योगः कर्मसु कौशलम्
IN PURSUIT OF PERFECTION

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

VIPS
योगः कर्मसु कौशलम्
IN PURSUIT OF PERFECTION
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.

KUNSH SABHARWAL

VIPS
योगः कर्मसु कौशलम्
IN PURSUIT OF PERFECTION

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			Remarks	Updated Marks	Faculty Signature
			Laboratory Assessment (15 Marks)	Class Participation (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. (k) Copy this file "TodaysMsg.txt" to OSLab directory (l) Delete the file "TodaysMsg.txt" from the 'yourname' Folder (m) Delete the directory 'yourname' (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.	Process Management (a) fork() (b) execv() (c) execlp() (d) wait() (e) sleep() (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							
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.							

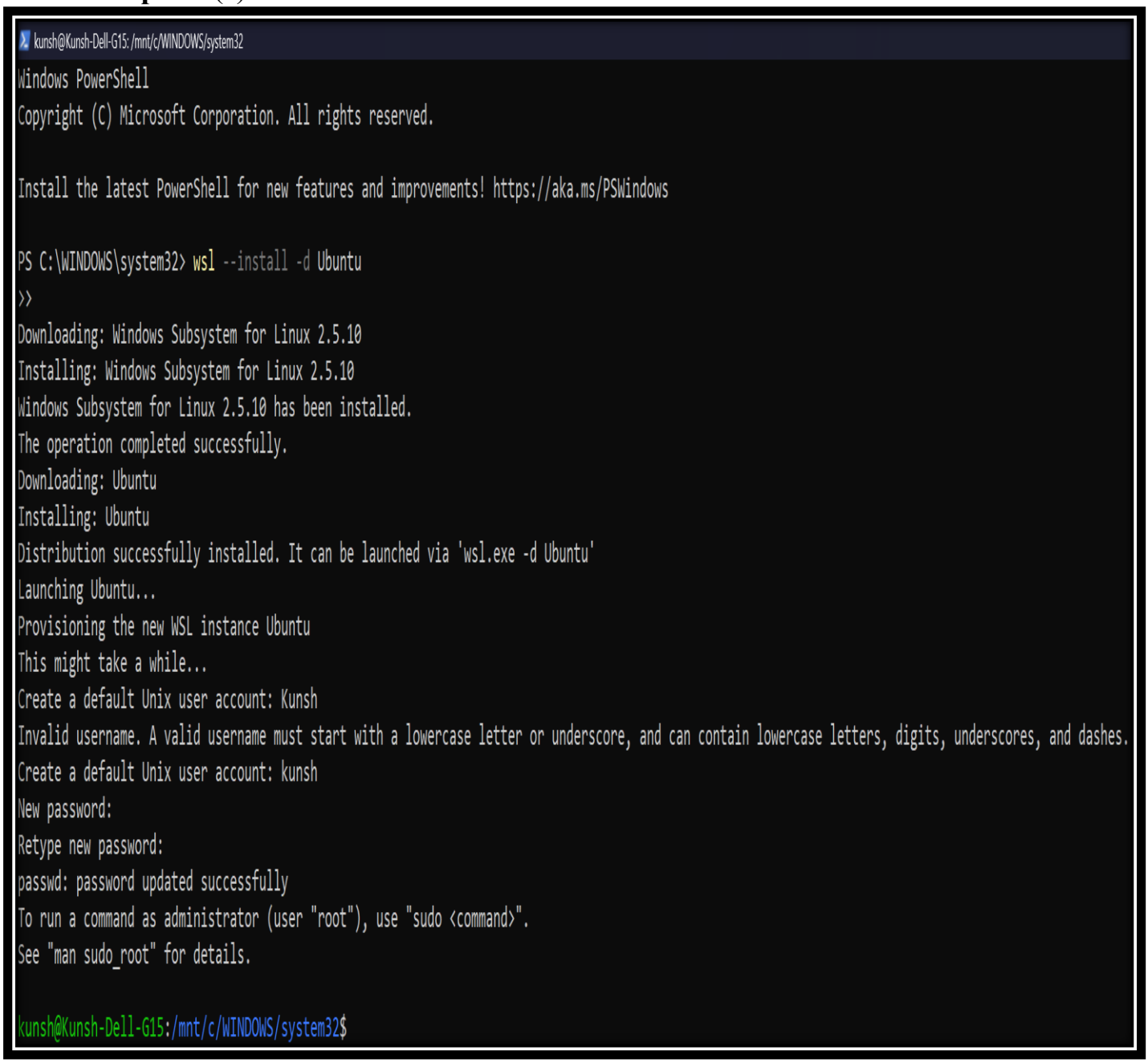
EXPERIMENT 1

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

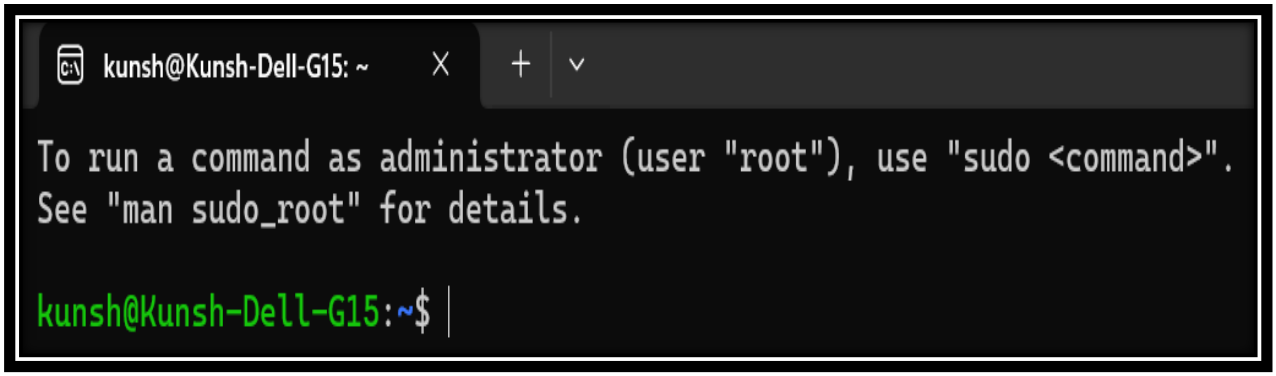
Theory:

KUNSH SABHARWAL

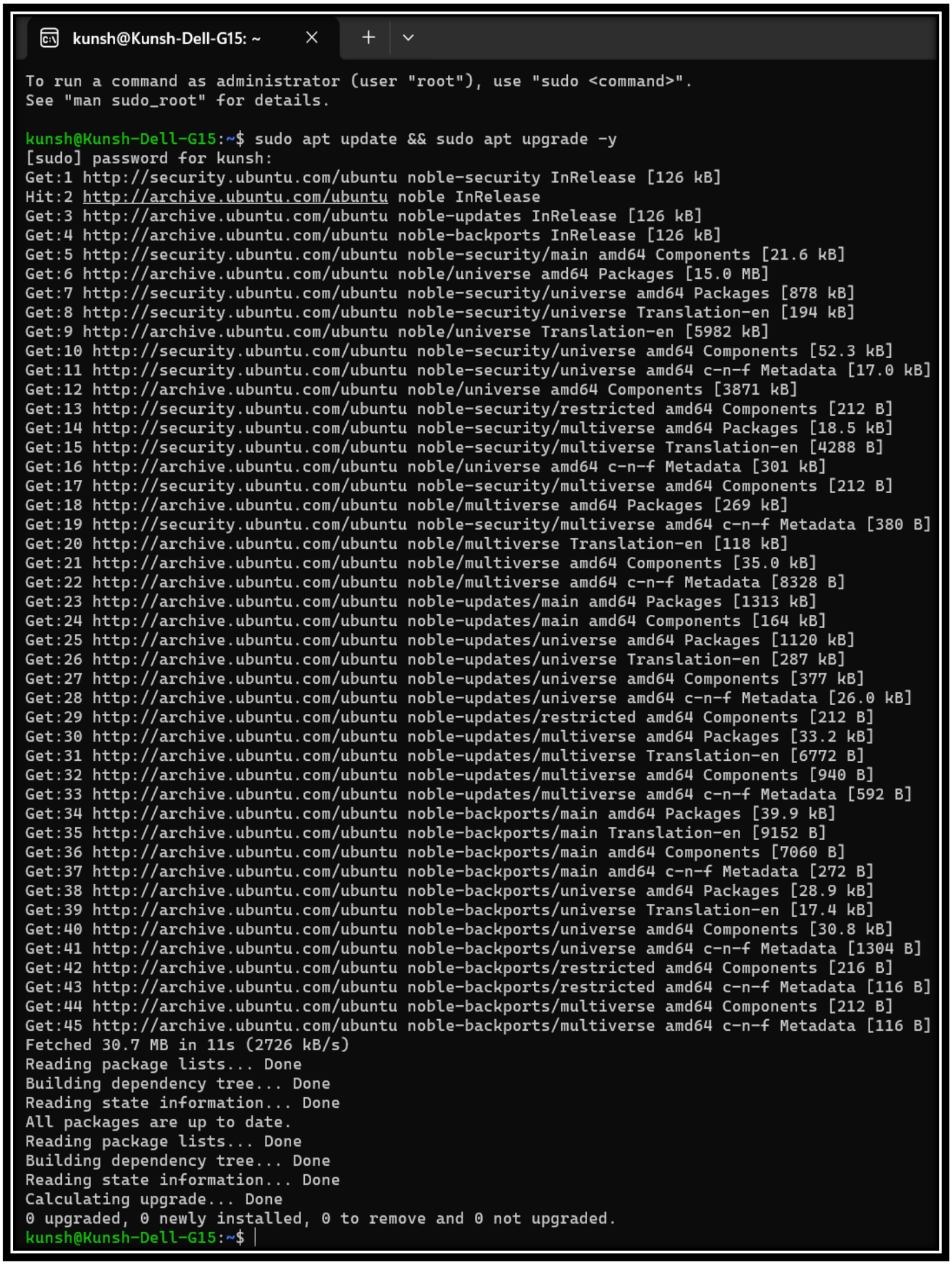
Outputs: (a) Installation Screenshot



(b) Ubuntu Terminal first launch screenshot



(c) Running Update commands for Ubuntu

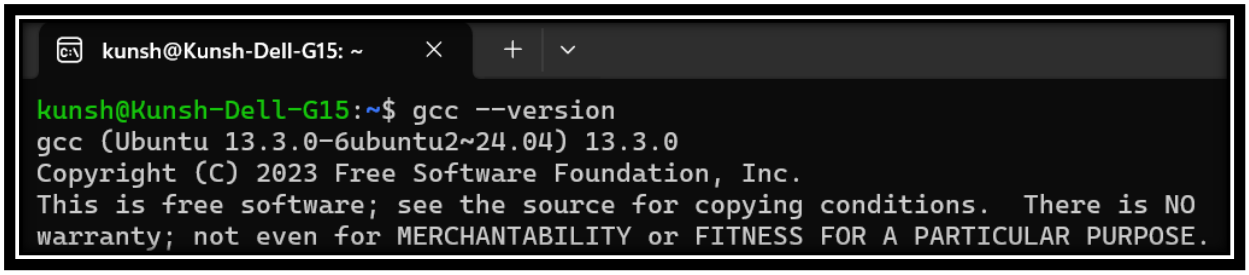


(d) Installing gnu and C Compiler

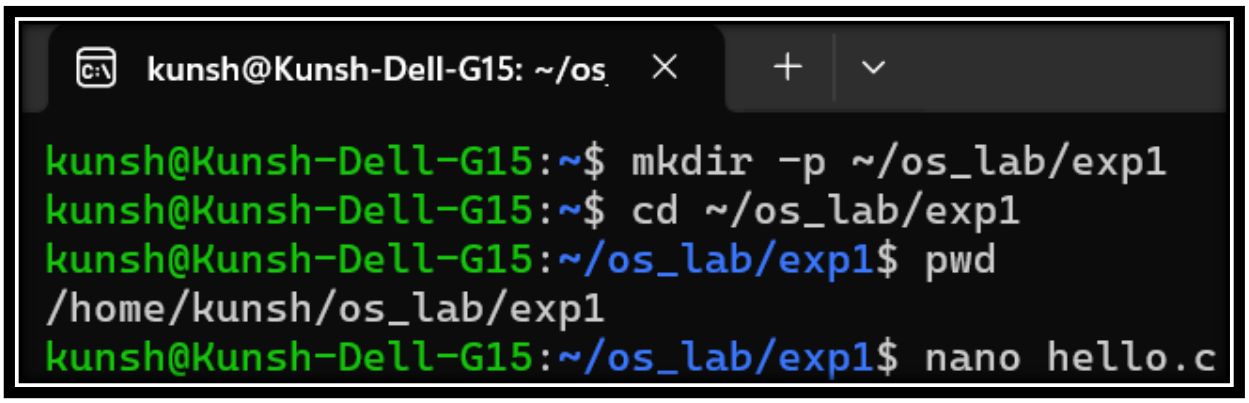
```
kunsh@Kunsh-Dell-G15: ~
kunsh@Kunsh-Dell-G15:~$ sudo apt install build-essential -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  bzip2 cpp cpp-13 cpp-13-x86-64-linux-gnu cpp-x86-64-linux-gnu dpkg-dev fakeroot g++ g++-13 g++-13-x86-64-linux-gnu g++-x86-64-linux-gnu gcc gcc-13 gcc-13-base gcc-13-x86-64-linux-gnu
  gcc-x86-64-linux-gnu libalgorithm-diff-perl libalgorithm-diff-xs-perl libalgorithm-merge-perl libaom3 libasan8 libatomic1 libc-dev-bin libc-devtools libc6-dev libcc1-0 libcrypt-dev
  libde265-0 libdpkg-perl libfakeroot libfile-fcntllock-perl libgcc-13-dev libgd3 libgomp1 libheif-plugin-aomdec libheif-plugin-aomenc libheif-plugin-libde265 libheif1 libhwasan0 libisl23
  libitm1 liblsan0 libmpc3 libquadmath0 libstdc++-13-dev libtsan2 libubsan1 libxpm4 linux-libc-dev lto-disabled-list make manpages-dev rpcsvc-proto
Suggested packages:
  bzip2-doc cpp-doc gcc-13-locales cpp-13-doc debian-keyring g++-multilib g++-13-multilib gcc-13-doc gcc-multilib autoconf automake libtool flex bison gdb gcc-doc gcc-13-multilib
  gdb-x86-64-linux-gnu glibc-doc bzr libgd-tools libheif-plugin-x265 libheif-plugin-ffmpegdec libheif-plugin-jpegdec libheif-plugin-jpegenc libheif-plugin-j2kdec libheif-plugin-j2kenc
  libheif-plugin-rav1e libheif-plugin-svtenc libstdc++-13-doc make-doc
The following NEW packages will be installed:
  build-essential bzip2 cpp cpp-13 cpp-13-x86-64-linux-gnu cpp-x86-64-linux-gnu dpkg-dev fakeroot g++ g++-13 g++-13-x86-64-linux-gnu g++-x86-64-linux-gnu gcc gcc-13 gcc-13-base
  gcc-13-x86-64-linux-gnu gcc-x86-64-linux-gnu libalgorithm-diff-perl libalgorithm-diff-xs-perl libalgorithm-merge-perl libaom3 libasan8 libatomic1 libc-dev-bin libc-devtools libc6-dev
  libcc1-0 libcrypt-dev libde265-0 libdpkg-perl libfakeroot libfile-fcntllock-perl libgcc-13-dev libgd3 libgomp1 libheif-plugin-aomdec libheif-plugin-aomenc libheif-plugin-libde265 libheif1
  libhwasan0 libisl23 libitm1 liblsan0 libmpc3 libquadmath0 libstdc++-13-dev libtsan2 libubsan1 libxpm4 linux-libc-dev lto-disabled-list make manpages-dev rpcsvc-proto
0 upgraded, 54 newly installed, 0 to remove and 0 not upgraded.
Need to get 71.4 MB of archives.
After this operation, 242 MB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 libc-dev-bin amd64 2.39-0ubuntu8.5 [20.4 kB]
Get:2 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 linux-libc-dev amd64 6.8.0-71.71 [1897 kB]
Get:3 http://archive.ubuntu.com/ubuntu noble/main amd64 libcrypt-dev amd64 1:4.4.36-4build1 [112 kB]
Get:4 http://archive.ubuntu.com/ubuntu noble/main amd64 rpcsvc-proto amd64 1.4.2-0ubuntu7 [67.4 kB]
Get:5 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 libc6-dev amd64 2.39-0ubuntu8.5 [2123 kB]
Get:6 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 gcc-13-base amd64 13.3.0-6ubuntu2~24.04 [51.5 kB]
Get:7 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 libisl23 amd64 0.26-3build1.1 [680 kB]
Get:8 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 libmpc3 amd64 1.3.1-1build1.1 [54.6 kB]
Get:9 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 cpp-13-x86-64-linux-gnu amd64 13.3.0-6ubuntu2~24.04 [10.7 MB]
Get:10 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 cpp-13 amd64 13.3.0-6ubuntu2~24.04 [1038 B]
Get:11 http://archive.ubuntu.com/ubuntu noble/main amd64 cpp-x86-64-linux-gnu amd64 4:13.2.0-7ubuntu1 [5326 B]
Get:12 http://archive.ubuntu.com/ubuntu noble/main amd64 cpp amd64 4:13.2.0-7ubuntu1 [22.4 kB]
Get:13 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 libcc1-0 amd64 14.2.0-4ubuntu2~24.04 [48.0 kB]
Get:14 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 libgomp1 amd64 14.2.0-4ubuntu2~24.04 [148 kB]
Get:15 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 libitm1 amd64 14.2.0-4ubuntu2~24.04 [29.7 kB]
Get:16 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 libatomic1 amd64 14.2.0-4ubuntu2~24.04 [10.5 kB]
Get:17 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 libasan8 amd64 14.2.0-4ubuntu2~24.04 [3031 kB]
Get:18 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 liblsan0 amd64 14.2.0-4ubuntu2~24.04 [1322 kB]
Get:19 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 libtsan2 amd64 14.2.0-4ubuntu2~24.04 [2772 kB]
Get:20 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 libubsan1 amd64 14.2.0-4ubuntu2~24.04 [1184 kB]
Get:21 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 libhwasan0 amd64 14.2.0-4ubuntu2~24.04 [1641 kB]
Get:22 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 libquadmath0 amd64 14.2.0-4ubuntu2~24.04 [153 kB]
Get:23 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 libgcc-13-dev amd64 13.3.0-6ubuntu2~24.04 [2681 kB]
Get:24 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 gcc-13-x86-64-linux-gnu amd64 13.3.0-6ubuntu2~24.04 [21.1 MB]
Get:25 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 gcc-13 amd64 13.3.0-6ubuntu2~24.04 [494 kB]
Get:26 http://archive.ubuntu.com/ubuntu noble/main amd64 gcc-x86-64-linux-gnu amd64 4:13.2.0-7ubuntu1 [1212 B]
Get:27 http://archive.ubuntu.com/ubuntu noble/main amd64 gcc amd64 4:13.2.0-7ubuntu1 [5018 B]
Get:28 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 libstdc++-13-dev amd64 13.3.0-6ubuntu2~24.04 [2420 kB]
```

```
kunsh@Kunsh-Dell-G15: ~
Get:29 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 g++-13-x86-64-linux-gnu amd64 13.3.0-6ubuntu2~24.04 [12.2 MB]
Get:30 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 g++-13 amd64 13.3.0-6ubuntu2~24.04 [16.1 kB]
Get:31 http://archive.ubuntu.com/ubuntu noble/main amd64 g++-x86-64-linux-gnu amd64 4:13.2.0-7ubuntu1 [964 B]
Get:32 http://archive.ubuntu.com/ubuntu noble/main amd64 g++ amd64 4:13.2.0-7ubuntu1 [1100 B]
Get:33 http://archive.ubuntu.com/ubuntu noble/main amd64 make amd64 4.3-4.1build2 [180 kB]
Get:34 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 libdpkg-perl all 1.22.6ubuntu6.1 [269 kB]
Get:35 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 bzip2 amd64 1.0.8-5.1build0.1 [34.5 kB]
Get:36 http://archive.ubuntu.com/ubuntu noble/main amd64 lto-disabled-list all 47 [12.4 kB]
Get:37 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 dpkg-dev all 1.22.6ubuntu6.1 [1074 kB]
Get:38 http://archive.ubuntu.com/ubuntu noble/main amd64 build-essential amd64 12.10ubuntu1 [4928 B]
Get:39 http://archive.ubuntu.com/ubuntu noble/main amd64 libfakeroot amd64 1.33-1 [32.4 kB]
Get:40 http://archive.ubuntu.com/ubuntu noble/main amd64 fakeroot amd64 1.33-1 [67.2 kB]
Get:41 http://archive.ubuntu.com/ubuntu noble/main amd64 libalgorithm-diff-perl all 1.201-1 [41.8 kB]
Get:42 http://archive.ubuntu.com/ubuntu noble/main amd64 libalgorithm-diff-xs-perl amd64 0.04-8build3 [11.2 kB]
Get:43 http://archive.ubuntu.com/ubuntu noble/main amd64 libalgorithm-merge-perl all 0.08-5 [11.4 kB]
Get:44 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 libaom3 amd64 3.8.2-2ubuntu0.1 [1941 kB]
Get:45 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 libheif-plugin-aomdec amd64 1.17.6-1ubuntu4.1 [10.4 kB]
Get:46 http://archive.ubuntu.com/ubuntu noble/main amd64 libde265-0 amd64 1.0.15-1build3 [166 kB]
Get:47 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 libheif-plugin-libde265 amd64 1.17.6-1ubuntu4.1 [8176 B]
Get:48 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 libheif1 amd64 1.17.6-1ubuntu4.1 [275 kB]
Get:49 http://archive.ubuntu.com/ubuntu noble/main amd64 libxpm4 amd64 1:3.5.17-1build2 [36.5 kB]
Get:50 http://archive.ubuntu.com/ubuntu noble/main amd64 libgd3 amd64 2.3.3-9ubuntu5 [128 kB]
Get:51 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 libc-devtools amd64 2.39-0ubuntu8.5 [29.3 kB]
Get:52 http://archive.ubuntu.com/ubuntu noble/main amd64 libfile-fcntllock-perl amd64 0.22-4ubuntu5 [30.7 kB]
Get:53 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 libheif-plugin-aomenc amd64 1.17.6-1ubuntu4.1 [14.7 kB]
Get:54 http://archive.ubuntu.com/ubuntu noble/main amd64 manpages-dev all 6.7-2 [2013 kB]
Fetched 71.4 MB in 21s (3395 kB/s)
Extracting templates from packages: 100%
Selecting previously unselected package libc-dev-bin.
(Reading database ... 40754 files and directories currently installed.)
Preparing to unpack .../00-libc-dev-bin_2.39-0ubuntu8.5_amd64.deb ...
Unpacking libc-dev-bin (2.39-0ubuntu8.5) ...
Selecting previously unselected package linux-libc-dev:amd64.
Preparing to unpack .../01-linux-libc-dev_6.8.0-71.71_amd64.deb ...
Unpacking linux-libc-dev:amd64 (6.8.0-71.71) ...
Selecting previously unselected package libcrypt-dev:amd64.
Preparing to unpack .../02-libcrypt-dev_1%3a4.4.36-4build1_amd64.deb ...
Unpacking libcrypt-dev:amd64 (1:4.4.36-4build1) ...
Selecting previously unselected package rpcsvc-proto.
Preparing to unpack .../03-rpcsvc-proto_1.4.2-0ubuntu7_amd64.deb ...
Unpacking rpcsvc-proto (1.4.2-0ubuntu7) ...
Selecting previously unselected package libc6-dev:amd64.
Preparing to unpack .../04-libc6-dev_2.39-0ubuntu8.5_amd64.deb ...
Unpacking libc6-dev:amd64 (2.39-0ubuntu8.5) ...
Selecting previously unselected package gcc-13-base:amd64.
Preparing to unpack .../05-gcc-13-base_13.3.0-6ubuntu2~24.04_amd64.deb ...
Unpacking gcc-13-base:amd64 (13.3.0-6ubuntu2~24.04) ...
Selecting previously unselected package libisl23:amd64.
Preparing to unpack .../06-libisl23_0.26-3build1.1_amd64.deb ...
Unpacking libisl23:amd64 (0.26-3build1.1) ...
Selecting previously unselected package libmpc3:amd64.
Preparing to unpack .../07-libmpc3_1.3.1-1build1.1_amd64.deb ...
Unpacking libmpc3:amd64 (1.3.1-1build1.1) ...
Selecting previously unselected package cpp-13-x86-64-linux-gnu.
Preparing to unpack .../08-cpp-13-x86-64-linux-gnu_13.3.0-6ubuntu2~24.04_amd64.deb ...
Unpacking cpp-13-x86-64-linux-gnu (13.3.0-6ubuntu2~24.04) ...
Selecting previously unselected package cpp-13.
Preparing to unpack .../09-cpp-13_13.3.0-6ubuntu2~24.04_amd64.deb ...
Unpacking cpp-13 (13.3.0-6ubuntu2~24.04) ...
Selecting previously unselected package cpp-x86-64-linux-gnu.
Preparing to unpack .../10-cpp-x86-64-linux-gnu_4%3a13.2.0-7ubuntu1_amd64.deb ...
```

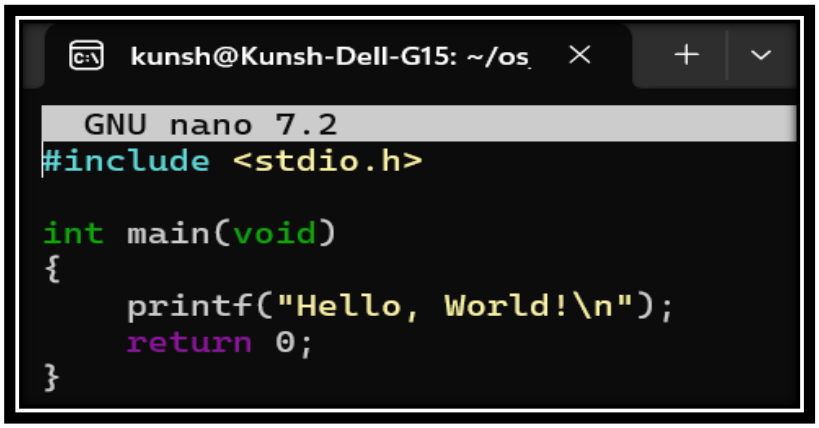
(e) GCC version screenshot



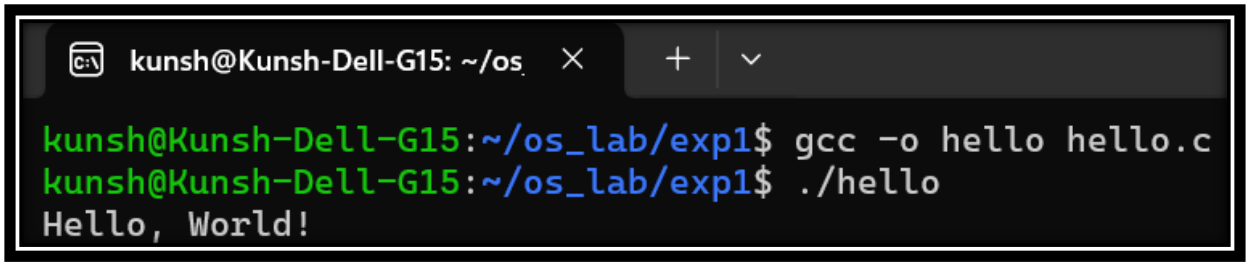
(f) Make new directory and go to it.



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



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



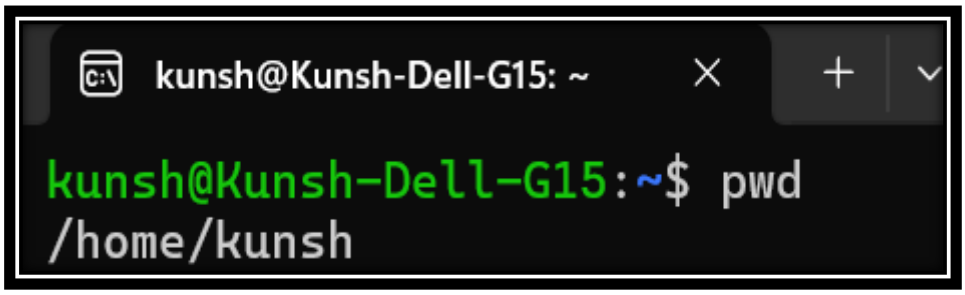
Learning Outcome:

EXPERIMENT 2

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

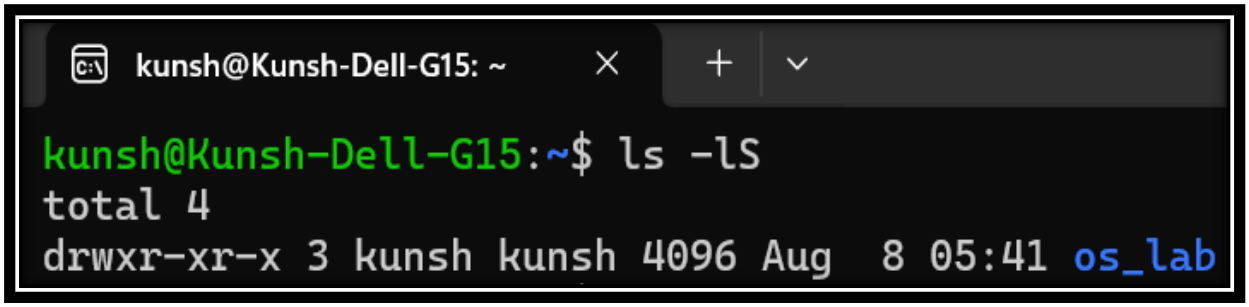
Theory:

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



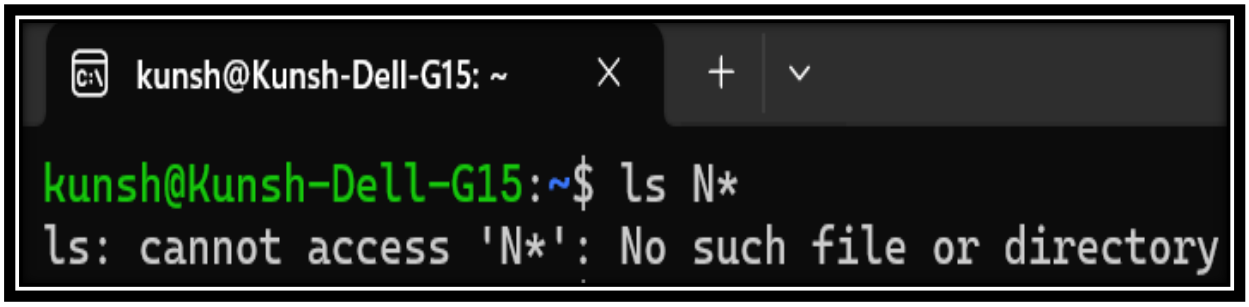
(b) List all the files in the current directory:

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



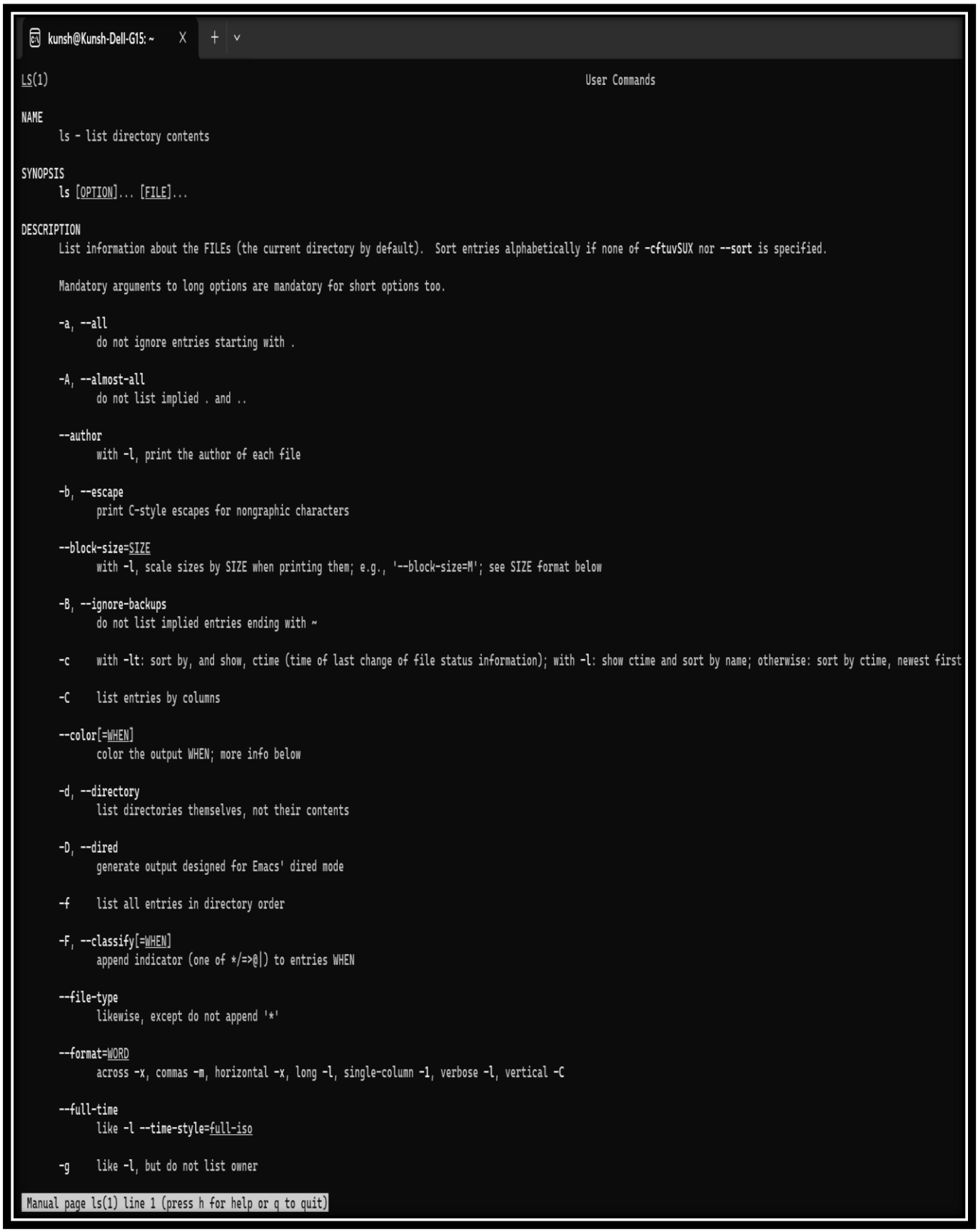
(d) List only directories in the current folder:

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

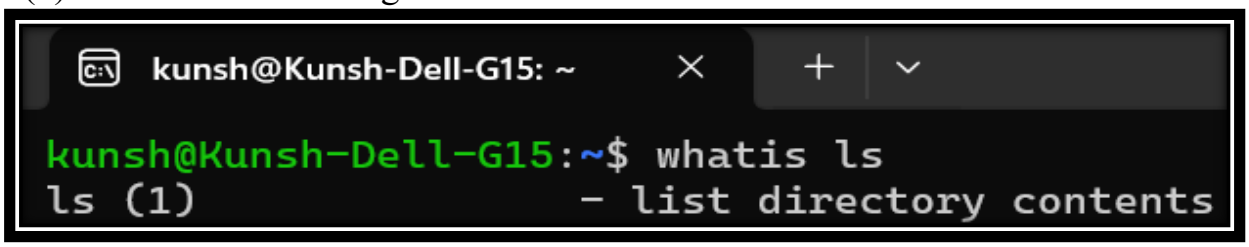


(f) Using help command, find the help on ‘man’ command:

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



(h) Demonstrate the usage of "whatis" command:



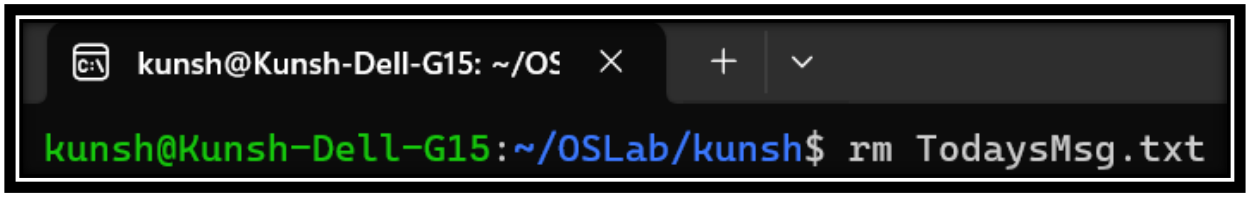
(i) Make directory named "OSLab/kunsh".

(j) Write command to reach to "kunsh" Directory:

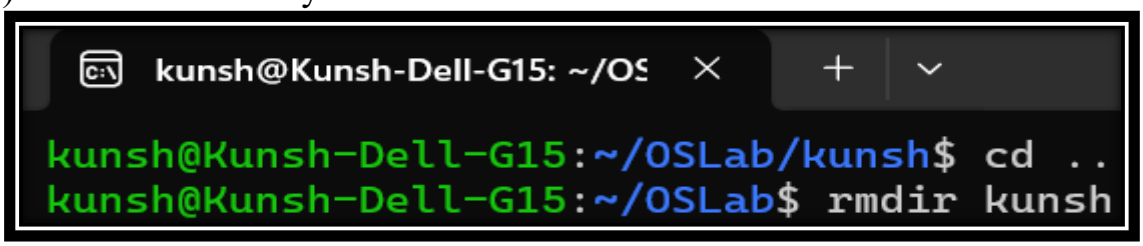
(k) Create a txt file named "TodaysMsg.txt" and write a greeting message in it:

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

(m) Delete the file "TodaysMsg.txt" from the 'kunsh' Folder:



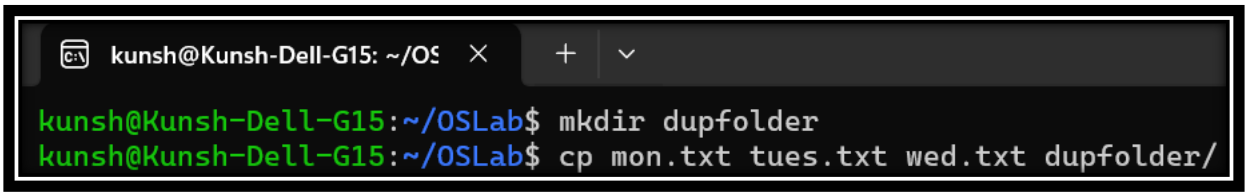
(n) Delete the directory 'kunsh':



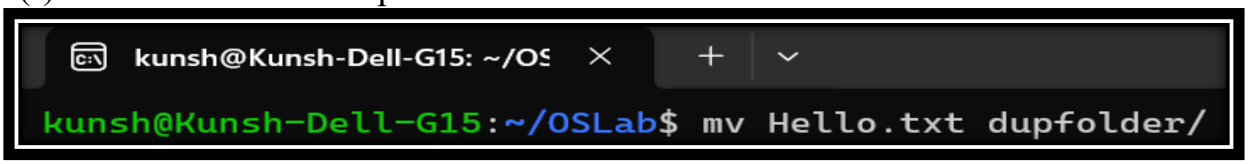
(o) Create a text file named “Hello.txt” and write a suitable message in it:

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

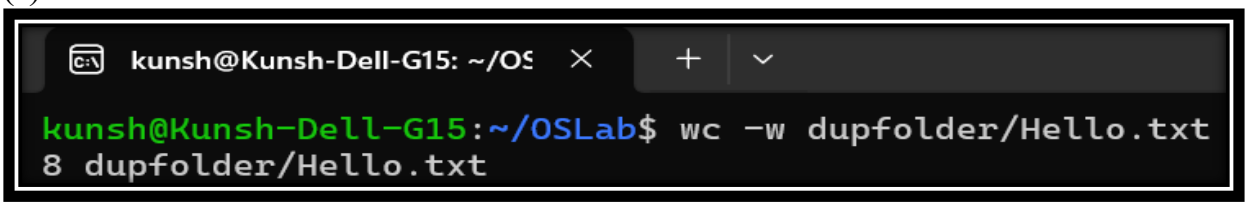
(q) Copy these newly created files to a folder named “dupfolder” after creating it:



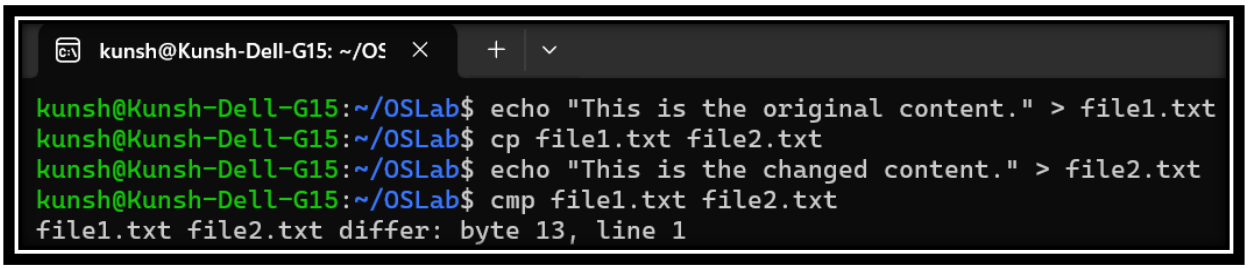
(r) Move Hello.txt to dupfolder:



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



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



Learning Outcome:

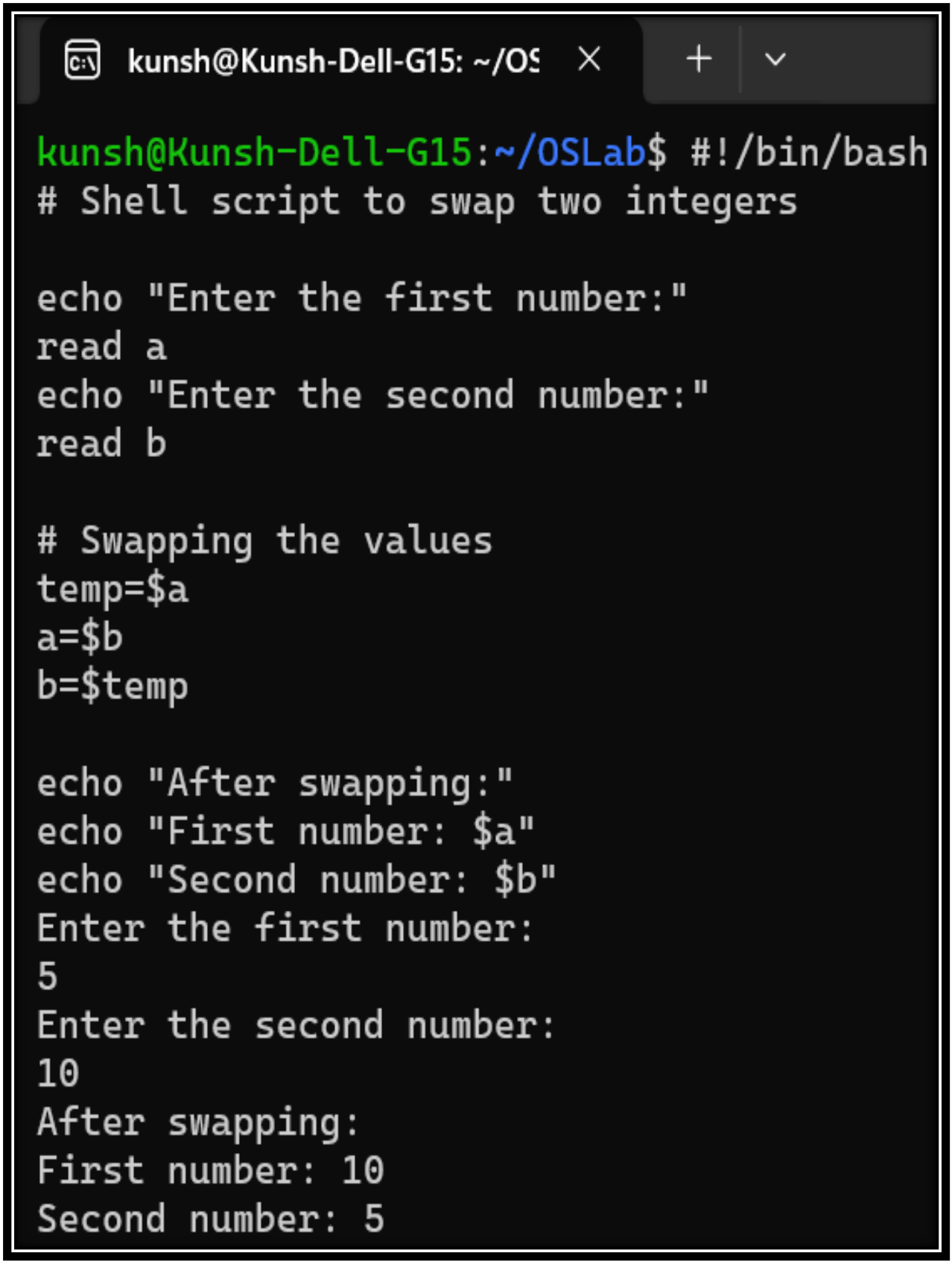
EXPERIMENT 3

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:

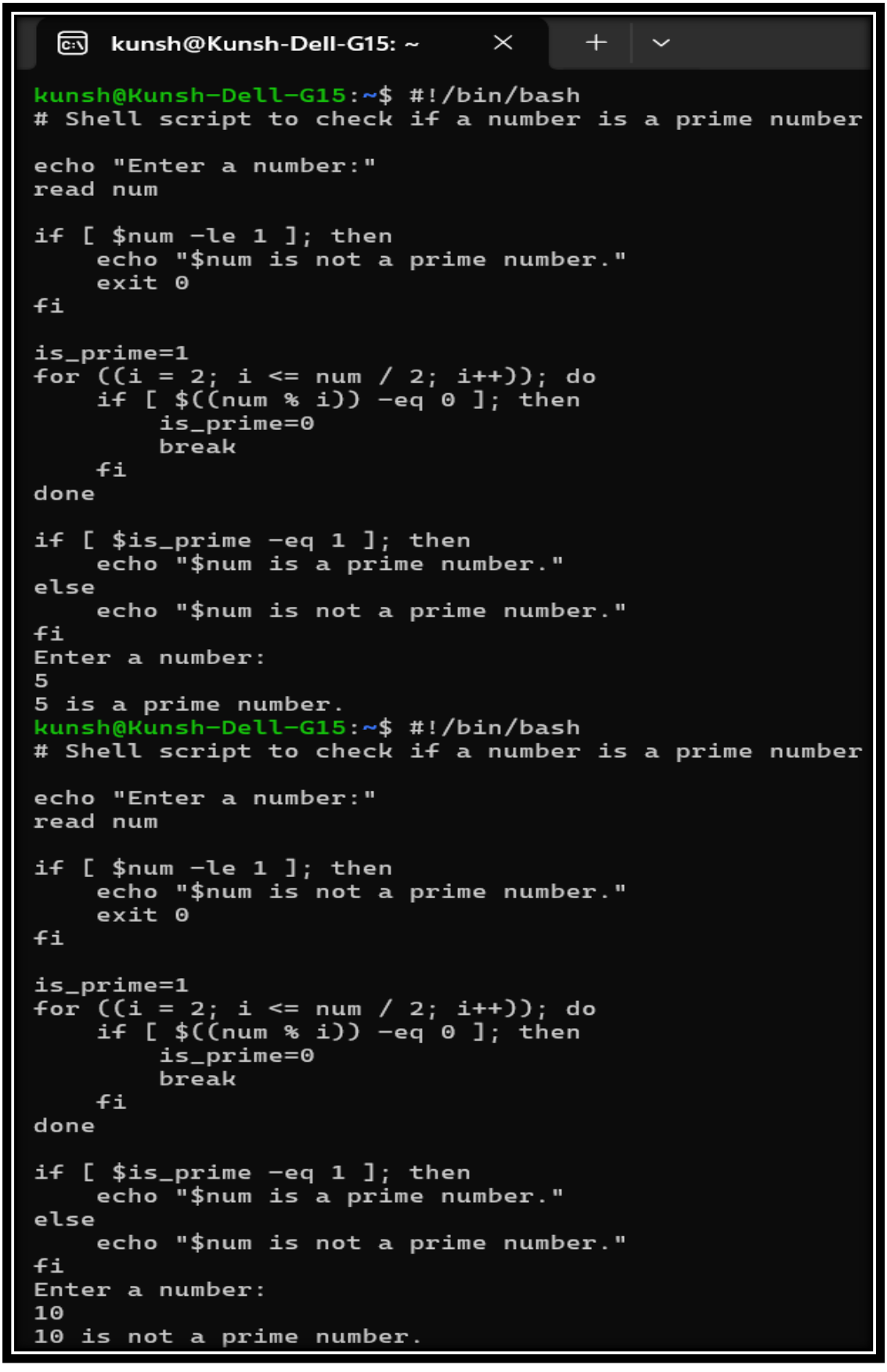


(b) Shell Script to Back Up Files from a Directory:

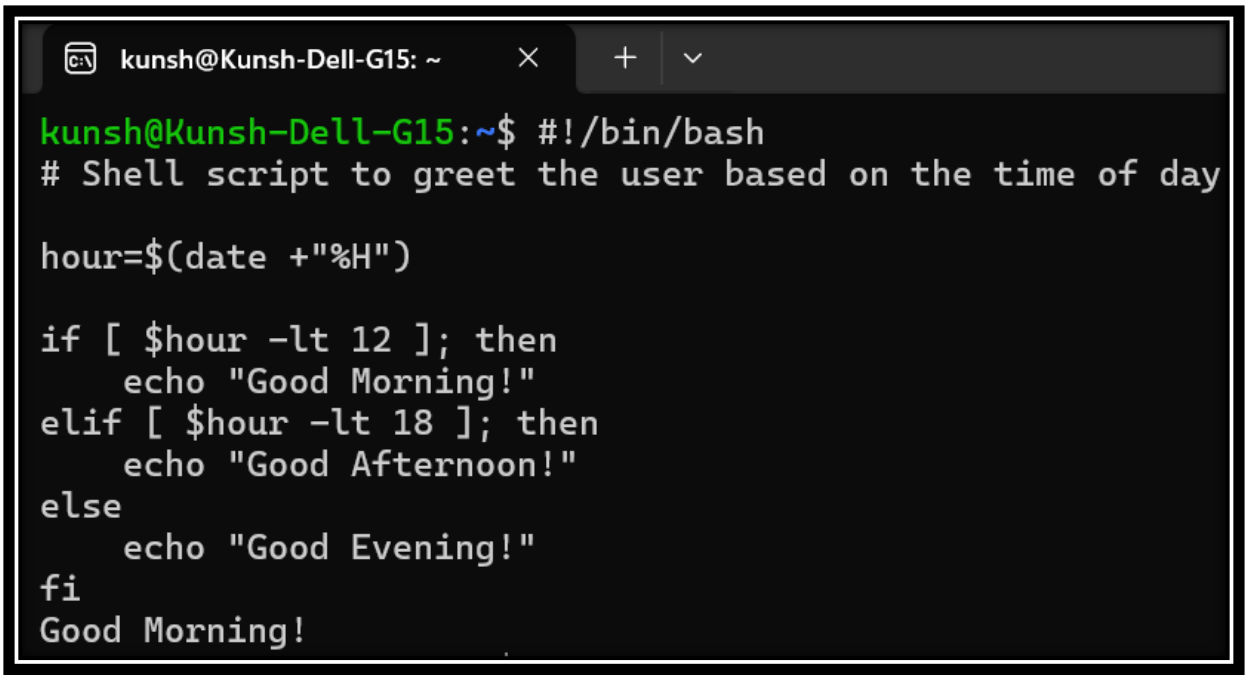
kunsh@Kunsh-Dell-G15: ~

```
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 $SOURCE_DIR does not exist."
fi
The following files were copied to /home/kunsh/os_lab:
TodaysMsg.txt
dupfolder
exp1
file1.txt
file2.txt
mon.txt
tues.txt
wed.txt
kunsh@Kunsh-Dell-G15:~$ #!/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:



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



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

FirstComeFirstServe.c

```c
#include <stdio.h>

// updated code for FCFS scheduling algorithm - EXP 4
void findWaitingTime(int processes[], int n, int bt[], int wt[])
{
    wt[0] = 0;
    for (int i = 1; i < n; i++)
    {
        wt[i] = bt[i - 1] + wt[i - 1];
    }
}

void findTurnAroundTime(int processes[], int n, int bt[], int wt[], int tat[])
{
    for (int i = 0; i < n; i++)
    {
        tat[i] = bt[i] + wt[i];
    }
}

void findAverageTime(int processes[], int n, int bt[])
{
    int wt[n], tat[n];
    findWaitingTime(processes, n, bt, wt);
    findTurnAroundTime(processes, n, bt, wt, tat);

    printf("Processes   Burst Time   Waiting Time   Turnaround Time\n");
    int total_wt = 0, total_tat = 0;
    for (int i = 0; i < n; i++)
    {
        total_wt += wt[i];
        total_tat += tat[i];
        printf("   %d ", (i + 1));
        printf("          %d ", bt[i]);
        printf("           %d ", wt[i]);
        printf("               %d\n", tat[i]);
    }

    printf("Average waiting time = %.2f\n", (float)total_wt / (float)n);
    printf("Average turnaround time = %.2f\n", (float)total_tat / (float)n);
}
```

```
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

```cpp
void findTurnAroundTime(int processes[], int n, int bt[], int wt[], int tat[])
{
    for (int i = 0; i < n; i++)
    {
        tat[i] = bt[i] + wt[i];
    }
}

void findAverageTime(int processes[], int n, int bt[], int quantum)
{
    int wt[n], tat[n];
    findWaitingTime(processes, n, bt, wt, quantum);
    findTurnAroundTime(processes, n, bt, wt, tat);

    printf("Processes   Burst Time   Waiting Time   Turnaround Time\n");
    int total_wt = 0, total_tat = 0;
    for (int i = 0; i < n; i++)
    {
        total_wt += wt[i];
        total_tat += tat[i];
        printf("   %d ", (i + 1));
        printf("          %d ", bt[i]);
        printf("           %d ", wt[i]);
        printf("               %d\n", tat[i]);
    }

    printf("Average waiting time = %.2f\n", (float)total_wt / (float)n);
    printf("Average turnaround time = %.2f\n", (float)total_tat / (float)n);
}

int main()
{
    int processes[] = {1, 2, 3};
    int n = sizeof processes / sizeof processes[0];
    int burst_time[] = {10, 5, 8};
    int quantum = 2;

    findAverageTime(processes, n, burst_time, quantum);
    return 0;
}
```

(c) Shortest Job First Scheduling Algorithm

ShortestJobFirst.cpp

```
#include <stdio.h>

void findWaitingTime(int processes[], int n, int bt[], int wt[])
{
    int sorted_processes[n], sorted_bt[n], temp;

    // Sorting burst times in ascending order using Bubble Sort
    for (int i = 0; i < n; i++)
    {
        sorted_processes[i] = i;
        sorted_bt[i] = bt[i];
    }
    for (int i = 0; i < n - 1; i++)
    {
        for (int j = 0; j < n - i - 1; j++)
        {
            if (sorted_bt[j] > sorted_bt[j + 1])
            {
                temp = sorted_bt[j];
                sorted_bt[j] = sorted_bt[j + 1];
                sorted_bt[j + 1] = temp;
                temp = sorted_processes[j];
                sorted_processes[j] = sorted_processes[j + 1];
                sorted_processes[j + 1] = temp;
            }
        }
    }

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

void findTurnAroundTime(int processes[], int n, int bt[], int wt[], int tat[])
{
    for (int i = 0; i < n; i++)
    {
        tat[i] = bt[i] + wt[i];
    }
}
```

ShortestJobFirst.cpp

```cpp
void findAverageTime(int processes[], int n, int bt[])
{
    int wt[n], tat[n];
    findWaitingTime(processes, n, bt, wt);
    findTurnAroundTime(processes, n, bt, wt, tat);

    printf("Processes   Burst Time   Waiting Time   Turnaround Time\n");
    int total_wt = 0, total_tat = 0;
    for (int i = 0; i < n; i++)
    {
        total_wt += wt[i];
        total_tat += tat[i];
        printf("   %d ", (i + 1));
        printf("          %d ", bt[i]);
        printf("           %d ", wt[i]);
        printf("                %d\n", tat[i]);
    }

    printf("Average waiting time = %.2f\n", (float)total_wt / (float)n);
    printf("Average turnaround time = %.2f\n", (float)total_tat / (float)n);
}

int main()
{
    int processes[] = {1, 2, 3};
    int n = sizeof processes / sizeof processes[0];
    int burst_time[] = {10, 5, 8};

    findAverageTime(processes, n, burst_time);
    return 0;
}
```

(d) Shortest Job Remaining First Scheduling Algorithm

ShortestJobRemainingFirst.cpp

```cpp
#include <stdio.h>
#include <limits.h>

void findWaitingTime(int processes[], int n, int bt[], int wt[])
{
    int rt[n];
    for (int i = 0; i < n; i++)
        rt[i] = bt[i];

    int complete = 0, t = 0, minm = INT_MAX;
    int shortest = 0, finish_time;
    int check = 0;

    while (complete != n)
    {
        for (int j = 0; j < n; j++)
        {
            if ((rt[j] < minm) && (rt[j] > 0))
            {
                minm = rt[j];
                shortest = j;
                check = 1;
            }
        }

        if (check == 0)
        {
            t++;
            continue;
        }

        rt[shortest]--;
        minm = rt[shortest];
        if (minm == 0)
            minm = INT_MAX;

        if (rt[shortest] == 0)
        {
            complete++;
            check = 0;
            finish_time = t + 1;
            wt[shortest] = finish_time - bt[shortest];
            if (wt[shortest] < 0)
                wt[shortest] = 0;
        }
        t++;
    }
}
```

ShortestJobRemainingFirst.cpp

```cpp
void findTurnAroundTime(int processes[], int n, int bt[], int wt[], int tat[])
{
    for (int i = 0; i < n; i++)
        tat[i] = bt[i] + wt[i];
}

void findAverageTime(int processes[], int n, int bt[])
{
    int wt[n], tat[n];
    findWaitingTime(processes, n, bt, wt);
    findTurnAroundTime(processes, n, bt, wt, tat);

    printf("Processes   Burst Time   Waiting Time   Turnaround Time\n");
    int total_wt = 0, total_tat = 0;
    for (int i = 0; i < n; i++)
    {
        total_wt += wt[i];
        total_tat += tat[i];
        printf("   %d ", (i + 1));
        printf("          %d ", bt[i]);
        printf("           %d ", wt[i]);
        printf("                %d\n", tat[i]);
    }

    printf("Average waiting time = %.2f\n", (float)total_wt / (float)n);
    printf("Average turnaround time = %.2f\n", (float)total_tat / (float)n);
}

int main()
{
    int processes[] = {1, 2, 3};
    int n = sizeof processes / sizeof processes[0];
    int burst_time[] = {6, 8, 7};

    findAverageTime(processes, n, burst_time);
    return 0;
}
```

Outputs: (a) First Come First Serve Scheduling Algorithm

```
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 ($?) { gc
c FirstComeFirstServe.c -o FirstComeFirstServe } ; if ($?) { .\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

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

```
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

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

FirstFitMemoryAllocation.c U

```c
#include <stdio.h>
void firstFit(int blockSize[], int m, int processSize[], int n)
{
    int allocation[n];
    for (int i = 0; i < n; i++)
    {
        allocation[i] = -1; // Initially, no block is assigned
    }

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

    printf("Process 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\n", allocation[i] + 1);
        else
            printf("Not Allocated\n");
    }
}

int main()
{
    int blockSize[] = {100, 500, 200, 300, 600};
    int processSize[] = {212, 417, 112, 426};
    int m = sizeof(blockSize) / sizeof(blockSize[0]);
    int n = sizeof(processSize) / sizeof(processSize[0]);

    firstFit(blockSize, m, processSize, n);
    return 0;
}
```

(b) Worst Fit Memory Allocation

WorstFitMemoryAllocation.c U

```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; // Initially, no block is assigned
    }

    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[j] > blockSize[wstIdx])
                {
                    wstIdx = j;
                }
            }
        }

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

    printf("Process 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\n", allocation[i] + 1);
        else
            printf("Not Allocated\n");
    }
}

int main()
{
    int blockSize[] = {100, 500, 200, 300, 600};
    int processSize[] = {212, 417, 112, 426};
    int m = sizeof(blockSize) / sizeof(blockSize[0]);
    int n = sizeof(processSize) / sizeof(processSize[0]);

    worstFit(blockSize, m, processSize, n);
    return 0;
}
```

(c) Best Fit Memory Allocation

BestFitMemoryAllocation.c U

```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; // Initially, no block is assigned
    }

    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[j] < blockSize[bestIdx])
                {
                    bestIdx = j;
                }
            }
        }

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

    printf("Process 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\n", allocation[i] + 1);
        else
            printf("Not Allocated\n");
    }
}
```

BestFitMemoryAllocation.c U

```c
int main()
{
    int blockSize[] = {100, 500, 200, 300, 600};
    int processSize[] = {212, 417, 112, 426};
    int m = sizeof(blockSize) / sizeof(blockSize[0]);
    int n = sizeof(processSize) / sizeof(processSize[0]);

    bestFit(blockSize, m, processSize, n);
    return 0;
}
```

Outputs: (a) First Fit Memory Allocation

```
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 ($?) { gcc FirstFitMemoryAllocation.c -o FirstFitMemoryAlloc
ation } ; if ($?) { .\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

```
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 ($?) { gcc WorstFitMemoryAllocation.c -o Wor
stFitMemoryAllocation } ; if ($?) { .\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

```
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 ($?) { gc
c BestFitMemoryAllocation.c -o BestFitMemoryAllocation } ; if ($?) { .\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:

kunsh@Kunsh-Dell-G15: /mnt

GNU nano 7.2

```c
#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;
}
```

kunsh@Kunsh-Dell-G15: /mnt

GNU nano 7.2

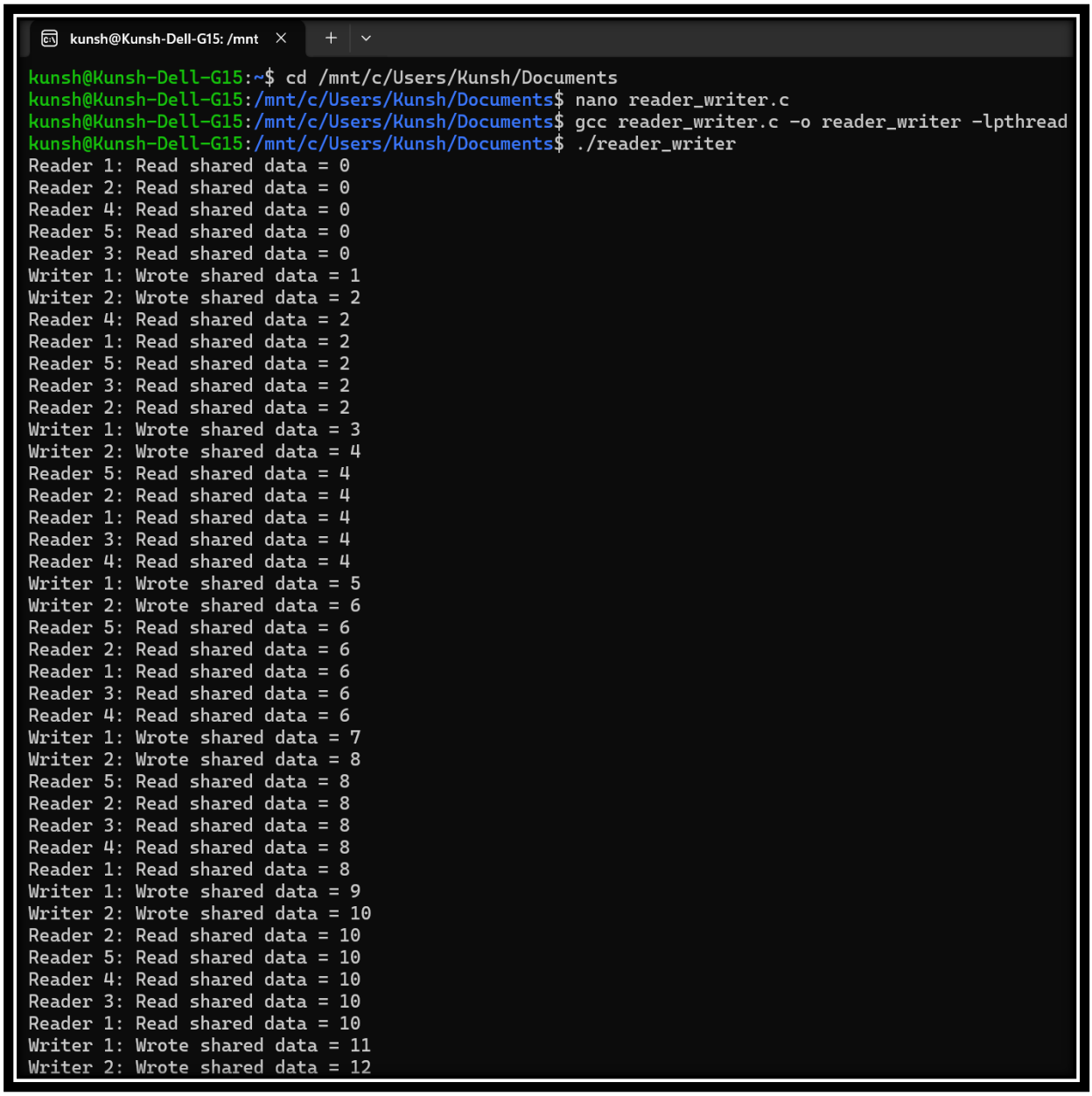
```c
void *writer(void *arg)
{
    int writer_id = *((int *)arg);
    while (1)
    {
        // Writer entry section
        sem_wait(&rw_mutex);

        // 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;
}
```

kunsh@Kunsh-Dell-G15: /mnt

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:



Learning Outcome:

EXPERIMENT 7

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

Theory:

Source Code:

BankersAlgorithm.c

```c
#include <stdio.h>
#include <stdbool.h>

#define P 5
#define R 3

// updated and modified version

bool isSafe(int processes[], int n, int avail[], int max[][R], int allot[][R])
{
    int work[R];
    bool finish[n];
    for (int i = 0; i < R; i++)
        work[i] = avail[i];
    for (int i = 0; i < n; i++)
        finish[i] = false;
    int safeSeq[n];
    int count = 0;
    while (count < n)
    {
        bool found = false;
        for (int p = 0; p < n; p++)
        {
            if (finish[p] == false)
            {
                int j;
                for (j = 0; j < R; j++)
                    if (max[p][j] - allot[p][j] > work[j])
                        break;
                if (j == R)
                {
                    for (int k = 0; k < R; k++)
                        work[k] += allot[p][k];
                    safeSeq[count++] = p;
                    finish[p] = true;
```

BankersAlgorithm.c

```c
                found = true;
            }
        }
    }
    if (found == false)
    {
        printf("System is not in a safe state\n");
        return false;
    }
}
printf("System is in a safe state.\nSafe sequence is: ");
for (int i = 0; i < n; i++)
    printf("P%d ", safeSeq[i]);
printf("\n");
return true;
}

bool requestResources(int processes[], int n, int avail[], int max[][R], int allot[][R], int req[])
{
    int need[n][R];
    for (int i = 0; i < n; i++)
        for (int j = 0; j < R; j++)
            need[i][j] = max[i][j] - allot[i][j];
    for (int j = 0; j < R; j++)
        if (req[j] > need[processes[0]][j])
            return false;
    for (int j = 0; j < R; j++)
        if (req[j] > avail[j])
            return false;
    int old_avail[R];
    int old_allot[n][R];
    for (int j = 0; j < R; j++)
    {
        old_avail[j] = avail[j];
        avail[j] -= req[j];
```

BankersAlgorithm.c

```c
    }
    for (int i = 0; i < n; i++)
        for (int j = 0; j < R; j++)
            old_allot[i][j] = allot[i][j];
    for (int j = 0; j < R; j++)
        allot[processes[0]][j] += req[j];
    if (isSafe(processes, n, avail, max, allot))
    {
        return true;
    }
    else
    {
        for (int j = 0; j < R; j++)
        {
            avail[j] = old_avail[j];
            allot[processes[0]][j] = old_allot[processes[0]][j];
        }
        return false;
    }
}

int main()
{
    int processes[] = {0, 1, 2, 3, 4};
    int n = sizeof(processes) / sizeof(processes[0]);
    int resources[] = {10, 5, 7};
    int m = sizeof(resources) / sizeof(resources[0]);
    int avail[] = {3, 3, 3};
    int max[][R] = {
        {7, 5, 3},
        {3, 2, 2},
        {9, 0, 2},
        {2, 2, 2},
        {4, 3, 3}};
```

BankersAlgorithm.c

```c
    int allot[][R] = {
        {0, 1, 0},
        {2, 0, 0},
        {3, 0, 2},
        {2, 1, 1},
        {0, 0, 2}};
    int request[] = {1, 0, 2};
    if (requestResources(processes, n, avail, max, allot, request))
    {
        printf("Request can be granted safely.\n");
    }
    else
    {
        printf("Request cannot be granted safely.\n");
    }
    return 0;
}

```

Output:

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:

kunsh@Kunsh-Dell-G15: ~

GNU nano 7.2

```c
#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:

kunsh@Kunsh-Dell-G15: ~

GNU nano 7.2

```c
#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:

kunsh@Kunsh-Dell-G15: ~

GNU nano 7.2

```c
#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:

kunsh@Kunsh-Dell-G15: ~

GNU nano 7.2

```c
#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:

kunsh@Kunsh-Dell-G15: ~

```
  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
-rwxr-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
-rwxr-xr-x 1 kunsh kunsh 16048 Oct 12 12:29 execlp
-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
-rwxr-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:

kunsh@Kunsh-Dell-G15: ~

GNU nano 7.2

```c
#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:

kunsh@Kunsh-Dell-G15: ~

GNU nano 7.2

```c
#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:

kunsh@Kunsh-Dell-G15: ~

GNU nano 7.2

```c
#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:

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:

kunsh@Kunsh-Dell-G15: ~

GNU nano 7.2

```c
#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: ~

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: