

VIVEKANANDA INSTITUTE OF PROFESSIONAL STUDIES - TECHNICAL CAMPUS

Grade **A++** Accredited Institution by NAAC

NBA Accredited for MCA Programme; Recognized under Section 2(f) by UGC;
Affiliated to GGSIP University, Delhi; Recognized by Bar Council of India and AICTE

An ISO 9001:2015 Certified Institution

SCHOOL OF ENGINEERING & TECHNOLOGY

B. Tech Programme: AI-ML (A) (5th Semester)

Course Title: Operating Systems Lab

Course Code: AIML- 351

Submitted To:

Dr. Shivanka

Submitted By:

Name: Kunsh Sabharwal

Enrolment No: 01117711623



VIVEKANANDA INSTITUTE OF PROFESSIONAL STUDIES - TECHNICAL CAMPUS

Grade A++ Accredited Institution by NAAC

NBA Accredited for MCA Programme; Recognized under Section 2(f) by UGC;

Affiliated to GGSIP University, Delhi; Recognized by Bar Council of India and AICTE

An ISO 9001:2015 Certified Institution

SCHOOL OF ENGINEERING & TECHNOLOGY

VISION OF INSTITUTE

To be an educational institute that empowers the field of engineering to build a sustainable future by providing quality education with innovative practices that supports people, planet and profit.

MISSION OF INSTITUTE

To groom the future engineers by providing value-based education and awakening students' curiosity, nurturing creativity and building capabilities to enable them to make significant contributions to the world.



VIVEKANANDA INSTITUTE OF PROFESSIONAL STUDIES - TECHNICAL CAMPUS

Grade A++ Accredited Institution by NAAC

NBA Accredited for MCA Programme; Recognized under Section 2(f) by UGC;
Affiliated to GGSIP University, Delhi; Recognized by Bar Council of India and AICTE

An ISO 9001:2015 Certified Institution

SCHOOL OF ENGINEERING & TECHNOLOGY

INDEX

S. No.	Experiment	Date	Marks			Rem arks	Updated Marks	Faculty Signature
			Laboratory Assessment (15 Marks)	Class Participatio n (5 Marks)	Viva (5 Marks)			
1.	Install Ubuntu on windows and, compile and run first C program using gcc.							
2.	Open ubuntu terminal and write the command for following operations and share the output screen (a) Command to know your current working directory (b) List all the files in the current directory (c) List all the files in the order of their file size (d) List only directories in the current folder (e) List only files starting with “N” alphabet (f) Using help command find the help on ‘man’ command. (g) Display the content of manual pages on ‘ls’ command (h) Demonstrate the usage of “whatis” command. (i) Make directory named “OSLab/kunsh”. (j) Write command to reach to “kunsh” Directory. (k) Create a txt file named “TodaysMsg.txt” and write a greeting message in it. (l) Copy this file “TodaysMsg.txt” to OSLab directory (m) Delete the file “TodaysMsg.txt” from the ‘yourname’ Folder (n) Create a text file named “Hello.txt” and write a							

	suitable message in it. (o) Using touch command create files with names mon.txt, tues.txt, and wed.txt (p) Copy these newly created files to a folder named “dupfolder” after creating it. (q) Move Hello.txt to dupfolder (r) Count number of words in the Hello.txt file						
3.	Perform following shell script-based programs (a) Write a Shell Program to swap the two integers. (b) Create a shell script that checks if a specific directory exists. If it does, the script should back up all files from that directory into a specified backup directory. The script should then loop through the files in the backup directory and list all files that were successfully copied. If the directory does not exist, the script should print an error message. (c) Write a shell script to check if a given number is a prime number or not (d) Write a shell script to greet the user as per the time whenever he/ she opens terminal.						
4.	Write a c program to implement the following scheduling algorithms. (a) First come first serve (b) Round Robin Scheduling (c) Shortest job first (d) Shortest Job remaining first.						
5.	Implementation of the following Memory Allocation Methods for fixed partition (a) First Fit (b) Worst Fit (c) Best Fit.						
6.	Write a program to implement reader/writer problems using semaphore.						
7.	Write a program to implement Banker’s algorithm for deadlock avoidance.						

8.	<p>Process Management</p> <ul style="list-style-type: none">(a) fork()(b) execv()(c) execlp()(d) wait()(e) sleep() <p>(A) Program to implement the fork function using C. (B) Program to implement execv function using C. (C) Program to implement execlp function. (D) Program to implement wait function using C. (E) Program to implement sleep function using C</p>						
9.	Write a program to implement Inter Process Communication (IPC) using Message Queues.						
10.	Write a program to implement IPC using pipes.						
11.	Write a program using Pthread, where main thread calculates number of lines in a file and child calculates number of words.						

EXPERIMENT 1

Problem statement: Install Ubuntu on windows and, compile and run first C program using gcc.

Theory:

KUNSH SABHARWAL

Outputs: (a) Installation Screenshot

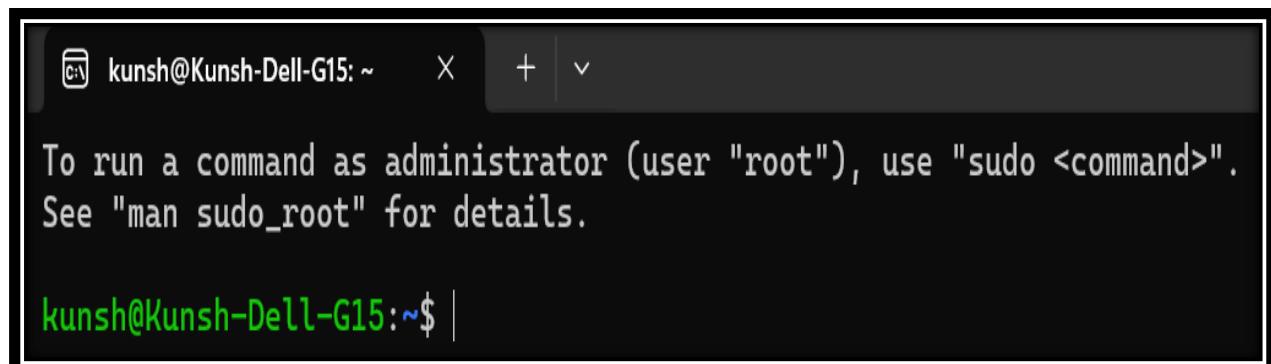
```
kunsh@Kunsh-Dell-G15:/mnt/c/WINDOWS/system32
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\WINDOWS\system32> wsl --install -d Ubuntu
>>
Downloading: Windows Subsystem for Linux 2.5.10
Installing: Windows Subsystem for Linux 2.5.10
Windows Subsystem for Linux 2.5.10 has been installed.
The operation completed successfully.
Downloading: Ubuntu
Installing: Ubuntu
Distribution successfully installed. It can be launched via 'wsl.exe -d Ubuntu'
Launching Ubuntu...
Provisioning the new WSL instance Ubuntu
This might take a while...
Create a default Unix user account: Kunsh
Invalid username. A valid username must start with a lowercase letter or underscore, and can contain lowercase letters, digits, underscores, and dashes.
Create a default Unix user account: kunsh
New password:
Retype new password:
passwd: password updated successfully
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

kunsh@Kunsh-Dell-G15:/mnt/c/WINDOWS/system32$
```

(b) Ubuntu Terminal first launch screenshot



A screenshot of a dark-themed Ubuntu terminal window. The title bar shows the session name 'kunsh@Kunsh-Dell-G15: ~'. The main area of the terminal displays the following text:

```
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.
```

The prompt at the bottom of the terminal is 'kunsh@Kunsh-Dell-G15:~\$ |'

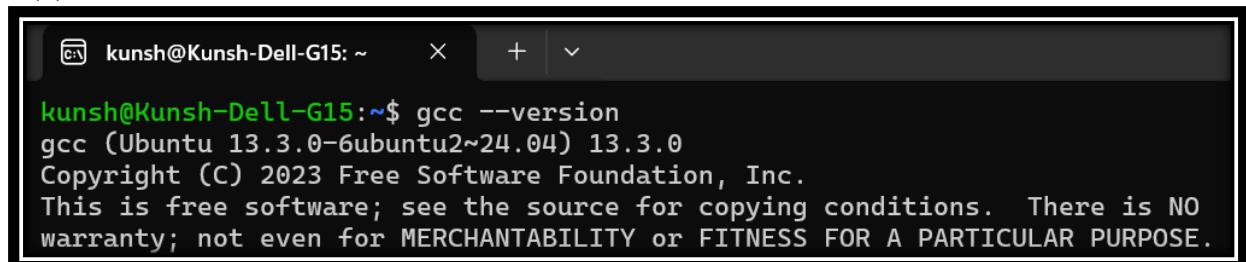
(c) Running Update commands for Ubuntu

```
kunsh@Kunsh-Dell-G15: ~      X  +  ▾

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

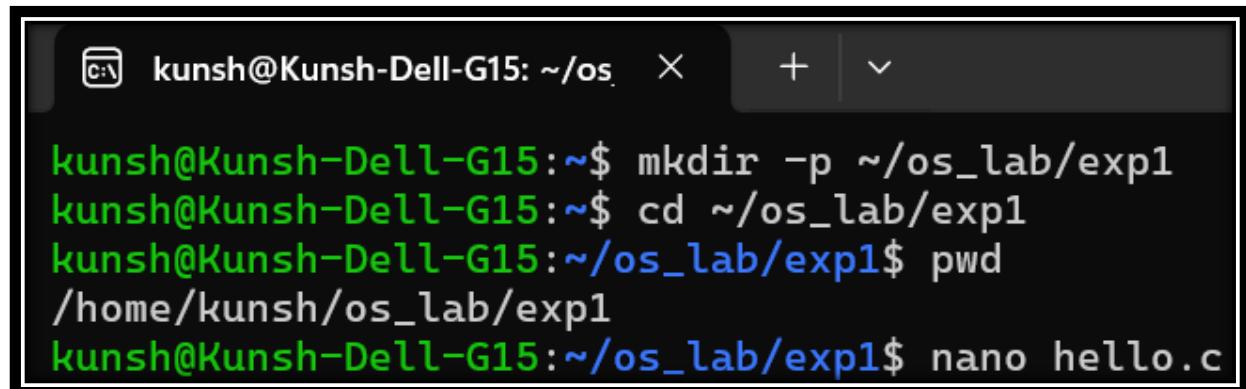
kunsh@Kunsh-Dell-G15:~$ sudo apt update && sudo apt upgrade -y
[sudo] password for kunsh:
Get:1 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]
Hit:2 http://archive.ubuntu.com/ubuntu noble InRelease
Get:3 http://archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Get:4 http://archive.ubuntu.com/ubuntu noble-backports InRelease [126 kB]
Get:5 http://security.ubuntu.com/ubuntu noble-security/main amd64 Components [21.6 kB]
Get:6 http://archive.ubuntu.com/ubuntu noble/universe amd64 Packages [15.0 MB]
Get:7 http://security.ubuntu.com/ubuntu noble-security/universe amd64 Packages [878 kB]
Get:8 http://security.ubuntu.com/ubuntu noble-security/universe Translation-en [194 kB]
Get:9 http://archive.ubuntu.com/ubuntu noble/universe Translation-en [5982 kB]
Get:10 http://security.ubuntu.com/ubuntu noble-security/universe amd64 Components [52.3 kB]
Get:11 http://security.ubuntu.com/ubuntu noble-security/universe amd64 c-n-f Metadata [17.0 kB]
Get:12 http://archive.ubuntu.com/ubuntu noble/universe amd64 Components [3871 kB]
Get:13 http://security.ubuntu.com/ubuntu noble-security/restricted amd64 Components [212 B]
Get:14 http://security.ubuntu.com/ubuntu noble-security/multiverse amd64 Packages [18.5 kB]
Get:15 http://security.ubuntu.com/ubuntu noble-security/multiverse Translation-en [4288 B]
Get:16 http://archive.ubuntu.com/ubuntu noble/universe amd64 c-n-f Metadata [301 kB]
Get:17 http://security.ubuntu.com/ubuntu noble-security/multiverse amd64 Components [212 B]
Get:18 http://archive.ubuntu.com/ubuntu noble/multiverse amd64 Packages [269 kB]
Get:19 http://security.ubuntu.com/ubuntu noble-security/multiverse amd64 c-n-f Metadata [380 B]
Get:20 http://archive.ubuntu.com/ubuntu noble/multiverse Translation-en [118 kB]
Get:21 http://archive.ubuntu.com/ubuntu noble/multiverse amd64 Components [35.0 kB]
Get:22 http://archive.ubuntu.com/ubuntu noble/multiverse amd64 c-n-f Metadata [8328 B]
Get:23 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 Packages [1313 kB]
Get:24 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 Components [164 kB]
Get:25 http://archive.ubuntu.com/ubuntu noble-updates/universe amd64 Packages [1120 kB]
Get:26 http://archive.ubuntu.com/ubuntu noble-updates/universe Translation-en [287 kB]
Get:27 http://archive.ubuntu.com/ubuntu noble-updates/universe amd64 Components [377 kB]
Get:28 http://archive.ubuntu.com/ubuntu noble-updates/universe amd64 c-n-f Metadata [26.0 kB]
Get:29 http://archive.ubuntu.com/ubuntu noble-updates/restricted amd64 Components [212 B]
Get:30 http://archive.ubuntu.com/ubuntu noble-updates/multiverse amd64 Packages [33.2 kB]
Get:31 http://archive.ubuntu.com/ubuntu noble-updates/multiverse Translation-en [6772 B]
Get:32 http://archive.ubuntu.com/ubuntu noble-updates/multiverse amd64 Components [940 B]
Get:33 http://archive.ubuntu.com/ubuntu noble-updates/multiverse amd64 c-n-f Metadata [592 B]
Get:34 http://archive.ubuntu.com/ubuntu noble-backports/main amd64 Packages [39.9 kB]
Get:35 http://archive.ubuntu.com/ubuntu noble-backports/main Translation-en [9152 B]
Get:36 http://archive.ubuntu.com/ubuntu noble-backports/main amd64 Components [7060 B]
Get:37 http://archive.ubuntu.com/ubuntu noble-backports/main amd64 c-n-f Metadata [272 B]
Get:38 http://archive.ubuntu.com/ubuntu noble-backports/universe amd64 Packages [28.9 kB]
Get:39 http://archive.ubuntu.com/ubuntu noble-backports/universe Translation-en [17.4 kB]
Get:40 http://archive.ubuntu.com/ubuntu noble-backports/universe amd64 Components [30.8 kB]
Get:41 http://archive.ubuntu.com/ubuntu noble-backports/universe amd64 c-n-f Metadata [1304 B]
Get:42 http://archive.ubuntu.com/ubuntu noble-backports/restricted amd64 Components [216 B]
Get:43 http://archive.ubuntu.com/ubuntu noble-backports/restricted amd64 c-n-f Metadata [116 B]
Get:44 http://archive.ubuntu.com/ubuntu noble-backports/multiverse amd64 Components [212 B]
Get:45 http://archive.ubuntu.com/ubuntu noble-backports/multiverse amd64 c-n-f Metadata [116 B]
Fetched 30.7 MB in 11s (2726 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
All packages are up to date.
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Calculating upgrade... Done
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
kunsh@Kunsh-Dell-G15:~$ |
```


(e) GCC version screenshot



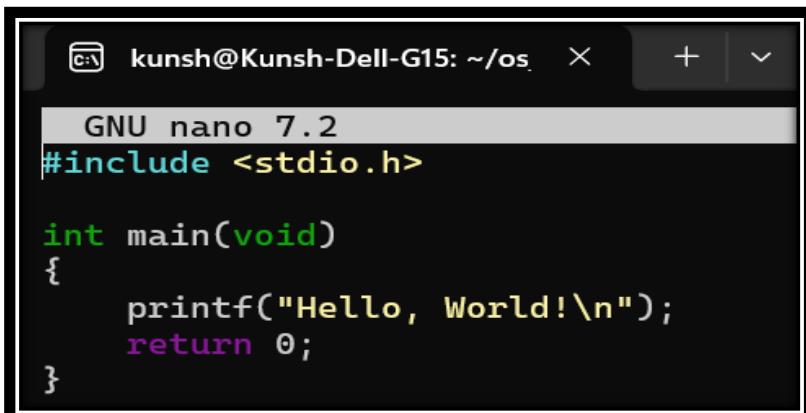
```
kunsh@Kunsh-Dell-G15:~$ gcc --version
gcc (Ubuntu 13.3.0-6ubuntu2~24.04) 13.3.0
Copyright (C) 2023 Free Software Foundation, Inc.
This is free software; see the source for copying conditions. There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

(f) Make new directory and go to it.



```
kunsh@Kunsh-Dell-G15:~/os$ mkdir -p ~/os_lab/exp1
kunsh@Kunsh-Dell-G15:~/os$ cd ~/os_lab/exp1
kunsh@Kunsh-Dell-G15:~/os_lab/exp1$ pwd
/home/kunsh/os_lab/exp1
kunsh@Kunsh-Dell-G15:~/os_lab/exp1$ nano hello.c
```

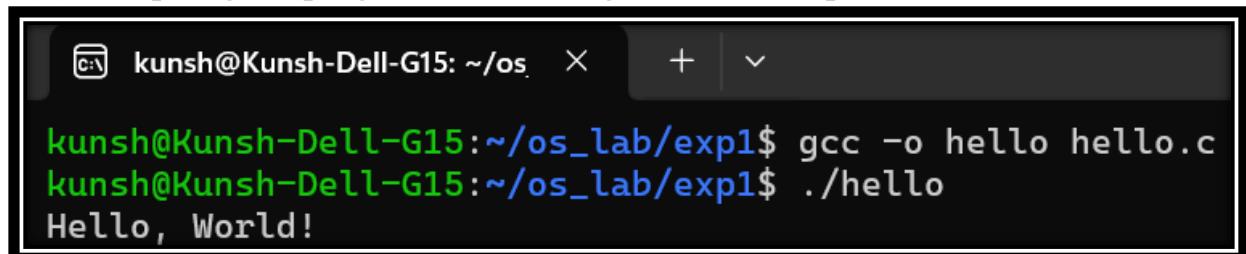
(g) Make a new file with name hello.c in nano text editor



```
GNU nano 7.2
#include <stdio.h>

int main(void)
{
    printf("Hello, World!\n");
    return 0;
}
```

(h) Compiling the program and running to see the output



```
kunsh@Kunsh-Dell-G15:~/os_lab/exp1$ gcc -o hello hello.c
kunsh@Kunsh-Dell-G15:~/os_lab/exp1$ ./hello
Hello, World!
```

Learning Outcome:

EXPERIMENT 2

Problem statement: Open ubuntu terminal and write the command for following operations and share the output screen.

Theory:

Outputs: (a) Command to know current working directory:

```
kunsh@Kunsh-Dell-G15: ~      X + | v
kunsh@Kunsh-Dell-G15:~$ pwd
/home/kunsh
```

(b) List all the files in the current directory:

```
kunsh@Kunsh-Dell-G15: ~      X + | v
kunsh@Kunsh-Dell-G15:~$ ls
os_lab
```

(c) List all files in order of their file size:

```
kunsh@Kunsh-Dell-G15: ~      X + | v
kunsh@Kunsh-Dell-G15:~$ ls -ls
total 4
drwxr-xr-x 3 kunsh kunsh 4096 Aug  8 05:41 os_lab
```

(d) List only directories in the current folder:

```
kunsh@Kunsh-Dell-G15: ~      X + | v
kunsh@Kunsh-Dell-G15:~$ ls -d */
os_lab/
```

(e) List only files starting with the alphabet “N”:

```
kunsh@Kunsh-Dell-G15: ~      X + | v
kunsh@Kunsh-Dell-G15:~$ ls N*
ls: cannot access 'N*': No such file or directory
```

(f) Using help command, find the help on ‘man’ command:

```
kunsh@Kunsh-Dell-G15:~$ man --help
Usage: man [OPTION...] [SECTION] PAGE...

-C, --config-file=FILE      use this user configuration file
-d, --debug                 emit debugging messages
-D, --default                reset all options to their default values
--warnings[=WARNINGS]       enable warnings from groff

Main modes of operation:
-f, --whatis                equivalent to whatis
-k, --apropos                equivalent to apropos
-K, --global-apropos         search for text in all pages
-l, --local-file              interpret PAGE argument(s) as local filename(s)
-w, --where, --path, --location
                           print physical location of man page(s)
-W, --where-cat, --location-cat
                           print physical location of cat file(s)

-c, --catman                 used by catman to reformat out of date cat pages
-R, --recode=ENCODING        output source page encoded in ENCODING

Finding manual pages:
-L, --locale=LOCALE          define the locale for this particular man search
-m, --systems=SYSTEM          use manual pages from other systems
-M, --manpath=PATH            set search path for manual pages to PATH

-S, -s, --sections=LIST      use colon separated section list
-e, --extension=EXTENSION    limit search to extension type EXTENSION

-i, --ignore-case             look for pages case-insensitively (default)
-I, --match-case              look for pages case-sensitively

--regex                      show all pages matching regex
--wildcard                   show all pages matching wildcard

--names-only                 make --regex and --wildcard match page names only,
                            not descriptions

-a, --all                     find all matching manual pages
-u, --update                  force a cache consistency check

--no-subpages                don't try subpages, e.g. 'man foo bar' => 'man
                            foo-bar'

Controlling formatted output:
-P, --pager=PAGER            use program PAGER to display output
-r, --prompt=STRING           provide the 'less' pager with a prompt

-7, --ascii                  display ASCII translation of certain latin1 chars
-E, --encoding=ENCODING      use selected output encoding
--no-hyphenation, --nh        turn off hyphenation
--no-justification,           --nj    turn off justification
-p, --preprocessor=STRING    STRING indicates which preprocessors to run:
                            e - [n]eqn, p - pic, t - tbl,
g - grap, r - refer, v - vgrind

-t, --troff                  use groff to format pages
-T, --troff-device[=DEVICE]   use groff with selected device

-H, --html[=BROWSER]          use www-browser or BROWSER to display HTML output
```

(g) Display the content of manual pages on ‘ls’ command:

```
kunsh@Kunsh-Dell-G15: ~ + v
LS(1)                                         User Commands

NAME
    ls - list directory contents

SYNOPSIS
    ls [OPTION]... [FILE]...

DESCRIPTION
    List information about the FILES (the current directory by default). Sort entries alphabetically if none of -cftuvSUX nor --sort is specified.

    Mandatory arguments to long options are mandatory for short options too.

-a, --all
        do not ignore entries starting with .

-A, --almost-all
        do not list implied . and ..

--author
        with -l, print the author of each file

-b, --escape
        print C-style escapes for nongraphic characters

--block-size=SIZE
        with -l, scale sizes by SIZE when printing them; e.g., '--block-size=M'; see SIZE format below

-B, --ignore-backups
        do not list implied entries ending with ~

-c      with -lt: sort by, and show, ctime (time of last change of file status information); with -l: show ctime and sort by name; otherwise: sort by ctime, newest first

-C      list entries by columns

--color[=WHEN]
        color the output WHEN; more info below

-d, --directory
        list directories themselves, not their contents

-D, --dired
        generate output designed for Emacs' dired mode

-f      list all entries in directory order

-F, --classify[=WHEN]
        append indicator (one of */>@|) to entries WHEN

--file-type
        likewise, except do not append '*'

--format=WORD
        across -x, commas -m, horizontal -x, long -l, single-column -1, verbose -l, vertical -C

--full-time
        like -l --time-style=full-iso

-g      like -l, but do not list owner

Manual page ls(1) line 1 (press h for help or q to quit)
```

(h) Demonstrate the usage of “whatis” command:

```
kunsh@Kunsh-Dell-G15:~$ whatis ls
ls (1)
- list directory contents
```

(i) Make directory named “OSLab/kunsh”.

```
kunsh@Kunsh-Dell-G15:~$ mkdir -p OSLab/kunsh
```

(j) Write command to reach to “kunsh” Directory:

```
kunsh@Kunsh-Dell-G15:~/OS$ cd OSLab/kunsh
kunsh@Kunsh-Dell-G15:~/OSLab/kunsh$
```

(k) Create a txt file named “TodaysMsg.txt” and write a greeting message in it:

```
kunsh@Kunsh-Dell-G15:~/OS$ echo "Hello, hope you have a great day!" > TodaysMsg.txt
```

(l) Copy this file “TodaysMsg.txt” to OSLab directory:

```
kunsh@Kunsh-Dell-G15:~/OS$ cp TodaysMsg.txt ../
```

(m) Delete the file “TodaysMsg.txt” from the ‘kunsh’ Folder:

```
kunsh@Kunsh-Dell-G15:~/OS$ rm TodaysMsg.txt
```

(n) Delete the directory ‘kunsh’:

```
kunsh@Kunsh-Dell-G15:~/OS$ cd ..
kunsh@Kunsh-Dell-G15:~/OSLab$ rmdir kunsh
```

(o) Create a text file named “Hello.txt” and write a suitable message in it:

```
kunsh@Kunsh-Dell-G15:~/OSLab$ echo "This is a message in the Hello.txt file." > Hello.txt
```

(p) Using touch command create files with names mon.txt, tues.txt, and wed.txt:

```
kunsh@Kunsh-Dell-G15:~/OSLab$ touch mon.txt tues.txt wed.txt
```

(q) Copy these newly created files to a folder named “dupfolder” after creating it:

```
kunsh@Kunsh-Dell-G15:~/OSLab$ mkdir dupfolder
kunsh@Kunsh-Dell-G15:~/OSLab$ cp mon.txt tues.txt wed.txt dupfolder/
```

(r) Move Hello.txt to dupfolder:

```
kunsh@Kunsh-Dell-G15:~/OSLab$ mv Hello.txt dupfolder/
```

(s) Count the number of words in the Hello.txt file:

```
kunsh@Kunsh-Dell-G15:~/OSLab$ wc -w dupfolder/Hello.txt
8 dupfolder/Hello.txt
```

(t) Create two files with identical content, change one alphabet in one of these and compare them using cmp command:

```
kunsh@Kunsh-Dell-G15:~/OSLab$ echo "This is the original content." > file1.txt
kunsh@Kunsh-Dell-G15:~/OSLab$ cp file1.txt file2.txt
kunsh@Kunsh-Dell-G15:~/OSLab$ echo "This is the changed content." > file2.txt
kunsh@Kunsh-Dell-G15:~/OSLab$ cmp file1.txt file2.txt
file1.txt file2.txt differ: byte 13, line 1
```

Learning Outcome:

EXPERIMENT 3

Problem statement: Perform following shell script-based programs:

- a. Write a Shell Program to swap the two integers.
- b. Create a shell script that checks if a specific directory exists. If it does, the script should back up all files from that directory into a specified backup directory. The script should then loop through the files in the backup directory and list all files that were successfully copied. If the directory does not exist, the script should print an error message.
- c. Write a shell script to check if a given number is a prime number or not
- d. Write a shell script to greet the user as per the time whenever he/she opens terminal.

Theory:

Outputs: (a) Shell Program to Swap Two Integers:

```
kunsh@Kunsh-Dell-G15:~/OSLab$#!/bin/bash
# Shell script to swap two integers

echo "Enter the first number:"
read a
echo "Enter the second number:"
read b

# Swapping the values
temp=$a
a=$b
b=$temp

echo "After swapping:"
echo "First number: $a"
echo "Second number: $b"
Enter the first number:
5
Enter the second number:
10
After swapping:
First number: 10
Second number: 5
```

(b) Shell Script to Back Up Files from a Directory:

```
kunsh@Kunsh-Dell-G15:~$ #!/bin/bash
# Shell script to back up files from a specific directory

# Define the directories (inside your home folder)
SOURCE_DIR="$HOME/OSLab"
BACKUP_DIR="$HOME/os_lab"

# Check if the source directory exists
if [ -d "$SOURCE_DIR" ]; then
    # Create the backup directory if it doesn't exist
    mkdir -p "$BACKUP_DIR"

    # Copy files to the backup directory
    cp -r "$SOURCE_DIR"/* "$BACKUP_DIR/"

    # List all files that were successfully copied
    echo "The following files were copied to $BACKUP_DIR:"
    for file in "$BACKUP_DIR"/*; do
        echo "$(basename "$file")"
    done
else
    # Print an error message if the directory doesn't exist
    echo "Error: Directory $SOURCE_DIR does not exist."
fi
The following files were copied to /home/kunsh/os_lab:
TodaysMsg.txt
dupfolder
expl
file1.txt
file2.txt
mon.txt
tues.txt
wed.txt
kunsh@Kunsh-Dell-G15:~$ #!/bin/bash
# Shell script to back up files from a specific directory

# Define the directories (inside your home folder)
SOURCE_DIR="$HOME/kunsh"
BACKUP_DIR="$HOME/os_lab"

# Check if the source directory exists
if [ -d "$SOURCE_DIR" ]; then
    # Create the backup directory if it doesn't exist
    mkdir -p "$BACKUP_DIR"

    # Copy files to the backup directory
    cp -r "$SOURCE_DIR"/* "$BACKUP_DIR/"

    # List all files that were successfully copied
    echo "The following files were copied to $BACKUP_DIR:"
    for file in "$BACKUP_DIR"/*; do
        echo "$(basename "$file")"
    done
else
    # Print an error message if the directory doesn't exist
    echo "Error: Directory $SOURCE_DIR does not exist."
fi
Error: Directory /home/kunsh/kunsh does not exist.
kunsh@Kunsh-Dell-G15:~$ |
```

(c) Shell Script to Check if a Number is Prime:

```
kunsh@Kunsh-Dell-G15: ~      X  +  ▾
kunsh@Kunsh-Dell-G15:~$ #!/bin/bash
# Shell script to check if a number is a prime number

echo "Enter a number:"
read num

if [ $num -le 1 ]; then
    echo "$num is not a prime number."
    exit 0
fi

is_prime=1
for ((i = 2; i <= num / 2; i++)); do
    if [ $((num % i)) -eq 0 ]; then
        is_prime=0
        break
    fi
done

if [ $is_prime -eq 1 ]; then
    echo "$num is a prime number."
else
    echo "$num is not a prime number."
fi
Enter a number:
5
5 is a prime number.
kunsh@Kunsh-Dell-G15:~$ #!/bin/bash
# Shell script to check if a number is a prime number

echo "Enter a number:"
read num

if [ $num -le 1 ]; then
    echo "$num is not a prime number."
    exit 0
fi

is_prime=1
for ((i = 2; i <= num / 2; i++)); do
    if [ $((num % i)) -eq 0 ]; then
        is_prime=0
        break
    fi
done

if [ $is_prime -eq 1 ]; then
    echo "$num is a prime number."
else
    echo "$num is not a prime number."
fi
Enter a number:
10
10 is not a prime number.
```

(d) Shell Script to Greet the User Based on the Time of Day:

```
kunsh@Kunsh-Dell-G15:~$#!/bin/bash
# Shell script to greet the user based on the time of day

hour=$(date +"%H")

if [ $hour -lt 12 ]; then
    echo "Good Morning!"
elif [ $hour -lt 18 ]; then
    echo "Good Afternoon!"
else
    echo "Good Evening!"
fi
Good Morning!
```

Learning Outcome:

EXPERIMENT 4

Problem statement: Write a c program to implement the following scheduling algorithms.

- (a) First come first serve
- (b) Round Robin Scheduling
- (c) Shortest job first
- (d) Shortest Job remaining first.

Theory:

Source Code: (a) First Come First Serve Scheduling Algorithm

```
C FirstComeFirstServe.c X
1 #include <stdio.h>
2
3 // updated code for FCFS scheduling algorithm - EXP 4
4 void findWaitingTime(int processes[], int n, int bt[], int wt[])
5 {
6     wt[0] = 0;
7     for (int i = 1; i < n; i++)
8     {
9         wt[i] = bt[i - 1] + wt[i - 1];
10    }
11 }
12
13 void findTurnAroundTime(int processes[], int n, int bt[], int wt[], int tat[])
14 {
15     for (int i = 0; i < n; i++)
16     {
17         tat[i] = bt[i] + wt[i];
18     }
19 }
20
21 void findAverageTime(int processes[], int n, int bt[])
22 {
23     int wt[n], tat[n];
24     findWaitingTime(processes, n, bt, wt);
25     findTurnAroundTime(processes, n, bt, wt, tat);
26
27     printf("Processes    Burst Time    Waiting Time    Turnaround Time\n");
28     int total_wt = 0, total_tat = 0;
29     for (int i = 0; i < n; i++)
30     {
31         total_wt += wt[i];
32         total_tat += tat[i];
33         printf("    %d ", (i + 1));
34         printf("        %d ", bt[i]);
35         printf("        %d ", wt[i]);
36         printf("        %d\n", tat[i]);
37     }
38
39     printf("Average waiting time = %.2f\n", (float)total_wt / (float)n);
40     printf("Average turnaround time = %.2f\n", (float)total_tat / (float)n);
41 }
```

```

43 int main()
44 {
45     int processes[] = {1, 2, 3};
46     int n = sizeof processes / sizeof processes[0];
47     int burst_time[] = {10, 5, 8};
48
49     findAverageTime(processes, n, burst_time);
50     return 0;
51 }

```

(b) Round Robin Scheduling Algorithm

```

RoundRobin.cpp ×
1 #include <stdio.h>
2
3 void findWaitingTime(int processes[], int n, int bt[], int wt[], int quantum)
4 {
5     int rem_bt[n];
6     for (int i = 0; i < n; i++)
7     {
8         rem_bt[i] = bt[i];
9     }
10
11    int t = 0;
12    while (1)
13    {
14        int done = 1;
15        for (int i = 0; i < n; i++)
16        {
17            if (rem_bt[i] > 0)
18            {
19                done = 0;
20                if (rem_bt[i] > quantum)
21                {
22                    t += quantum;
23                    rem_bt[i] -= quantum;
24                }
25                else
26                {
27                    t += rem_bt[i];
28                    wt[i] = t - bt[i];
29                    rem_bt[i] = 0;
30                }
31            }
32        }
33        if (done == 1)
34            break;
35    }
36 }

```

```
RoundRobin.cpp X

38 void findTurnAroundTime(int processes[], int n, int bt[], int wt[], int tat[])
39 {
40     for (int i = 0; i < n; i++)
41     {
42         tat[i] = bt[i] + wt[i];
43     }
44 }
45
46 void findAverageTime(int processes[], int n, int bt[], int quantum)
47 {
48     int wt[n], tat[n];
49     findWaitingTime(processes, n, bt, wt, quantum);
50     findTurnAroundTime(processes, n, bt, wt, tat);
51
52     printf("Processes    Burst Time    Waiting Time    Turnaround Time\n");
53     int total_wt = 0, total_tat = 0;
54     for (int i = 0; i < n; i++)
55     {
56         total_wt += wt[i];
57         total_tat += tat[i];
58         printf("%d ", (i + 1));
59         printf("%d ", bt[i]);
60         printf("%d ", wt[i]);
61         printf("%d\n", tat[i]);
62     }
63
64     printf("Average waiting time = %.2f\n", (float)total_wt / (float)n);
65     printf("Average turnaround time = %.2f\n", (float)total_tat / (float)n);
66 }
67
68 int main()
69 {
70     int processes[] = {1, 2, 3};
71     int n = sizeof processes / sizeof processes[0];
72     int burst_time[] = {10, 5, 8};
73     int quantum = 2;
74
75     findAverageTime(processes, n, burst_time, quantum);
76     return 0;
77 }
```

(c) Shortest Job First Scheduling Algorithm

```
G+ ShortestJobFirst.cpp X
1 #include <stdio.h>
2
3 void findWaitingTime(int processes[], int n, int bt[], int wt[])
4 {
5     int sorted_processes[n], sorted_bt[n], temp;
6
7     // Sorting burst times in ascending order using Bubble Sort
8     for (int i = 0; i < n; i++)
9     {
10         sorted_processes[i] = i;
11         sorted_bt[i] = bt[i];
12     }
13     for (int i = 0; i < n - 1; i++)
14     {
15         for (int j = 0; j < n - i - 1; j++)
16         {
17             if (sorted_bt[j] > sorted_bt[j + 1])
18             {
19                 temp = sorted_bt[j];
20                 sorted_bt[j] = sorted_bt[j + 1];
21                 sorted_bt[j + 1] = temp;
22                 temp = sorted_processes[j];
23                 sorted_processes[j] = sorted_processes[j + 1];
24                 sorted_processes[j + 1] = temp;
25             }
26         }
27     }
28
29     wt[0] = 0;
30     for (int i = 1; i < n; i++)
31     {
32         wt[i] = sorted_bt[i - 1] + wt[i - 1];
33     }
34 }
35
36 void findTurnAroundTime(int processes[], int n, int bt[], int wt[], int tat[])
37 {
38     for (int i = 0; i < n; i++)
39     {
40         tat[i] = bt[i] + wt[i];
41     }
42 }
```

```
ShortestJobFirst.cpp X

44 void findAverageTime(int processes[], int n, int bt[])
45 {
46     int wt[n], tat[n];
47     findWaitingTime(processes, n, bt, wt);
48     findTurnAroundTime(processes, n, bt, wt, tat);
49
50     printf("Processes    Burst Time    Waiting Time    Turnaround Time\n");
51     int total_wt = 0, total_tat = 0;
52     for (int i = 0; i < n; i++)
53     {
54         total_wt += wt[i];
55         total_tat += tat[i];
56         printf("    %d ", (i + 1));
57         printf("    %d ", bt[i]);
58         printf("    %d ", wt[i]);
59         printf("    %d\n", tat[i]);
60     }
61
62     printf("Average waiting time = %.2f\n", (float)total_wt / (float)n);
63     printf("Average turnaround time = %.2f\n", (float)total_tat / (float)n);
64 }
65
66 int main()
67 {
68     int processes[] = {1, 2, 3};
69     int n = sizeof processes / sizeof processes[0];
70     int burst_time[] = {10, 5, 8};
71
72     findAverageTime(processes, n, burst_time);
73     return 0;
74 }
```

(d) Shortest Job Remaining First Scheduling Algorithm

```

ShortestJobRemainingFirst.cpp ×

1  #include <stdio.h>
2  #include <limits.h>
3
4  void findWaitingTime(int processes[], int n, int bt[], int wt[])
5  {
6      int rt[n];
7      for (int i = 0; i < n; i++)
8          rt[i] = bt[i];
9
10     int complete = 0, t = 0, minm = INT_MAX;
11     int shortest = 0, finish_time;
12     int check = 0;
13
14     while (complete != n)
15     {
16         for (int j = 0; j < n; j++)
17         {
18             if ((rt[j] < minm) && (rt[j] > 0))
19             {
20                 minm = rt[j];
21                 shortest = j;
22                 check = 1;
23             }
24         }
25
26         if (check == 0)
27         {
28             t++;
29             continue;
30         }
31
32         rt[shortest]--;
33         minm = rt[shortest];
34         if (minm == 0)
35             minm = INT_MAX;
36
37         if (rt[shortest] == 0)
38         {
39             complete++;
40             check = 0;
41             finish_time = t + 1;
42             wt[shortest] = finish_time - bt[shortest];
43             if (wt[shortest] < 0)
44                 wt[shortest] = 0;
45         }
46         t++;
47     }
48 }
```

ShortestJobRemainingFirst.cpp X

```
50 void findTurnAroundTime(int processes[], int n, int bt[], int wt[], int tat[])
51 {
52     for (int i = 0; i < n; i++)
53         tat[i] = bt[i] + wt[i];
54 }
55
56 void findAverageTime(int processes[], int n, int bt[])
57 {
58     int wt[n], tat[n];
59     findWaitingTime(processes, n, bt, wt);
60     findTurnAroundTime(processes, n, bt, wt, tat);
61
62     printf("Processes    Burst Time    Waiting Time    Turnaround Time\n");
63     int total_wt = 0, total_tat = 0;
64     for (int i = 0; i < n; i++)
65     {
66         total_wt += wt[i];
67         total_tat += tat[i];
68         printf("    %d ", (i + 1));
69         printf("    %d ", bt[i]);
70         printf("    %d ", wt[i]);
71         printf("    %d\n", tat[i]);
72     }
73
74     printf("Average waiting time = %.2f\n", (float)total_wt / (float)n);
75     printf("Average turnaround time = %.2f\n", (float)total_tat / (float)n);
76 }
77
78 int main()
79 {
80     int processes[] = {1, 2, 3};
81     int n = sizeof processes / sizeof processes[0];
82     int burst_time[] = {6, 8, 7};
83
84     findAverageTime(processes, n, burst_time);
85     return 0;
86 }
```

Outputs: (a) First Come First Serve Scheduling Algorithm

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Code + v ⌂ ⌂ ... | ⌂ X

PS C:\Users\DeLL\OneDrive\Desktop\VIPS\3rd Year\Operating Systems\OS Practical File Codes> cd "c:\Users\DeLL\OneDrive\Desktop\VIPS\3rd Year\Operating Systems\OS Practical File Codes\" ; if ($?) { gc
c FirstComeFirstServe.c -o FirstComeFirstServe } ; if ($?) { .\FirstComeFirstServe }

Processes Burst Time Waiting Time Turnaround Time
1 10 0 10
2 5 10 15
3 8 15 23

Average waiting time = 8.33
Average turnaround time = 16.00
PS C:\Users\DeLL\OneDrive\Desktop\VIPS\3rd Year\Operating Systems\OS Practical File Codes>
```

(b) Round Robin Scheduling Algorithm

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Code + v ⌂ ⌂ ... | ⌂ X

PS C:\Users\DeLL\OneDrive\Desktop\VIPS\3rd Year\Operating Systems\OS Practical File Codes> cd "c:\Users\DeLL\OneDrive\Desktop\VIPS\3rd Year\Operating Systems\OS Practical File Codes\" ; if ($?) { g+
+ RoundRobin.cpp -o RoundRobin } ; if ($?) { .\RoundRobin }

Processes Burst Time Waiting Time Turnaround Time
1 10 13 23
2 5 10 15
3 8 13 21

Average waiting time = 12.00
Average turnaround time = 19.67
PS C:\Users\DeLL\OneDrive\Desktop\VIPS\3rd Year\Operating Systems\OS Practical File Codes>
```

(c) Shortest Job First Scheduling Algorithm

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Code + v ⌂ ⌂ ... | ⌂ X

PS C:\Users\DeLL\OneDrive\Desktop\VIPS\3rd Year\Operating Systems\OS Practical File Codes> cd "c:\Users\DeLL\OneDrive\Desktop\VIPS\3rd Year\Operating Systems\OS Practical File Codes\" ; if ($?) { gc
c ShortestJobFirst.c -o ShortestJobFirst } ; if ($?) { .\ShortestJobFirst }

Processes Burst Time Waiting Time Turnaround Time
1 10 0 10
2 5 5 10
3 8 13 21

Average waiting time = 6.00
Average turnaround time = 13.67
PS C:\Users\DeLL\OneDrive\Desktop\VIPS\3rd Year\Operating Systems\OS Practical File Codes>
```

(d) Shortest Job Remaining First Scheduling Algorithm

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Code + v ⌂ ⌂ ... | ⌂ X

PS C:\Users\DeLL\OneDrive\Desktop\VIPS\3rd Year\Operating Systems\OS Practical File Codes> cd "c:\Users\DeLL\OneDrive\Desktop\VIPS\3rd Year\Operating Systems\OS Practical File Codes\" ; if ($?) { gc
c ShortestJobRemainingFirst.c -o ShortestJobRemainingFirst } ; if ($?) { .\ShortestJobRemainingFirst }

Processes Burst Time Waiting Time Turnaround Time
1 6 0 6
2 8 13 21
3 7 6 13

Average waiting time = 6.33
Average turnaround time = 13.33
PS C:\Users\DeLL\OneDrive\Desktop\VIPS\3rd Year\Operating Systems\OS Practical File Codes>
```

Learning Outcome:

EXPERIMENT 5

Problem statement: Implementation of the following Memory Allocation Methods for fixed partition

- a) First Fit
- b) Worst Fit
- c) Best Fit.

Theory:

Source Code: (a) First Fit Memory Allocation

```
C FirstFitMemoryAllocation.c U ×

1 #include <stdio.h>
2 void firstFit(int blockSize[], int m, int processSize[], int n)
3 {
4     int allocation[n];
5     for (int i = 0; i < n; i++)
6     {
7         allocation[i] = -1; // Initially, no block is assigned
8     }
9
10    for (int i = 0; i < n; i++)
11    {
12        for (int j = 0; j < m; j++)
13        {
14            if (blockSize[j] >= processSize[i])
15            {
16                allocation[i] = j;
17                blockSize[j] -= processSize[i];
18                break;
19            }
20        }
21    }
22
23    printf("Process No.\tProcess Size\tBlock No.\n");
24    for (int i = 0; i < n; i++)
25    {
26        printf("%d\t%d\t", i + 1, processSize[i]);
27        if (allocation[i] != -1)
28            printf("%d\n", allocation[i] + 1);
29        else
30            printf("Not Allocated\n");
31    }
32 }
33
34 int main()
35 {
36     int blockSize[] = {100, 500, 200, 300, 600};
37     int processSize[] = {212, 417, 112, 426};
38     int m = sizeof(blockSize) / sizeof(blockSize[0]);
39     int n = sizeof(processSize) / sizeof(processSize[0]);
40
41     firstFit(blockSize, m, processSize, n);
42     return 0;
43 }
```

(b) Worst Fit Memory Allocation

```
C WorstFitMemoryAllocation.c X
1 #include <stdio.h>
2
3 void worstFit(int blockSize[], int m, int processSize[], int n)
4 {
5     int allocation[n];
6     for (int i = 0; i < n; i++)
7     {
8         allocation[i] = -1; // Initially, no block is assigned
9     }
10
11    for (int i = 0; i < n; i++)
12    {
13        int wstIdx = -1;
14        for (int j = 0; j < m; j++)
15        {
16            if (blockSize[j] >= processSize[i])
17            {
18                if (wstIdx == -1 || blockSize[j] > blockSize[wstIdx])
19                {
20                    wstIdx = j;
21                }
22            }
23        }
24
25        if (wstIdx != -1)
26        {
27            allocation[i] = wstIdx;
28            blockSize[wstIdx] -= processSize[i];
29        }
30    }
31
32    printf("Process No.\tProcess Size\tBlock No.\n");
33    for (int i = 0; i < n; i++)
34    {
35        printf("%d\t\t%d\t\t", i + 1, processSize[i]);
36        if (allocation[i] != -1)
37            printf("%d\n", allocation[i] + 1);
38        else
39            printf("Not Allocated\n");
40    }
41 }
42
43 int main()
44 {
45     int blockSize[] = {100, 500, 200, 300, 600};
46     int processSize[] = {212, 417, 112, 426};
47     int m = sizeof(blockSize) / sizeof(blockSize[0]);
48     int n = sizeof(processSize) / sizeof(processSize[0]);
49
50     worstFit(blockSize, m, processSize, n);
51     return 0;
52 }
```

(c) Best Fit Memory Allocation

```
C BestFitMemoryAllocation.c X
1 #include <stdio.h>
2
3 void bestFit(int blockSize[], int m, int processSize[], int n)
4 {
5     int allocation[n];
6     for (int i = 0; i < n; i++)
7     {
8         allocation[i] = -1; // Initially, no block is assigned
9     }
10
11    for (int i = 0; i < n; i++)
12    {
13        int bestIdx = -1;
14        for (int j = 0; j < m; j++)
15        {
16            if (blockSize[j] >= processSize[i])
17            {
18                if (bestIdx == -1 || blockSize[j] < blockSize[bestIdx])
19                {
20                    bestIdx = j;
21                }
22            }
23        }
24
25        if (bestIdx != -1)
26        {
27            allocation[i] = bestIdx;
28            blockSize[bestIdx] -= processSize[i];
29        }
30    }
31
32    printf("Process No.\tProcess Size\tBlock No.\n");
33    for (int i = 0; i < n; i++)
34    {
35        printf("%d\t%d\t", i + 1, processSize[i]);
36        if (allocation[i] != -1)
37            printf("%d\n", allocation[i] + 1);
38        else
39            printf("Not Allocated\n");
40    }
41 }
```

```
C BestFitMemoryAllocation.c U X
43 int main()
44 {
45     int blockSize[] = {100, 500, 200, 300, 600};
46     int processSize[] = {212, 417, 112, 426};
47     int m = sizeof(blockSize) / sizeof(blockSize[0]);
48     int n = sizeof(processSize) / sizeof(processSize[0]);
49
50     bestFit(blockSize, m, processSize, n);
51
52 }
```

Outputs: (a) First Fit Memory Allocation

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\DeLL\OneDrive\Desktop\VIPS\3rd Year\Operating Systems\OS Practical File Codes> cd "c:\Users\DeLL\OneDrive\Desktop\VIPS\3rd Year\Operating Systems\OS Practical File Codes"; if ($?) { gcc FirstFitMemoryAllocation } ; if ($?) { .\FirstFitMemoryAllocation }
Process No. Process Size Block No.
1          212        2
2          417        5
3          112        2
4          426    Not Allocated
PS C:\Users\DeLL\OneDrive\Desktop\VIPS\3rd Year\Operating Systems\OS Practical File Codes>
```

(b) Worst Fit Memory Allocation

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\DeLL\OneDrive\Desktop\VIPS\3rd Year\Operating Systems\OS Practical File Codes> cd "c:\Users\DeLL\OneDrive\Desktop\VIPS\3rd Year\Operating Systems\OS Practical File Codes"; if ($?) { gcc WorstFitMemoryAllocation } ; if ($?) { .\WorstFitMemoryAllocation }
Process No. Process Size Block No.
1          212        5
2          417        2
3          112        5
4          426    Not Allocated
PS C:\Users\DeLL\OneDrive\Desktop\VIPS\3rd Year\Operating Systems\OS Practical File Codes>
```

(c) Best Fit Memory Allocation

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\DeLL\OneDrive\Desktop\VIPS\3rd Year\Operating Systems\OS Practical File Codes> cd "c:\Users\DeLL\OneDrive\Desktop\VIPS\3rd Year\Operating Systems\OS Practical File Codes"; if ($?) { gcc BestFitMemoryAllocation.c -o BestfitMemoryAllocation } ; if ($?) { .\BestFitMemoryAllocation }
Process No. Process Size Block No.
1          212        4
2          417        2
3          112        3
4          426        5
PS C:\Users\DeLL\OneDrive\Desktop\VIPS\3rd Year\Operating Systems\OS Practical File Codes>
```

Learning Outcome:

EXPERIMENT 6

Problem statement: Write a program to implement reader/writer problems using semaphore.

Theory:

Source Code:

```
GNU nano 7.2
#include <stdio.h>
#include <pthread.h>
#include <semaphore.h>
#include <unistd.h>

sem_t rw_mutex; // Semaphore for writer access
sem_t mutex; // Semaphore for reader access
int read_count = 0;
int shared_data = 0; // Shared resource

void *reader(void *arg)
{
    int reader_id = *((int *)arg);
    while (1)
    {
        // Reader entry section
        sem_wait(&mutex);
        read_count++;
        if (read_count == 1)
        {
            sem_wait(&rw_mutex);
        }
        sem_post(&mutex);

        // Reading section
        printf("Reader %d: Read shared data = %d\n", reader_id, shared_data);
        sleep(1); // Simulate reading time

        // Reader exit section
        sem_wait(&mutex);
        read_count--;
        if (read_count == 0)
        {
            sem_post(&rw_mutex);
        }
        sem_post(&mutex);
        sleep(1); // Simulate delay before next reading
    }
    return NULL;
}
```

```
GNU nano 7.2
void *writer(void *arg)
{
    int writer_id = *((int *)arg);
    while (1)
    {
        // Writer entry section
        sem_wait(&rw_mutex);

        // Writing section
        shared_data++;
        printf("Writer %d: Wrote shared data = %d\n", writer_id, shared_data);
        sleep(1); // Simulate writing time

        // Writer exit section
        sem_post(&rw_mutex);
        sleep(2); // Simulate delay before next writing
    }
    return NULL;
}
```

```
GNU nano 7.2
    return NULL;
}

int main()
{
    pthread_t rtid[5], wtid[2];
    int reader_ids[5] = {1, 2, 3, 4, 5};
    int writer_ids[2] = {1, 2};

    // Initialize semaphores
    sem_init(&mutex, 0, 1);
    sem_init(&rw_mutex, 0, 1);

    // Create reader threads
    for (int i = 0; i < 5; i++)
    {
        pthread_create(&rtid[i], NULL, reader, (void *)&reader_ids[i]);
    }

    // Create writer threads
    for (int i = 0; i < 2; i++)
    {
        pthread_create(&wtid[i], NULL, writer, (void *)&writer_ids[i]);
    }

    // Join all threads
    for (int i = 0; i < 5; i++)
    {
        pthread_join(rtid[i], NULL);
    }
    for (int i = 0; i < 2; i++)
    {
        pthread_join(wtid[i], NULL);
    }

    // Destroy semaphores
    sem_destroy(&mutex);
    sem_destroy(&rw_mutex);

    return 0;
}
```

Output:

```
kunsh@Kunsh-Dell-G15:~$ cd /mnt/c/Users/Kunsh/Documents
kunsh@Kunsh-Dell-G15:/mnt/c/Users/Kunsh/Documents$ nano reader_writer.c
kunsh@Kunsh-Dell-G15:/mnt/c/Users/Kunsh/Documents$ gcc reader_writer.c -o reader_writer -lpthread
kunsh@Kunsh-Dell-G15:/mnt/c/Users/Kunsh/Documents$ ./reader_writer
Reader 1: Read shared data = 0
Reader 2: Read shared data = 0
Reader 4: Read shared data = 0
Reader 5: Read shared data = 0
Reader 3: Read shared data = 0
Writer 1: Wrote shared data = 1
Writer 2: Wrote shared data = 2
Reader 4: Read shared data = 2
Reader 1: Read shared data = 2
Reader 5: Read shared data = 2
Reader 3: Read shared data = 2
Reader 2: Read shared data = 2
Writer 1: Wrote shared data = 3
Writer 2: Wrote shared data = 4
Reader 5: Read shared data = 4
Reader 2: Read shared data = 4
Reader 1: Read shared data = 4
Reader 3: Read shared data = 4
Reader 4: Read shared data = 4
Writer 1: Wrote shared data = 5
Writer 2: Wrote shared data = 6
Reader 5: Read shared data = 6
Reader 2: Read shared data = 6
Reader 1: Read shared data = 6
Reader 3: Read shared data = 6
Reader 4: Read shared data = 6
Writer 1: Wrote shared data = 7
Writer 2: Wrote shared data = 8
Reader 5: Read shared data = 8
Reader 2: Read shared data = 8
Reader 3: Read shared data = 8
Reader 4: Read shared data = 8
Reader 1: Read shared data = 8
Writer 1: Wrote shared data = 9
Writer 2: Wrote shared data = 10
Reader 2: Read shared data = 10
Reader 5: Read shared data = 10
Reader 4: Read shared data = 10
Reader 3: Read shared data = 10
Reader 1: Read shared data = 10
Writer 1: Wrote shared data = 11
Writer 2: Wrote shared data = 12
```

Learning Outcome:

EXPERIMENT 7

Problem statement: Write a program to implement Banker's algorithm for deadlock avoidance.

Theory:

Source Code:

```
C BankersAlgorithm.c X

1 #include <stdio.h>
2 #include <stdbool.h>
3
4 #define P 5
5 #define R 3
6
7 // updated and modified version
8
9 bool isSafe(int processes[], int n, int avail[], int max[][R], int allot[][][R])
10 {
11     int work[R];
12     bool finish[n];
13     for (int i = 0; i < R; i++)
14         work[i] = avail[i];
15     for (int i = 0; i < n; i++)
16         finish[i] = false;
17     int safeSeq[n];
18     int count = 0;
19     while (count < n)
20     {
21         bool found = false;
22         for (int p = 0; p < n; p++)
23         {
24             if (finish[p] == false)
25             {
26                 int j;
27                 for (j = 0; j < R; j++)
28                     if (max[p][j] - allot[p][j] > work[j])
29                         break;
30                 if (j == R)
31                 {
32                     for (int k = 0; k < R; k++)
33                         work[k] += allot[p][k];
34                     safeSeq[count++] = p;
35                     finish[p] = true;
36                 }
37             }
38         }
39     }
40 }
```

```
C BankersAlgorithm.c X

36         found = true;
37     }
38 }
39 if (found == false)
40 {
41     printf("System is not in a safe state\n");
42     return false;
43 }
44 }
45 }
46 printf("System is in a safe state.\nSafe sequence is: ");
47 for (int i = 0; i < n; i++)
48     printf("P%d ", safeSeq[i]);
49 printf("\n");
50 return true;
51 }

52
53 bool requestResources(int processes[], int n, int avail[], int max[][R], int allot[][][R], int req[])
54 {
55     int need[n][R];
56     for (int i = 0; i < n; i++)
57         for (int j = 0; j < R; j++)
58             need[i][j] = max[i][j] - allot[i][j];
59     for (int j = 0; j < R; j++)
60         if (req[j] > need[processes[0]][j])
61             return false;
62     for (int j = 0; j < R; j++)
63         if (req[j] > avail[j])
64             return false;
65     int old_avail[R];
66     int old_allot[n][R];
67     for (int j = 0; j < R; j++)
68     {
69         old_avail[j] = avail[j];
70         avail[j] -= req[j];
```

C BankersAlgorithm.c X

```
71     }
72     for (int i = 0; i < n; i++)
73         for (int j = 0; j < R; j++)
74             old_allot[i][j] = allot[i][j];
75     for (int j = 0; j < R; j++)
76         allot[processes[0]][j] += req[j];
77     if (isSafe(processes, n, avail, max, allot))
78     {
79         return true;
80     }
81     else
82     {
83         for (int j = 0; j < R; j++)
84         {
85             avail[j] = old_avail[j];
86             allot[processes[0]][j] = old_allot[processes[0]][j];
87         }
88         return false;
89     }
90 }
91
92 int main()
93 {
94     int processes[] = {0, 1, 2, 3, 4};
95     int n = sizeof(processes) / sizeof(processes[0]);
96     int resources[] = {10, 5, 7};
97     int m = sizeof(resources) / sizeof(resources[0]);
98     int avail[] = {3, 3, 3};
99     int max[][][R] = {
100         {7, 5, 3},
101         {3, 2, 2},
102         {9, 0, 2},
103         {2, 2, 2},
104         {4, 3, 3}};
```

```
C BankersAlgorithm.c X
105     int allot[][][R] = {
106         {0, 1, 0},
107         {2, 0, 0},
108         {3, 0, 2},
109         {2, 1, 1},
110         {0, 0, 2}};
111     int request[] = {1, 0, 2};
112     if (requestResources(processes, n, avail, max, allot, request))
113     {
114         printf("Request can be granted safely.\n");
115     }
116     else
117     {
118         printf("Request cannot be granted safely.\n");
119     }
120     return 0;
121 }
122
```

Output:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Code + v ... [ ] X
PS C:\Users\DeLL\OneDrive\Desktop\VIPS\3rd Year\Operating Systems\OS Practical File Codes> cd "c:/Users/DeLL/OneDrive/Desktop/VIPS/3rd Year/Operating Systems/OS Practical File Codes\" ; if ($?) { gcc BankersAlgorithm.c -o BankersAlgorithm } ; if ($?) { ./BankersAlgorithm }
System is in a safe state.
Safe sequence is: P3 P4 P1 P2 P0
Request can be granted safely.
PS C:\Users\DeLL\OneDrive\Desktop\VIPS\3rd Year\Operating Systems\OS Practical File Codes>
```

Learning Outcome:

EXPERIMENT 8

Problem statement: Perform following related to process management: -

- Program to implement the fork function using C.
- Program to implement execv function using C.
- Program to implement execlp function.
- Program to implement wait function using C.
- Program to implement sleep function using C.

Theory:

Source Code:

(a) Program to implement the fork function using C:

```
GNU nano 7.2
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>

int main() {
    pid_t pid = fork(); // Create a new process

    if (pid < 0) {
        // Fork failed
        printf("Fork failed.\n");
        return 1;
    } else if (pid == 0) {
        // Child process
        printf("Child process: PID = %d, Parent PID = %d\n", getpid(), getppid());
    } else {
        // Parent process
        printf("Parent process: PID = %d, Child PID = %d\n", getpid(), pid);
    }

    return 0;
}
```

(b) Program to implement execv function using C:

```
GNU nano 7.2
#include <stdio.h>
#include <unistd.h>

int main() {
    char *args[] = {"ls", "-l", NULL}; // Arguments for execv

    printf("Executing 'ls -l' using execv...\n");
    execv("/bin/ls", args); // Replace the current process with 'ls -l'

    // This line will not be executed if execv is successful
    perror("execv failed");
    return 1;
}
```

(c) Program to implement execlp function:

```
GNU nano 7.2
#include <stdio.h>
#include <unistd.h>

int main() {
    printf("Executing 'ls -l' using execlp...\n");
    execlp("ls", "ls", "-l", NULL); // Replace the current process with 'ls -l'

    // This line will not be executed if execlp is successful
    perror("execlp failed");
    return 1;
}
```

(d) Program to implement wait function using C:

```
GNU nano 7.2
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>

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

    if (pid < 0) {
        // Fork failed
        printf("Fork failed.\n");
        return 1;
    } else if (pid == 0) {
        // Child process
        printf("Child process: PID = %d\n", getpid());
        sleep(2); // Simulate work
        printf("Child process exiting...\n");
        return 0;
    } else {
        // Parent process
        int status;
        waitpid(pid, &status, 0); // Wait for the child process to exit
        if (WIFEXITED(status)) {
            printf("Parent process: Child exited with status %d\n", WEXITSTATUS(status));
        }
    }

    return 0;
}
```

(e) Program to implement sleep function using C:

```
GNU nano 7.2
#include <stdio.h>
#include <unistd.h>

int main() {
    printf("Sleeping for 5 seconds...\n");
    sleep(5); // Sleep for 5 seconds
    printf("Woke up after 5 seconds.\n");

    return 0;
}
```

Output:

(a) Program to implement the fork function using C:

```
kunsh@Kunsh-Dell-G15:~$ gcc fork.c -o fork
kunsh@Kunsh-Dell-G15:~$ ./fork
Parent process: PID = 583, Child PID = 584
Child process: PID = 584, Parent PID = 299
```

(b) Program to implement execv function using C:

```
kunsh@Kunsh-Dell-G15:~$ gcc execv.c -o execv
kunsh@Kunsh-Dell-G15:~$ ./execv
Executing 'ls -l' using execv...
total 60
drwxr-xr-x 3 kunsh kunsh 4096 Aug  8 06:51 OSLab
-rw-r--r-- 1 kunsh kunsh   292 Oct 12 12:24 execlp.c
-rwxr-xr-x 1 kunsh kunsh 16096 Oct 12 12:28 execv
-rw-r--r-- 1 kunsh kunsh   344 Oct 12 12:23 execv.c
-rwrxr-xr-x 1 kunsh kunsh 16128 Oct 12 12:27 fork
-rw-r--r-- 1 kunsh kunsh   493 Oct 12 12:21 fork.c
drwxr-xr-x 4 kunsh kunsh 4096 Aug  8 07:01 os_lab
-rw-r--r-- 1 kunsh kunsh   192 Oct 12 12:26 sleep.c
-rw-r--r-- 1 kunsh kunsh   710 Oct 12 12:25 wait.c
```

(c) Program to implement execlp function:

```
kunsh@Kunsh-Dell-G15:~$ gcc execlp.c -o execlp
kunsh@Kunsh-Dell-G15:~$ ./execlp
Executing 'ls -l' using execlp...
total 76
drwxr-xr-x 3 kunsh kunsh 4096 Aug  8 06:51 OSLab
-rw xr-xr-x 1 kunsh kunsh 16048 Oct 12 12:29 execlp
-rw-r--r-- 1 kunsh kunsh 292 Oct 12 12:24 execlp.c
-rw xr-xr-x 1 kunsh kunsh 16096 Oct 12 12:28 execv
-rw-r--r-- 1 kunsh kunsh 344 Oct 12 12:23 execv.c
-rw xr-xr-x 1 kunsh kunsh 16128 Oct 12 12:27 fork
-rw-r--r-- 1 kunsh kunsh 493 Oct 12 12:21 fork.c
drwxr-xr-x 4 kunsh kunsh 4096 Aug  8 07:01 os_lab
-rw-r--r-- 1 kunsh kunsh 192 Oct 12 12:26 sleep.c
-rw-r--r-- 1 kunsh kunsh 710 Oct 12 12:25 wait.c
```

(d) Program to implement wait function using C:

```
kunsh@Kunsh-Dell-G15:~$ gcc wait.c -o wait
kunsh@Kunsh-Dell-G15:~$ ./wait
Child process: PID = 605
Child process exiting...
Parent process: Child exited with status 0
```

(e) Program to implement sleep function using C:

```
kunsh@Kunsh-Dell-G15:~$ gcc sleep.c -o sleep
kunsh@Kunsh-Dell-G15:~$ ./sleep
Sleeping for 5 seconds...
Woke up after 5 seconds.
```

Learning Outcome:

EXPERIMENT 9

Problem statement: Write a program to implement Inter Process Communication (IPC) using Message Queues.

Theory:

Source Code:

(a) Sender End Code:

```
GNU nano 7.2
#include <stdio.h>
#include <stdlib.h>
#include <sys/ipc.h>
#include <sys/msg.h>
#include <unistd.h>
#include <string.h>

#define MSGKEY 1234
#define MSGSIZE 256

struct msg_buffer {
    long msg_type;
    char msg_text[MSGSIZE];
};

int main() {
    int msgid;
    struct msg_buffer message;

    // Create message queue
    msgid = msgget(MSGKEY, 0666 | IPC_CREAT);
    if (msgid < 0) {
        perror("msgget failed");
        exit(EXIT_FAILURE);
    }

    // Prepare message
    message.msg_type = 1; // Message type
    strcpy(message.msg_text, "Hello from sender!");

    // Send message
    if (msgsnd(msgid, &message, sizeof(message.msg_text), 0) < 0) {
        perror("msgsnd failed");
        exit(EXIT_FAILURE);
    }

    printf("Message sent: %s\n", message.msg_text);

    return 0;
}
```

(b) Receiver End Code:

```
GNU nano 7.2
#include <stdio.h>
#include <stdlib.h>
#include <sys/ipc.h>
#include <sys/msg.h>
#include <unistd.h>

#define MSGKEY 1234
#define MSGSIZE 256

struct msg_buffer {
    long msg_type;
    char msg_text[MSGSIZE];
};

int main() {
    int msgid;
    struct msg_buffer message;

    // Access message queue
    msgid = msgget(MSGKEY, 0666 | IPC_CREAT);
    if (msgid < 0) {
        perror("msgget failed");
        exit(EXIT_FAILURE);
    }

    // Receive message
    if (msgrcv(msgid, &message, sizeof(message.msg_text), 1, 0) < 0) {
        perror("msgrcv failed");
        exit(EXIT_FAILURE);
    }

    printf("Message received: %s\n", message.msg_text);

    // Remove message queue
    if (msgctl(msgid, IPC_RMID, NULL) < 0) {
        perror("msgctl failed");
        exit(EXIT_FAILURE);
    }

    return 0;
}
```

Output:

```
kunsh@Kunsh-Dell-G15:~$ gcc sender.c -o sender
kunsh@Kunsh-Dell-G15:~$ ./sender
Message sent: Hello from sender!
kunsh@Kunsh-Dell-G15:~$ gcc receiver.c -o receiver
kunsh@Kunsh-Dell-G15:~$ ./receiver
Message received: Hello from sender!
```

Learning Outcome:

EXPERIMENT 10

Problem statement: Write a program to implement IPC using pipes.

Theory:

Source Code:

```
GNU nano 7.2
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/wait.h>
#include <string.h>

int main() {
    int pipefd[2]; // File descriptors for the pipe
    pid_t pid;
    char buffer[100];

    // Create the pipe
    if (pipe(pipefd) == -1) {
        perror("pipe failed");
        exit(EXIT_FAILURE);
    }

    pid = fork(); // Create a child process

    if (pid < 0) {
        // Fork failed
        perror("fork failed");
        exit(EXIT_FAILURE);
    }

    if (pid == 0) {
        // Child process: Read from the pipe
        close(pipefd[1]); // Close unused write end
        read(pipefd[0], buffer, sizeof(buffer));
        printf("Child process received: %s\n", buffer);
        close(pipefd[0]); // Close read end
        exit(EXIT_SUCCESS);
    } else {
        // Parent process: Write to the pipe
        close(pipefd[0]); // Close unused read end
        const char *message = "Hello from parent process!";
        write(pipefd[1], message, strlen(message) + 1);
        close(pipefd[1]); // Close write end
        wait(NULL); // Wait for the child process to finish
        exit(EXIT_SUCCESS);
    }
}
```

Output:

```
kunsh@Kunsh-Dell-G15:~$ gcc pipes.c -o pipes
kunsh@Kunsh-Dell-G15:~$ ./pipes
Child process received: Hello from parent process!
```

Learning Outcome:

EXPERIMENT 11

Problem statement: Write a program using Pthread, where main thread calculates number of lines in a file and child calculates number of words.

Theory:

Source Code:

```
GNU nano 7.2
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <string.h>

#define MAX_LINE_LENGTH 1024

// Global variable to hold the file name
const char *filename = "IPC_pipes.txt";

// Function prototypes
void *count_lines(void *arg);
void *count_words(void *arg);

// Global variables to store results
int line_count = 0;
int word_count = 0;

int main() {
    pthread_t line_thread, word_thread;

    // Create threads
    if (pthread_create(&line_thread, NULL, count_lines, NULL) != 0) {
        perror("Failed to create line thread");
        exit(EXIT_FAILURE);
    }
    if (pthread_create(&word_thread, NULL, count_words, NULL) != 0) {
        perror("Failed to create word thread");
        exit(EXIT_FAILURE);
    }

    // Wait for threads to finish
    pthread_join(line_thread, NULL);
    pthread_join(word_thread, NULL);

    // Output results
    printf("Number of lines: %d\n", line_count);
    printf("Number of words: %d\n", word_count);

    return 0;
}
```

```
kunsh@Kunsh-Dell-G15: ~      + | v
GNU nano 7.2

    return 0;
}

// Function to count lines
void *count_lines(void *arg) {
    FILE *file = fopen(filename, "r");
    if (file == NULL) {
        perror("Failed to open file");
        pthread_exit(NULL);
    }

    char line[MAX_LINE_LENGTH];
    while (fgets(line, sizeof(line), file)) {
        line_count++;
    }

    fclose(file);
    pthread_exit(NULL);
}

// Function to count words
void *count_words(void *arg) {
    FILE *file = fopen(filename, "r");
    if (file == NULL) {
        perror("Failed to open file");
        pthread_exit(NULL);
    }

    char line[MAX_LINE_LENGTH];
    while (fgets(line, sizeof(line), file)) {
        char *token = strtok(line, "\t\n");
        while (token != NULL) {
            word_count++;
            token = strtok(NULL, "\t\n");
        }
    }

    fclose(file);
    pthread_exit(NULL);
}
```

Output:

```
kunsh@Kunsh-Dell-G15:~$ nano word_count.c
kunsh@Kunsh-Dell-G15:~$ echo "This is a test file." > IPC_pipes.txt
kunsh@Kunsh-Dell-G15:~$ echo "It has multiple lines." >> IPC_pipes.txt
kunsh@Kunsh-Dell-G15:~$ echo "And several words per line." >> IPC_pipes.txt
kunsh@Kunsh-Dell-G15:~$ gcc word_count.c -o word_count -pthread
kunsh@Kunsh-Dell-G15:~$ ./word_count
Number of lines: 3
Number of words: 14
```

Learning Outcome: