork
devanshtyagi@JARVIS:~/OS_Practicals$ ./fork
Hello from Parent x = 0
Hello from Child x = 2
devanshtyagi@JARVIS:~/OS_Practicals$
```

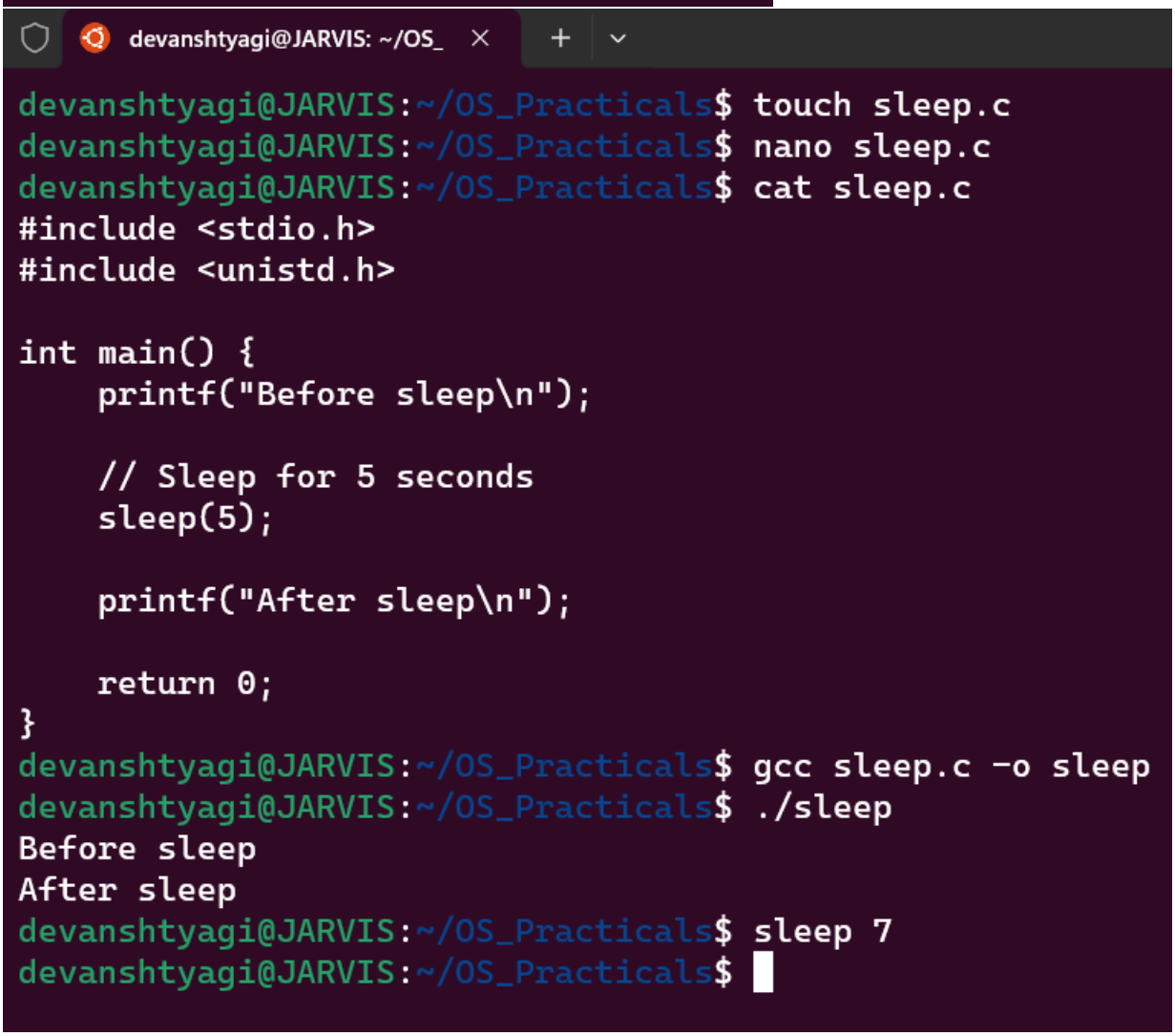
```
devanshtyagi@JARVIS: ~/OS_
devanshtyagi@JARVIS:~/OS_Practicals$ cat fork_id.c
#include <stdio.h>
#include <unistd.h>

int main() {
    pid_t child_pid = fork();

    if (child_pid < 0) {
        fprintf(stderr, "Fork failed\n");
        return 1;
    } else if (child_pid == 0) {
        printf("Child process: PID = %d\n", getpid());
    } else {
        printf("Parent process: PID = %d, Child PID = %d\n", getpid(), child_pid);
    }

    printf("This is printed by both parent and child processes\n");

    return 0;
}
devanshtyagi@JARVIS:~/OS_Practicals$ gcc fork_id.c -o fork_id
devanshtyagi@JARVIS:~/OS_Practicals$ ./fork_id
Parent process: PID = 926, Child PID = 927
This is printed by both parent and child processes
Child process: PID = 927
This is printed by both parent and child processes
devanshtyagi@JARVIS:~/OS_Practicals$
```

```
devanshtyagi@JARVIS: ~/OS_
devanshtyagi@JARVIS:~/OS_Practicals$ ps
    PID TTY          TIME CMD
    528 pts/0    00:00:00 bash
    948 pts/0    00:00:00 ps
devanshtyagi@JARVIS:~/OS_Practicals$
```

```
devanshtyagi@JARVIS: ~/OS_
devanshtyagi@JARVIS: ~
devanshtyagi@JARVIS:~/OS_Practicals$ touch inf.c
devanshtyagi@JARVIS:~/OS_Practicals$ nano inf.c
devanshtyagi@JARVIS:~/OS_Practicals$ cat inf.c
#include <stdio.h>
#include <unistd.h>

int main() {
    while (1) {
        printf("Running...\n");
        sleep(2);  // Simulate some work being done
    }

    return 0;
}
devanshtyagi@JARVIS:~/OS_Practicals$ gcc inf.c -o inf
devanshtyagi@JARVIS:~/OS_Practicals$ ./inf
Running...
Running...
Running...
Running...

Running...
Running...
Running...
```


devanshtyagi@JARVIS: ~/OS_F
devanshtyagi@JARVIS: ~
devanshtyagi@JARVIS:~$ ps aux | grep inf
devansh+ 838 0.0 0.0 2496 576 pts/0 S+ 21:08 0:00 ./inf
devansh+ 840 0.0 0.0 8168 648 pts/2 S+ 21:08 0:00 grep --color=auto inf
devanshtyagi@JARVIS:~$ kill 838
devanshtyagi@JARVIS:~$


Running...
Running...
Running...
Terminated
devanshtyagi@JARVIS:~/OS_Practicals$


devanshtyagi@JARVIS: ~/OS_
devanshtyagi@JARVIS:~/OS_Practicals$ touch sleep.c
devanshtyagi@JARVIS:~/OS_Practicals$ nano sleep.c
devanshtyagi@JARVIS:~/OS_Practicals$ cat sleep.c
#include <stdio.h>
#include <unistd.h>
int main() {
printf("Before sleep\n");
// Sleep for 5 seconds
sleep(5);
printf("After sleep\n");
return 0;
}
devanshtyagi@JARVIS:~/OS_Practicals$ gcc sleep.c -o sleep
devanshtyagi@JARVIS:~/OS_Practicals$ ./sleep
Before sleep
After sleep
devanshtyagi@JARVIS:~/OS_Practicals$ sleep 7
devanshtyagi@JARVIS:~/OS_Practicals$

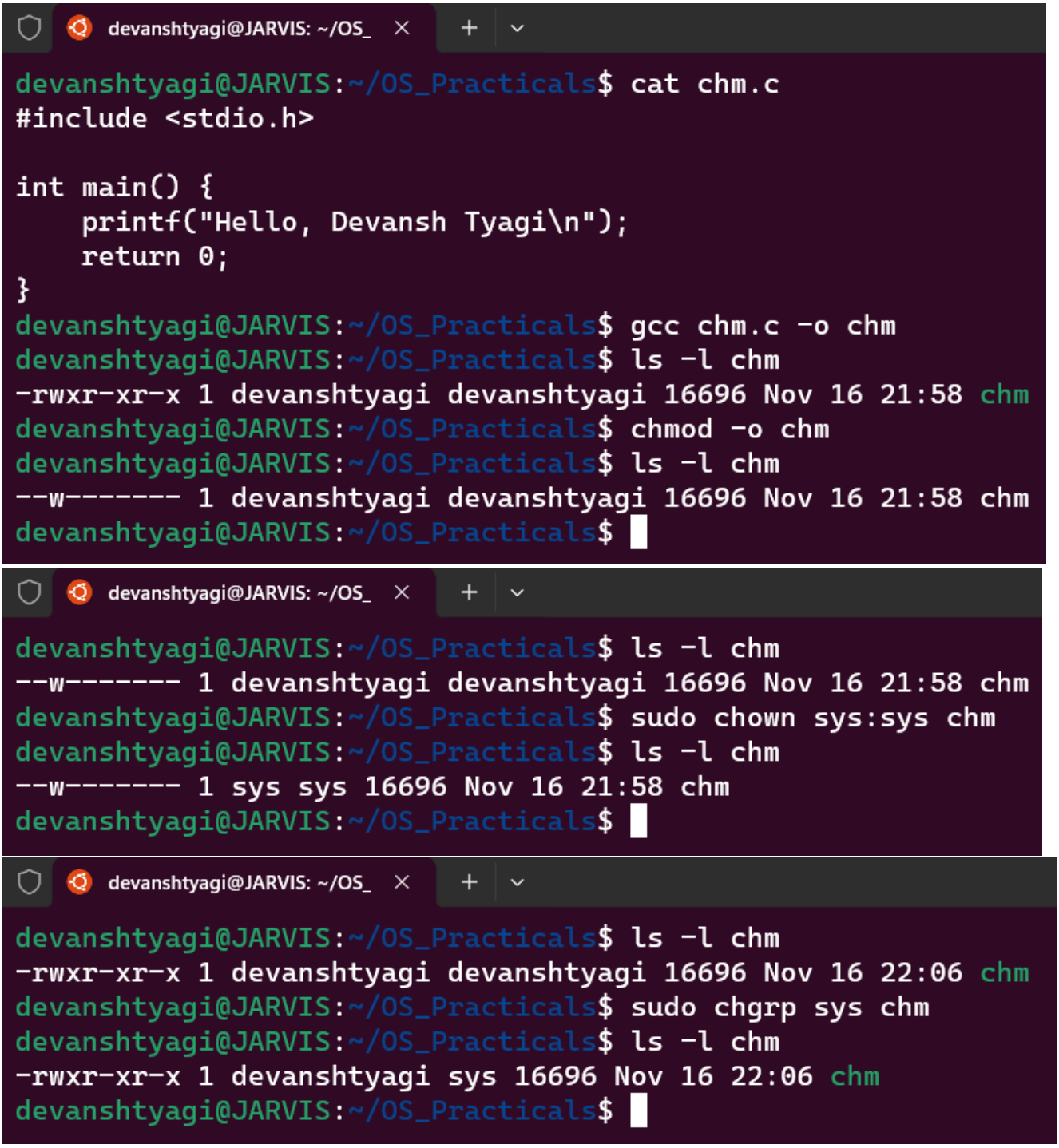
ii. Communication: Input-output redirection, Pipe

```
devanshtyagi@JARVIS: ~/OS_
devanshtyagi@JARVIS:~/OS_Practicals$ cat input_p.c
#include <stdio.h>

int main() {
    char buffer[100];
    fgets(buffer, sizeof(buffer), stdin);

    printf("You entered: %s", buffer);

    return 0;
}
devanshtyagi@JARVIS:~/OS_Practicals$ cat input.txt
This is Input file.
devanshtyagi@JARVIS:~/OS_Practicals$ gcc input_p.c -o input_p
devanshtyagi@JARVIS:~/OS_Practicals$ ./input_p < input.txt
You entered: This is Input file.
devanshtyagi@JARVIS:~/OS_Practicals$
```

```
devanshtyagi@JARVIS: ~/OS_
devanshtyagi@JARVIS:~/OS_Practicals$ cat output_p.c
#include <stdio.h>

int main() {
    printf("This is the output message.\n");

    return 0;
}
devanshtyagi@JARVIS:~/OS_Practicals$ cat output.txt
devanshtyagi@JARVIS:~/OS_Practicals$ gcc output_p.c -o output_p
devanshtyagi@JARVIS:~/OS_Practicals$ ./output_p > output.txt
devanshtyagi@JARVIS:~/OS_Practicals$ cat output.txt
This is the output message.
devanshtyagi@JARVIS:~/OS_Practicals$
```

```
devanshtyagi@JARVIS: ~/OS_
devanshtyagi@JARVIS:~/OS_Practicals$ touch producer.c
devanshtyagi@JARVIS:~/OS_Practicals$ touch consumer.c
devanshtyagi@JARVIS:~/OS_Practicals$ nano producer.c
devanshtyagi@JARVIS:~/OS_Practicals$ nano consumer.c
devanshtyagi@JARVIS:~/OS_Practicals$ cat producer.c
#include <stdio.h>

int main() {
    printf("Data from producer\n");
    return 0;
}
devanshtyagi@JARVIS:~/OS_Practicals$ cat consumer.c
#include <stdio.h>

int main() {
    char buffer[100];
    printf("Consumer is waiting for data...\n");
    fgets(buffer, sizeof(buffer), stdin);
    printf("Consumer received: %s", buffer);
    return 0;
}
devanshtyagi@JARVIS:~/OS_Practicals$ gcc producer.c -o producer
devanshtyagi@JARVIS:~/OS_Practicals$ gcc consumer.c -o consumer
devanshtyagi@JARVIS:~/OS_Practicals$ ./producer | ./consumer
Consumer is waiting for data...
Consumer received: Data from producer
devanshtyagi@JARVIS:~/OS_Practicals$
```

iii. Protection Management: chmod, chown, chgrp



3. Write a program(using fork() and/or exec() commands) where parent and child execute:

i. same program, same code.

```
devanshtyagi@JARVIS: ~/OS_
devanshtyagi@JARVIS:~/OS_Practicals$ cat same_program_same_code.c
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>

int main(){
        pid_t pid, par;
        par = fork();
        pid = getpid();
        if(par<0){
                fprintf(stderr, "Fork Failed!!!");
                return 1;
        }
        printf("Output of fork id : %d \n", par);
        printf("Process ID is : %d \n", pid);
        return 0;
}
devanshtyagi@JARVIS:~/OS_Practicals$ gcc same_program_same_code.c -o same_program_same_code
devanshtyagi@JARVIS:~/OS_Practicals$ ./same_program_same_code
Output of fork id : 664
Output of fork id : 0
Process ID is : 663
Process ID is : 664
devanshtyagi@JARVIS:~/OS_Practicals$
```

ii. same program, different code.

```
devanshtyagi@JARVIS: ~/OS_
devanshtyagi@JARVIS:~/OS_Practicals$ cat same_program_different_code.c
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>

int main(){
        int pid;
        pid = fork();
        if(pid<0){
                printf("Fork Filed\n");
                exit(1);
        }
        else if(pid==0){
                printf("\nHello i am child process...\n");
                printf("The PID is %d \n", getpid());
                exit(0);
        }
        else{
                printf("\nHello i am parent process...");
                printf("\nThe actual PID is %d \n", getpid());
                exit(1);
        }
}
devanshtyagi@JARVIS:~/OS_Practicals$ gcc same_program_different_code.c -o same_program_different_code
devanshtyagi@JARVIS:~/OS_Practicals$ ./same_program_different_code


Hello i am child process...
Hello i am parent process...
The actual PID is 686
The PID is 687
devanshtyagi@JARVIS:~/OS_Practicals$
```

iii. before terminating, the parent waits for the child to finish its task.

devanshtyagi@JARVIS: ~/OS_

```
devanshtyagi@JARVIS:~/OS_Practicals$ cat before_terminating.c
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/wait.h>

int main(){
        int pid;
        pid = fork();
        if(pid<0){
                printf("\n Fork Filed!!!");
                exit(1);
        }
        else if(pid==0){
                printf("\nThis is child process...\n");
                printf("The PID is %d\n", getpid());
                exit(0);
        }
        else if(pid>0){
                printf("\nThis is parent process...\n");
                printf("The Actual PID is %d \n", getpid());
                wait(NULL);
                exit(1);
        }
}
devanshtyagi@JARVIS:~/OS_Practicals$ gcc before_terminating.c -o before_terminating
devanshtyagi@JARVIS:~/OS_Practicals$ ./before_terminating


This is parent process...
This is child process...
The Actual PID is 743
The PID is 744
devanshtyagi@JARVIS:~/OS_Practicals$
```

4. Write a program to report behaviour of Linux kernel including kernel version, CPU type and model. (CPU information)

```
devanshtyagi@JARVIS: ~
devanshtyagi@JARVIS:~$ cat kernel_info.c
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>

int main()
{
        printf("\nKernel Version: \n");
        system("cat /proc/sys/kernel/osrelease");

        printf("\nCPU type & Model: \n");
        system("cat /proc/cpuinfo | awk 'NR==5 {print}'");

        return 0;
}
devanshtyagi@JARVIS:~$ gcc kernel_info.c -o kernel_info
devanshtyagi@JARVIS:~$ ./kernel_info

Kernel Version:
5.15.133.1-microsoft-standard-WSL2

CPU type & Model:
model name      : AMD Ryzen 5 3500U with Radeon Vega Mobile Gfx
devanshtyagi@JARVIS:~$
```

5. Write a program to report behaviour of Linux kernel including information on 19 configured memory, amount of free and used memory. (Memory information)

```
devanshtyagi@JARVIS: ~
devanshtyagi@JARVIS:~$ cat memory_info.c
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>

int main()
{
        printf("\nMemory Information:\n");
        system("cat /proc/meminfo | awk 'NR==1, NR==3 {print}'");

        return 0;
}
devanshtyagi@JARVIS:~$ gcc memory_info.c -o memory_info
devanshtyagi@JARVIS:~$ ./memory_info

Memory Information:
MemTotal:        9127912 kB
MemFree:         8185136 kB
MemAvailable:    8422092 kB
devanshtyagi@JARVIS:~$
```

6. Write a program to copy files using system calls.

devanshtyagi@JARVIS: ~/OS_

```
devanshtyagi@JARVIS:~/OS_Practicals$ cat copy_files.c
#include<stdio.h>
#include<unistd.h>
#include<stdlib.h>
#include<sys/types.h>
#include<sys/stat.h>
#include<fcntl.h>
#include<errno.h>

void copy(int, int);
void display(int);

int main(int argc, char *argv[])
{
        int fold, fnew;
        if(argc != 3)
        {
                printf("\nTwo Arguments Reqiured!\n");
                exit(1);
        }
        fold = open(argv[1], 0);
        if(fold == -1)
        {
                printf("\nUnable to open the file: %s\n", argv[1]);
                exit(1);
        }
        fnew = creat(argv[2], 0666);
        if(fnew == -1)
        {
                printf("\nUnable to Create the File: %s\n", argv[2]);
                exit(1);
        }
        copy(fold, fnew);
        close(fold);
        close(fnew);
```

devanshtyagi@JARVIS: ~/OS_

```
        fnew = open(argv[2], 0);
        printf("\nNew File:\n");
        display(fnew);
        close(fnew);

        exit(0);
}

void copy(int old, int fnew)
{
        int count = 0;
        char buffer[512];
        while((count = read(old, buffer, sizeof(buffer))) > 0)
        {
                write(fnew, buffer, count);
        }
}

void display(int fnew)
{
        int count = 0, i;
        char buffer[512];
        while((count = read(fnew, buffer, sizeof(buffer))) > 0)
        {
                for(i=0; i<count; i++)
                {
                        printf("%c", buffer[i]);
                }
        }
        printf("\n");
}
devanshtyagi@JARVIS:~/OS_Practicals$ gcc copy_files.c -o copy_files
devanshtyagi@JARVIS:~/OS_Practicals$ ./copy_files file1.txt file_copy.txt

New File:
```

```
This is File 1
Red
Blue
Green
Black

devanshtyagi@JARVIS:~/OS_Practicals$
```

7. Write a program to implement FCFS scheduling algorithm.

devanshtyagi@JARVIS: ~/OS_

```
devanshtyagi@JARVIS:~/OS_Practicals$ cat fcfs.c
#include<stdio.h>
void main()
{
    int bt[50], wt[80], at[80], wat[30], ft[80], tat[80];
    int i, n;
    float awt, att, sum = 0, sum1 = 0;
    char p[10][5];

    printf("\nEnter the number of processes: ");
    scanf("%d", &n);

    printf("\nEnter the process name and burst-time:\n");
    for (i = 0; i < n; i++)
        scanf("%s %d", p[i], &bt[i]);

    printf("\nEnter the arrival-time:\n");
    for (i = 0; i < n; i++)
        scanf("%d", &at[i]);

    wt[0] = 0;

    for (i = 1; i <= n; i++)
        wt[i] = wt[i - 1] + bt[i - 1]; // Corrected indexing

    ft[0] = bt[0];

    for (i = 1; i < n; i++) // Corrected indexing
        ft[i] = ft[i - 1] + bt[i];

    printf("\n\n\t\t\tGANTT CHART\n");
    printf("\n\n");
    for (i = 0; i < n; i++)
        printf("| %s\t\t", p[i]);
    printf("|\n");
```

devanshtyagi@JARVIS: ~/OS_

```
    printf("\n");

    for (i = 0; i < n; i++)
        printf("%d\t\t", wt[i]);

    printf("%d", ft[n - 1] + bt[n - 1]); // Corrected indexing
    printf("\n\n\n");

    for (i = 0; i < n; i++)
        wat[i] = wt[i] - at[i];

    for (i = 0; i < n; i++)
        tat[i] = ft[i] - at[i];

    printf("\n\t\t\t FIRST COME FIRST SERVE\n");
    printf("\n Process \t Burst-time \t Arrival-time \t Waiting-time \t Finish-time \t Turnaround-time\n");

    for (i = 0; i < n; i++)
        printf("\n\n %s \t\t %d\t\t %d \t\t %d\t\t %d \t\t%d", p[i], bt[i], at[i], wat[i], ft[i], tat[i]);

    for (i = 0; i < n; i++)
        sum = sum + wat[i];
    awt = sum / n;

    for (i = 0; i < n; i++)
        sum1 = sum1 + bt[i] + wat[i];
    att = sum1 / n;

    printf("\n\nAverage waiting time: %f", awt);
    printf("\n\nAverage turnaround time: %f", att);
    printf("\n");
}
devanshtyagi@JARVIS:~/OS_Practicals$ gcc fcfs.c -o fcfs
devanshtyagi@JARVIS:~/OS_Practicals$ ./fcfs
```

```
Enter the number of processes: 3

Enter the process name and burst-time:
A 4
B 6
C 3

Enter the arrival-time:
0
1
3


                        GANTT CHART


| A             | B             | C             |

0               4               10              16


                        FIRST COME FIRST SERVE

 Process         Burst-time      Arrival-time    Waiting-time    Finish-time     Turnaround-time

 A               4               0               0               4               4

 B               6               1               3               10              9

 C               3               3               7               13              10
```

```
Average waiting time: 3.333333

Average turnaround time: 7.666667
devanshtyagi@JARVIS:~/OS_Practicals$
```

8. Write a program to implement SJF scheduling algorithm

devanshtyagi@JARVIS: ~

```
devanshtyagi@JARVIS:~$ cat sjf.c
#include <stdio.h>

int main() {
    int A[100][4];
    int i, j, n, total = 0, index, temp;
    float avg_wt, avg_tat;

    printf("Enter number of process: ");
    scanf("%d", &n);

    printf("Enter Burst Time:\n");
    for (i = 0; i < n; i++) {
        printf("P%d: ", i + 1);
        scanf("%d", &A[i][1]);
        A[i][0] = i + 1;
    }

    for (i = 0; i < n; i++) {
        index = i;
        for (j = i + 1; j < n; j++)
            if (A[j][1] < A[index][1])
                index = j;
        temp = A[i][1];
        A[i][1] = A[index][1];
        A[index][1] = temp;

        temp = A[i][0];
        A[i][0] = A[index][0];
        A[index][0] = temp;
    }

    A[0][2] = 0;
    for (i = 1; i < n; i++) {
        A[i][2] = 0;
        for (j = 0; j < i; j++)
            A[i][2] += A[j][1];
        total += A[i][2];
    }

    avg_wt = (float)total / n;
    total = 0;

    printf("P\tBT\tWT\tTAT\n");
    for (i = 0; i < n; i++) {
        A[i][3] = A[i][1] + A[i][2];
        total += A[i][3];
        printf("P%d\t%d\t%d\t%d\n", A[i][0], A[i][1], A[i][2], A[i][3]);
    }

    avg_tat = (float)total / n;
    printf("Average Waiting Time= %f\n", avg_wt);
    printf("Average Turnaround Time= %f\n", avg_tat);

    return 0;
}
devanshtyagi@JARVIS:~$ gcc sjf.c -o sjf
devanshtyagi@JARVIS:~$ ./sjf
Enter number of process: 3
Enter Burst Time:
P1: 3
P2: 5
P3: 8
P       BT      WT      TAT
P1      3       0       3
P2      5       3       8
P3      8       8       16
Average Waiting Time= 3.666667
Average Turnaround Time= 9.000000
devanshtyagi@JARVIS:~$
```

9. Write a program to implement non-preemptive priority based scheduling algorithm.

```
devanshtyagi@JARVIS: ~
devanshtyagi@JARVIS:~$ cat non_ppbs.c
#include<stdio.h>
#define MIN -9999;
struct proc
{
    int no,at,bt,ct,wt,tat,pri,status;
};
struct proc read(int i)
{
    struct proc p;
    printf("\nProcess No: %d\n",i);
    p.no=i;
    printf("Enter Arrival Time: ");
    scanf("%d",&p.at);
    printf("Enter Burst Time: ");
    scanf("%d",&p.bt);
    printf("Enter Priority: ");
    scanf("%d",&p.pri);
    p.status=0;
    return p;
}

void main()
{
    int n,l,ct=0,remaining;
    struct proc p[10],temp;
    float avgtat=0,avgwt=0;
    printf("<--Highest Priority First Scheduling Algorithm (Non-Preemptive)-->\n");
    printf("Enter Number of Processes: ");
    scanf("%d",&n);
    for(int i=0;i<n;i++)
        p[i]=read(i+1);
    for(int i=0;i<n-1;i++)
        for(int j=0;j<n-i-1;j++)
            if(p[j].at>p[j+1].at)
            {
            temp=p[j];
            p[j]=p[j+1];
            p[j+1]=temp;
            }
    p[9].pri=MIN;
    remaining=n;
    printf("\nProcessNo\tAT\tBT\tPri\tCT\tTAT\tWT\tRT\n");
    for(ct=p[0].at;remaining!=0;)
    {
        l=9;
        for(int i=0;i<n;i++)
            if(p[i].at<=ct && p[i].status!=1 && p[i].pri>p[l].pri)
                l=i;
        p[l].ct=ct=ct+p[l].bt;
        p[l].tat=p[l].ct-p[l].at;
        avgtat+=p[l].tat;
        p[l].wt=p[l].tat-p[l].bt;
        avgwt+=p[l].wt;
        p[l].status=1;
        remaining--;
        printf("P%d\t\t%d\t%d\t%d\t%d\t%d\t%d\t%d\n",p[l].no,p[l].at,p[l].bt,p[l].pri,p[l].ct,p[l].tat,p[l].wt,p[l].wt);
    }
    avgtat/=n,avgwt/=n;
    printf("\nAverage TurnAroundTime=%f\nAverage WaitingTime=%f\n",avgtat,avgwt);
}
devanshtyagi@JARVIS:~$ gcc non_ppbs.c -o non_ppbs
devanshtyagi@JARVIS:~$ ./non_ppbs
<--Highest Priority First Scheduling Algorithm (Non-Preemptive)-->
Enter Number of Processes: 3

Process No: 1
Enter Arrival Time: 2
Enter Burst Time: 4
Enter Priority: 1

Process No: 2
Enter Arrival Time: 2
Enter Burst Time: 5
Enter Priority: 2

Process No: 3
Enter Arrival Time: 3
Enter Burst Time: 8
Enter Priority: 2

ProcessNo       AT      BT      Pri     CT      TAT     WT      RT
P2              2       5       2       7       5       0       0
P3              3       8       2       15      12      4       4
P1              2       4       1       19      17      13      13

Average TurnAroundTime=11.333333
Average WaitingTime=5.666667
devanshtyagi@JARVIS:~$
```

10. Write a program to implement SRTF scheduling algorithm.

```
devanshtyagi@JARVIS: ~
devanshtyagi@JARVIS:~$ touch srtf.c
devanshtyagi@JARVIS:~$ nano srtf.c
devanshtyagi@JARVIS:~$ cat srtf.c
#include<stdio.h>
#define MAX 9999
struct proc
{
    int no,at,bt,rt,ct,tat,wt;
};
struct proc read(int i)
{
    struct proc p;
    printf("\nProcess No: %d\n",i);
    p.no=i;
    printf("Enter Arrival Time: ");
    scanf("%d",&p.at);
    printf("Enter Burst Time: ");
    scanf("%d",&p.bt);
    p.rt=p.bt;
    return p;
}
int main()
{
    struct proc p[10],temp;
    float avgtat=0,avgwt=0;
    int n,s,remain=0,time;
    printf("<--SRTF Scheduling Algorithm (Preemptive)-->\n");
    printf("Enter Number of Processes: ");
    scanf("%d",&n);
    for(int i=0;i<n;i++)
        p[i]=read(i+1);
    for(int i=0;i<n-1;i++)
        for(int j=0;j<n-i-1;j++)
            if(p[j].at>p[j+1].at)
            {
            temp=p[j];
            p[j]=p[j+1];
            p[j+1]=temp;
            }
    printf("\nProcess\t\tAT\tBT\tCT\tTAT\tWT\n");
    p[9].rt=MAX;
    for(time=0;remain!=n;time++)
    {
        s=9;
        for(int i=0;i<n;i++)
            if(p[i].at<=time&&p[i].rt<p[s].rt&&p[i].rt>0)
                s=i;
        p[s].rt--;
        if(p[s].rt==0)
        {
            remain++;
            p[s].ct=time+1;
            p[s].tat=p[s].ct-p[s].at;
            avgtat+=p[s].tat;
            p[s].wt=p[s].tat-p[s].bt;
            avgwt+=p[s].wt;
            printf("P%d\t\t%d\t%d\t%d\t%d\t%d\n",p[s].no,p[s].at,p[s].bt,p[s].ct,p[s].tat,p[s].wt);
        }
    }
    avgtat/=n,avgwt/=n;
    printf("\nAverage TurnAroundTime=%f\nAverage WaitingTime=%f\n",avgtat,avgwt);
}
devanshtyagi@JARVIS:~$ gcc srtf.c -o srtf
devanshtyagi@JARVIS:~$ ./srtf
<--SRTF Scheduling Algorithm (Preemptive)-->
Enter Number of Processes: 3

Process No: 1
Enter Arrival Time: 2
Enter Burst Time: 5
Process No: 2
Enter Arrival Time: 1
Enter Burst Time: 5

Process No: 3
Enter Arrival Time: 4
Enter Burst Time: 8

Process         AT      BT      CT      TAT     WT
P2              1       5       6       5       0
P1              2       5       11      9       4
P3              4       8       19      15      7

Average TurnAroundTime=9.666667
Average WaitingTime=3.666667
devanshtyagi@JARVIS:~$
```

11. Write a program to calculate sum of n numbers using Pthreads. A list of n numbers is divided into two smaller list of equal size, two separate threads are used to sum the sublists.

devanshtyagi@JARVIS: ~/OS_

```
devanshtyagi@JARVIS:~/OS_Practicals$ cat pthreads.c
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>

#define MAX_SIZE 1000

// Structure to hold information about the sublist
struct SublistInfo {
    int* array;
    size_t start;
    size_t end;
    int sum;
};

// Function to calculate the sum of a sublist
void* calculateSum(void* arg) {
    struct SublistInfo* sublistInfo = (struct SublistInfo*)arg;
    sublistInfo->sum = 0;

    for (size_t i = sublistInfo->start; i < sublistInfo->end; ++i) {
        sublistInfo->sum += sublistInfo->array[i];
    }

    pthread_exit(NULL);
}

int main() {
    int n; // Number of elements in the array
    printf("Enter the number of elements: ");
    scanf("%d", &n);

    if (n <= 0 || n > MAX_SIZE) {
        printf("Invalid input size.\n");
        return 1;
    }

    int array[MAX_SIZE];
    printf("Enter %d numbers:\n", n);
    for (int i = 0; i < n; ++i) {
        scanf("%d", &array[i]);
    }

    // Creating two threads
    pthread_t thread1, thread2;

    // Dividing the array into two halves
    struct SublistInfo sublist1 = {array, 0, n / 2, 0};
    struct SublistInfo sublist2 = {array, n / 2, n, 0};

    // Creating threads and assigning tasks
    pthread_create(&thread1, NULL, calculateSum, (void*)&sublist1);
    pthread_create(&thread2, NULL, calculateSum, (void*)&sublist2);

    // Waiting for threads to finish
    pthread_join(thread1, NULL);
    pthread_join(thread2, NULL);

    // Combining the results
    int totalSum = sublist1.sum + sublist2.sum;

    printf("Sum of the numbers: %d\n", totalSum);

    return 0;
}
devanshtyagi@JARVIS:~/OS_Practicals$ gcc -o pthreads pthreads.c -pthread
devanshtyagi@JARVIS:~/OS_Practicals$ ./pthreads
Enter the number of elements: 4
Enter 4 numbers:
45
6
7
345
Sum of the numbers: 403
devanshtyagi@JARVIS:~/OS_Practicals$
```

12. Write a program to implement first-fit, best-fit and worst-fit allocation strategies.

```
devanshtyagi@JARVIS: ~/OS_
devanshtyagi@JARVIS:~/OS_Practicals$ cat first_fit.c
#include<stdio.h>

void firstFit(int blockSize[], int m, int processSize[], int n) {
    int i, j;
    int allocation[n];

    for (i = 0; i < n; i++) {
        allocation[i] = -1;
    }

    for (i = 0; i < n; i++) {
        for (j = 0; j < m; j++) {
            if (blockSize[j] >= processSize[i]) {
                allocation[i] = j;
                blockSize[j] -= processSize[i];
                break;
            }
        }
    }

    printf("\nProcess No.\tProcess Size\tBlock no.\n");
    for (int i = 0; i < n; i++) {
        printf(" %i\t\t\t", i + 1);
        printf("%i\t\t\t", processSize[i]);
        if (allocation[i] != -1)
            printf("%i", allocation[i] + 1);
        else
            printf("Not Allocated");
        printf("\n");
    }
}

int main() {
    int m, n;
    printf("Enter the number of blocks: ");
    scanf("%d", &m);

    printf("Enter the number of processes: ");
    scanf("%d", &n);

    int blockSize[m];
    int processSize[n];

    printf("Enter the sizes of the blocks:\n");
    for (int i = 0; i < m; i++) {
        printf("Block %d: ", i + 1);
        scanf("%d", &blockSize[i]);
    }

    printf("Enter the sizes of the processes:\n");
    for (int i = 0; i < n; i++) {
        printf("Process %d: ", i + 1);
        scanf("%d", &processSize[i]);
    }

    firstFit(blockSize, m, processSize, n);

    return 0;
}
devanshtyagi@JARVIS:~/OS_Practicals$ gcc first_fit.c -o first_fit
devanshtyagi@JARVIS:~/OS_Practicals$ ./first_fit
Enter the number of blocks: 4
Enter the number of processes: 3
Enter the sizes of the blocks:
Block 1: 100
Block 2: 200
Block 3: 400
Block 4: 200
Enter the sizes of the processes:
Process 1: 324
Process 2: 213
Process 3: 123

Process No.     Process Size    Block no.
 1                       324                             3
 2                       213                             Not Allocated
 3                       123                             2
devanshtyagi@JARVIS:~/OS_Practicals$
```

```
devanshtyagi@JARVIS: ~/OS_
devanshtyagi@JARVIS:~/OS_Practicals$ cat best_fit.c
#include <stdio.h>

void bestFit(int blockSize[], int m, int processSize[], int n) {
    int allocation[n];

    for (int i = 0; i < n; i++)
        allocation[i] = -1;

    for (int i = 0; i < n; i++) {
        int bestIdx = -1;
        for (int j = 0; j < m; j++) {
            if (blockSize[j] >= processSize[i]) {
                if (bestIdx == -1 || blockSize[bestIdx] > blockSize[j])
                    bestIdx = j;
            }
        }

        if (bestIdx != -1) {
            allocation[i] = bestIdx;
            blockSize[bestIdx] -= processSize[i];
        }
    }

    printf("\nProcess No.\tProcess Size\tBlock no.\n");
    for (int i = 0; i < n; i++) {
        printf(" %d\t\t%d\t\t", i + 1, processSize[i]);
        if (allocation[i] != -1)
            printf("%d", allocation[i] + 1);
        else
            printf("Not Allocated");
        printf("\n");
    }
}

int main() {
    int m, n;

    printf("Enter the number of blocks: ");
    scanf("%d", &m);

    printf("Enter the number of processes: ");
    scanf("%d", &n);

    int blockSize[m];
    int processSize[n];

    printf("Enter the sizes of the blocks:\n");
    for (int i = 0; i < m; i++) {
        printf("Block %d: ", i + 1);
        scanf("%d", &blockSize[i]);
    }

    printf("Enter the sizes of the processes:\n");
    for (int i = 0; i < n; i++) {
        printf("Process %d: ", i + 1);
        scanf("%d", &processSize[i]);
    }

    bestFit(blockSize, m, processSize, n);

    return 0;
}
devanshtyagi@JARVIS:~/OS_Practicals$ gcc best_fit.c -o best_fit
devanshtyagi@JARVIS:~/OS_Practicals$ ./best_fit
Enter the number of blocks: 5
Enter the number of processes: 3
Enter the sizes of the blocks:
Block 1: 100
Block 2: 230
Block 3: 400
Block 4: 500
Block 5: 100
Enter the sizes of the processes:
Process 1: 231
Process 2: 234
Process 3: 436

Process No.     Process Size    Block no.
 1              231             3
 2              234             4
 3              436             Not Allocated
devanshtyagi@JARVIS:~/OS_Practicals$
```

devanshtyagi@JARVIS: ~/OS_

```
devanshtyagi@JARVIS:~/OS_Practicals$ cat worst_fit.c
#include <stdio.h>

void worstFit(int blockSize[], int m, int processSize[], int n) {
    int allocation[n];

    for (int i = 0; i < n; i++)
        allocation[i] = -1;

    for (int i = 0; i < n; i++) {
        int wstIdx = -1;
        for (int j = 0; j < m; j++) {
            if (blockSize[j] >= processSize[i]) {
                if (wstIdx == -1 || blockSize[wstIdx] < blockSize[j])
                    wstIdx = j;
            }
        }

        if (wstIdx != -1) {
            allocation[i] = wstIdx;
            blockSize[wstIdx] -= processSize[i];
        }
    }

    printf("\nProcess No.\tProcess Size\tBlock no.\n");
    for (int i = 0; i < n; i++) {
        printf(" %d\t\t%d\t\t", i + 1, processSize[i]);
        if (allocation[i] != -1)
            printf("%d", allocation[i] + 1);
        else
            printf("Not Allocated");
        printf("\n");
    }
}
```

```
int main() {
    int m, n;

    printf("Enter the number of blocks: ");
    scanf("%d", &m);

    printf("Enter the number of processes: ");
    scanf("%d", &n);

    int blockSize[m];
    int processSize[n];

    printf("Enter the sizes of the blocks:\n");
    for (int i = 0; i < m; i++) {
        printf("Block %d: ", i + 1);
        scanf("%d", &blockSize[i]);
    }

    printf("Enter the sizes of the processes:\n");
    for (int i = 0; i < n; i++) {
        printf("Process %d: ", i + 1);
        scanf("%d", &processSize[i]);
    }

    worstFit(blockSize, m, processSize, n);

    return 0;
}
devanshtyagi@JARVIS:~/OS_Practicals$ gcc worst_fit.c -o worst_fit
devanshtyagi@JARVIS:~/OS_Practicals$ ./worst_fit
Enter the number of blocks: 3
Enter the number of processes: 4
Enter the sizes of the blocks:
Block 1: 100
Block 2: 230
Block 3: 456
Enter the sizes of the processes:
Process 1: 123
Process 2: 355
Process 3: 245
Process 4: 547

Process No.     Process Size    Block no.
 1              123             3
 2              355             Not Allocated
 3              245             3
 4              547             Not Allocated
devanshtyagi@JARVIS:~/OS_Practicals$
```