EXPERIMENT - 1

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

THEORY:

OUTPUT:

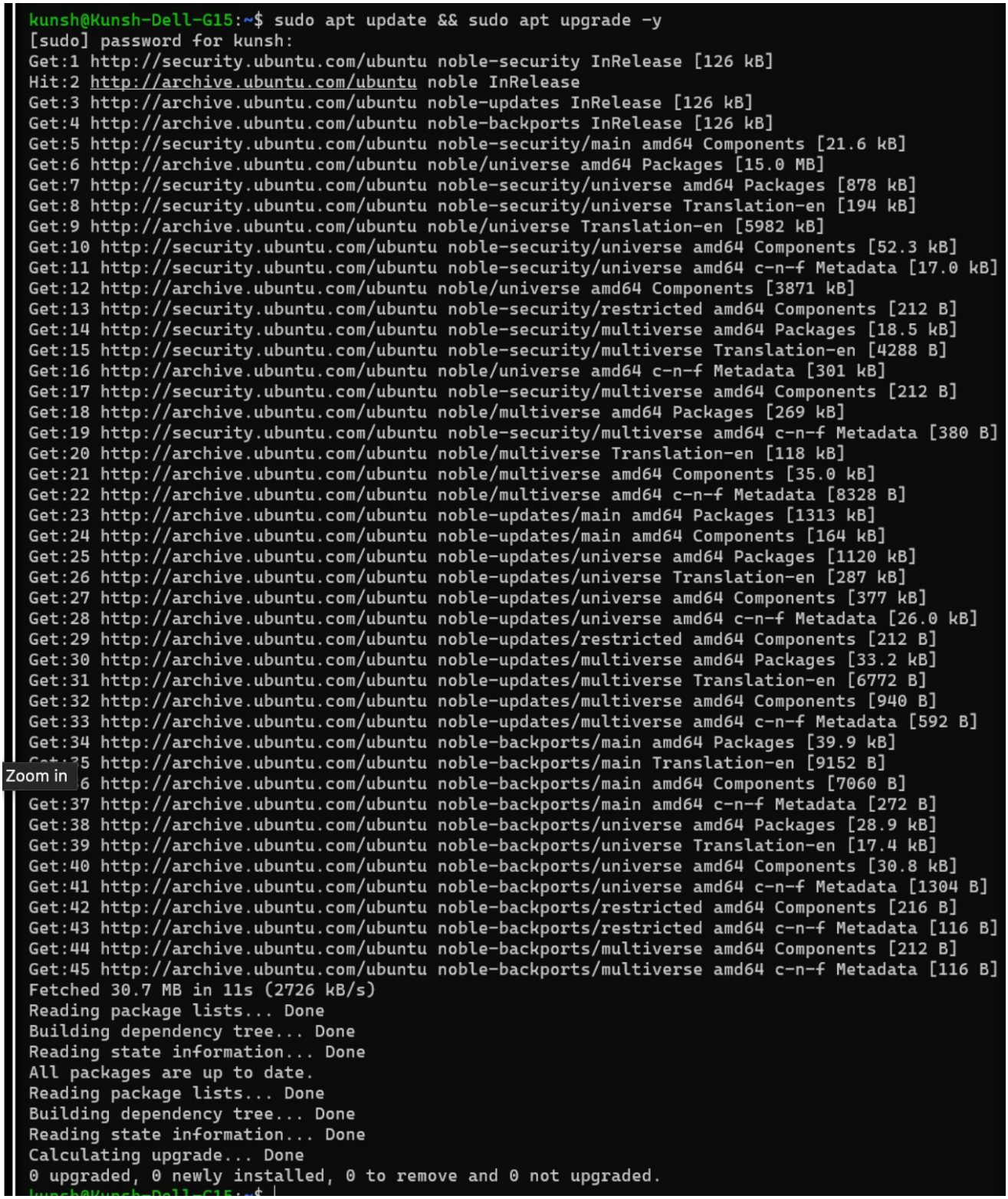
a. Installation Screenshot

```
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.
```

b. Ubuntu Terminal first launch screenshot

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

c. Running Update commands for Ubuntu



d. Installing gnu and C Compiler

```
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: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 ...
```

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.

```
:~$ mkdir -p ~/os_lab/exp1
:~$ cd ~/os_lab/exp1
:~/os_lab/exp1$ pwd
xp1
:~/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.

```
~/os_lab/exp1$ gcc -o hello hello.c
~/os_lab/exp1$ ./hello

 Hello, World!
```

LEARNING OUTCOME:

EXPERIMENT - 2

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

THEORY:

OUTPUT:

a. Command to know current working directory.

```
ubuntu@ubuntu:~$ pwd
/home/ubuntu
```

b. List all the files in the current directory.

```
ubuntu@ubuntu:~$ ls
Desktop    Downloads         Hello.txt  OSLab  Pictures  snap       Videos
Documents  firstfit.sh.save  Music      OSLAB  Public    Templates
```

c. List all files in order of their file size.

```
ubuntu@ubuntu:~$ ls -lS
total 52
drwxr-xr-x 2 ubuntu ubuntu 4096 Sep 21  2024 Desktop
drwxr-xr-x 2 ubuntu ubuntu 4096 Sep 21  2024 Documents
drwxr-xr-x 2 ubuntu ubuntu 4096 Sep 21  2024 Downloads
drwxr-xr-x 2 ubuntu ubuntu 4096 Sep 21  2024 Music
drwxrwxr-x 4 ubuntu ubuntu 4096 Aug 12 12:48 OSLab
drwxrwxr-x 3 ubuntu ubuntu 4096 Aug 11 16:05 OSLAB
drwxr-xr-x 3 ubuntu ubuntu 4096 Oct  9  2024 Pictures
drwxr-xr-x 2 ubuntu ubuntu 4096 Sep 21  2024 Public
drwx------ 4 ubuntu ubuntu 4096 Oct 29  2024 snap
drwxr-xr-x 2 ubuntu ubuntu 4096 Sep 21  2024 Templates
drwxr-xr-x 2 ubuntu ubuntu 4096 Sep 21  2024 Videos
-rw------- 1 ubuntu ubuntu  549 Oct 29  2024 firstfit.sh.save
-rw-rw-r-- 1 ubuntu ubuntu   39 Aug 11 16:06 Hello.txt
```

d. List only directories in the current folder.

```
ubuntu@ubuntu:~$ ls -d */
Desktop/    Downloads/  OSLab/  Pictures/  snap/      Videos/
Documents/  Music/      OSLAB/  Public/    Templates/
```

e. List only files starting with the alphabet "**N**".

```
ubuntu@ubuntu:~$ ls N*
ls: cannot access 'N*': No such file or directory
```

f. Using help command, find the help on '**man**' command.

```
ubuntu@ubuntu:~$ 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
```

g. Display the content of manual pages on '**ls**' command.

```
ubuntu@ubuntu:~$ man ls
```

h. Demonstrate the usage of "**whatis**" command.

```
ubuntu@ubuntu:~$ whatis ls
ls: nothing appropriate.
```

i. Make directory named **"OSLab/Chelsi".**

```
ubuntu@ubuntu:~$ mkdir -p OSLab/Chelsi
```

j. Write command to reach to "**Chelsi**" Directory.

```
ubuntu@ubuntu:~$ cd OSLab/Chelsi
```

k. Create a txt file named "**TodaysMsg.txt**" and write a greeting message in it.

```
ubuntu@ubuntu:~/OSLab/Chelsi$ echo "Hello, hope you have a great day!" > TodaysM
sg.txt
```

l. Copy this file "**TodaysMsg.txt**" to **OSLab** directory.

```
ubuntu@ubuntu:~/OSLab/Chelsi$ cp TodaysMsg.txt ../
```

m. Delete the file "TodaysMsg.txt" from the '**Chelsi**' Folder.

```
ubuntu@ubuntu:~/OSLab/Chelsi$ rm TodaysMsg.txt
```

n. Create a text file named "**Hello.txt**" and write a suitable message in it.

```
ubuntu@ubuntu:~/OSLab/Chelsi$ echo "This is a message in the Hello.txt file." >
Hello.txt
```

o. Using touch command create files with names **mon.txt, tues.txt**, and **wed.txt**.

```
ubuntu@ubuntu:~/OSLab/Chelsi$ touch mon.txt tues.txt wed.txt
```

p. Copy these newly created files to a folder named "**dupfolder**" after creating it.

```
ubuntu@ubuntu:~/OSLab/Chelsi$ mkdir dupfolder
```

```
ubuntu@ubuntu:~/OSLab/Chelsi$ cp mon.txt tues.txt wed.txt dupfolder/
```

q. Move Hello.txt to **dupfolder**.

```
ubuntu@ubuntu:~/OSLab/Chelsi$ mv Hello.txt dupfolder/
```

r. Count the number of words in the **Hello.txt** file.

```
ubuntu@ubuntu:~/OSLab/Chelsi$ wc -w dupfolder/Hello.txt
8 dupfolder/Hello.txt
```

s. Count the number of words in the **Hello.txt** file.

```
ubuntu@ubuntu:~/OSLab/Chelsi$ 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.

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

LEARNING OUTCOME:

EXPERIMENT - 3

AIM: 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:

OUTPUT:

a. Shell Program to Swap Two Integers.

```
#!/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"
```

```
ubuntu@ubuntu:~/OSLab/Chelsi$ sh 357.sh
Enter the first number:
678
Enter the second number:
123
After swapping:
First number: 123
Second number: 678
```

b. Shell Script to Back Up Files from a Directory.

```
  GNU nano 7.2                                          backup.sh *
#!/bin/bash
# Shell Script to backup files from a specific Directory

# Define the Directiories
SOURCE_DIR="OSLab"
BACKUP_DIR="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 succesfully 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
```

```
uc@UC-PC:~$ sh backup.sh
The following files were copied to os_lab:
TodaysMsg.txt
dupfolder
exp1
file1.txt
file2.txt
mon.txt
swap.sh
tues.txt
wed.txt
```

```
  GNU nano 7.2                                                backup.sh *
#!/bin/bash
# Shell Script to backup files from a specific Directory

# Define the Directiories
SOURCE_DIR="Ujjwal"
BACKUP_DIR="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 succesfully 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 $SOURCE_DIR does not exist."
fi
uc@UC-PC:~$ sh backup.sh
Error: Directory Ujjwal does not exist.
```

c. Shell Script to Check if a Number is Prime.

```
  GNU nano 7.2                          prime.sh *
#!/bin/bash
# Shell Script to check if a number is a prime.

echo "Enter Number:"
read num
```

```
if [ $num -le 1 ]; then
        echo "$num is not a Prime."
        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."
else
        echo "$num is not a Prime."
fi
```

```
uc@UC-PC:~/OSLab$ bash ./prime.sh
Enter Number:
7
7 is a Prime.
uc@UC-PC:~/OSLab$ bash ./prime.sh
Enter Number:
22
22 is not a Prime.
```

d. Shell Script to Greet the User Based on the Time of Day.

```
 GNU nano 7.2                    greet.sh *
#!/bin/bash
# Shell Script to greet the user based on the time of the day

hour=$(date +"%H")

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

```
uc@UC-PC:~/OSLab$ bash greet.sh
Good Evening!
```

LEARNING OUTCOME:

EXPERIMENT 4

AIM :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 :

Scheduling algorithms

a. First come first serve

b. Round Robin Scheduling

c. Shortest job first

d. Shortest Job remaining first.

SOURCE CODE:

a. First come first serve

```c
#include <stdio.h>

typedef struct {
    int pid;
    int arrival_time;
    int burst_time;
    int completion_time;
    int turnaround_time;
    int waiting_time;
} Process;

void fcfs(Process p[], int n);
void print_process_info(Process p[], int n);

int main() {
    int n;
    Process processes[20];

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

    for (int i = 0; i < n; i++) {
        printf("Enter arrival time and burst time for process %d: ", i + 1);
        scanf("%d %d", &processes[i].arrival_time, &processes[i].burst_time);
        processes[i].pid = i + 1;
    }

    fcfs(processes, n);
    print_process_info(processes, n);

    return 0;
}

```

```c
void fcfs(Process p[], int n) {
    // Sort processes by arrival time
    for (int i = 0; i < n - 1; i++) {
        for (int j = 0; j < n - i - 1; j++) {
            if (p[j].arrival_time > p[j + 1].arrival_time) {
                Process temp = p[j];
                p[j] = p[j + 1];
                p[j + 1] = temp;
            }
        }
    }

    int current_time = 0;

    for (int i = 0; i < n; i++) {
        if (current_time < p[i].arrival_time)
            current_time = p[i].arrival_time;

        p[i].completion_time = current_time + p[i].burst_time;
        p[i].turnaround_time = p[i].completion_time - p[i].arrival_time;
        p[i].waiting_time = p[i].turnaround_time - p[i].burst_time;

        current_time = p[i].completion_time;
    }
}

void print_process_info(Process p[], int n) {
    int total_tat = 0, total_wt = 0;

    printf("\nPID\tArrival\tBurst\tCompletion\tTurnaround\tWaiting\n");
    for (int i = 0; i < n; i++) {
        printf("%d\t%d\t%d\t%d\t\t%d\t\t%d\n",
               p[i].pid, p[i].arrival_time, p[i].burst_time,
               p[i].completion_time, p[i].turnaround_time, p[i].waiting_time);
        total_tat += p[i].turnaround_time;
        total_wt += p[i].waiting_time;
    }

    printf("\nAverage Turnaround Time = %.2f\n", (float)total_tat / n);
    printf("Average Waiting Time = %.2f\n", (float)total_wt / n);
}
```

b. Round Robin Scheduling

```c
#include <stdio.h>

typedef struct {
    int pid;
    int arrival_time;
    int burst_time;
    int remaining_time;
    int completion_time;
    int turnaround_time;
    int waiting_time;
} Process;

void round_robin(Process p[], int n, int quantum);
void print_process_info(Process p[], int n);

int main() {
    int n, quantum;
    Process processes[20];

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

    for (int i = 0; i < n; i++) {
        printf("Enter arrival time and burst time for process %d: ", i + 1);
        scanf("%d %d", &processes[i].arrival_time, &processes[i].burst_time);
        processes[i].pid = i + 1;
        processes[i].remaining_time = processes[i].burst_time;
    }

    printf("Enter time quantum: ");
    scanf("%d", &quantum);

    round_robin(processes, n, quantum);
    print_process_info(processes, n);
```

```
                }
            }
            if (idx == -1) break;
            queue[rear++] = idx;
            visited[idx] = 1;
            current_time = min_arrival;
        }

        int i = queue[front++];
        if (p[i].remaining_time > quantum) {
            current_time += quantum;
            p[i].remaining_time -= quantum;
        } else {
            current_time += p[i].remaining_time;
            p[i].remaining_time = 0;
            p[i].completion_time = current_time;
            p[i].turnaround_time = p[i].completion_time - p[i].arrival_time;
            p[i].waiting_time = p[i].turnaround_time - p[i].burst_time;
            completed++;
        }


        for (int j = 0; j < n; j++) {
            if (!visited[j] && p[j].arrival_time <= current_time) {
                queue[rear++] = j;
                visited[j] = 1;
            }
        }

        if (p[i].remaining_time > 0) {
            queue[rear++] = i;
        }
    }
}
```

```
    return 0;
}

void round_robin(Process p[], int n, int quantum) {
    int current_time = 0, completed = 0;
    int queue[100], front = 0, rear = 0;
    int visited[20] = {0};


    for (int i = 0; i < n - 1; i++) {
        for (int j = 0; j < n - i - 1; j++) {
            if (p[j].arrival_time > p[j + 1].arrival_time) {
                Process temp = p[j];
                p[j] = p[j + 1];
                p[j + 1] = temp;
            }
        }
    }


    queue[rear++] = 0;
    visited[0] = 1;
    current_time = p[0].arrival_time;

    while (completed < n) {
        if (front == rear) {

            int idx = -1, min_arrival = 9999;
            for (int i = 0; i < n; i++) {
                if (!visited[i] && p[i].arrival_time < min_arrival) {
                    min_arrival = p[i].arrival_time;
                    idx = i;
                }
```

```c
void print_process_info(Process p[], int n) {
    int total_tat = 0, total_wt = 0;

    printf("\nPID\tArrival\tBurst\tCompletion\tTurnaround\tWaiting\n");
    for (int i = 0; i < n; i++) {
        printf("%d\t%d\t%d\t%d\t\t%d\t\t%d\n",
               p[i].pid, p[i].arrival_time, p[i].burst_time,
               p[i].completion_time, p[i].turnaround_time, p[i].waiting_time);
        total_tat += p[i].turnaround_time;
        total_wt += p[i].waiting_time;
    }

    printf("\nAverage Turnaround Time = %.2f\n", (float)total_tat / n);
    printf("Average Waiting Time = %.2f\n", (float)total_wt / n);
}

```

c. Shortest job first

```c
#include <stdio.h>

typedef struct {
    int pid;
    int arrival_time;
    int burst_time;
    int completion_time;
    int turnaround_time;
    int waiting_time;
} Process;

void sjf(Process p[], int n);
void print_process_info(Process p[], int n);

int main() {
    int n;
    Process processes[20];

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

    for (int i = 0; i < n; i++) {
        printf("Enter arrival time and burst time for process %d: ", i + 1);
        scanf("%d %d", &processes[i].arrival_time, &processes[i].burst_time);
        processes[i].pid = i + 1;
    }

    sjf(processes, n);
    print_process_info(processes, n);

    return 0;
}

void sjf(Process p[], int n) {
    int completed = 0, current_time = 0;
```

```c
void sjf(Process p[], int n) {
    int completed = 0, current_time = 0;
    int is_completed[20] = {0};

    while (completed != n) {
        int idx = -1, min_burst = 9999;
        for (int i = 0; i < n; i++) {
            if (p[i].arrival_time <= current_time && !is_completed[i]) {
                if (p[i].burst_time < min_burst) {
                    min_burst = p[i].burst_time;
                    idx = i;
                } else if (p[i].burst_time == min_burst) {
                    if (p[i].arrival_time < p[idx].arrival_time)
                        idx = i;
                }
            }
        }

        if (idx != -1) {
            p[idx].completion_time = current_time + p[idx].burst_time;
            p[idx].turnaround_time = p[idx].completion_time - p[idx].arrival_time;
            p[idx].waiting_time = p[idx].turnaround_time - p[idx].burst_time;

            current_time = p[idx].completion_time;
            is_completed[idx] = 1;
            completed++;
        } else {
            current_time++;
        }
    }
}

void print_process_info(Process p[], int n) {
    int total_tat = 0, total_wt = 0;

    printf("\nPID\tArrival\tBurst\tCompletion\tTurnaround\tWaiting\n");
    for (int i = 0; i < n; i++) {
        printf("%d\t%d\t%d\t%d\t\t%d\t\t%d\n",
               p[i].pid, p[i].arrival_time, p[i].burst_time,
               p[i].completion_time, p[i].turnaround_time, p[i].waiting_time);
        total_tat += p[i].turnaround_time;
        total_wt += p[i].waiting_time;
    }

    printf("\nAverage Turnaround Time = %.2f\n", (float)total_tat / n);
    printf("Average Waiting Time = %.2f\n", (float)total_wt / n);
}
```

d. Shortest Job remaining first.

```c
#include <stdio.h>

typedef struct {
    int pid;
    int arrival_time;
    int burst_time;
    int remaining_time;
    int completion_time;
    int turnaround_time;
    int waiting_time;
} Process;

void srjf(Process p[], int n);
void print_process_info(Process p[], int n);

int main() {
    int n;
    Process processes[20];

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

    for (int i = 0; i < n; i++) {
        printf("Enter arrival time and burst time for process %d: ", i + 1);
        scanf("%d %d", &processes[i].arrival_time, &processes[i].burst_time);
        processes[i].pid = i + 1;
        processes[i].remaining_time = processes[i].burst_time;
    }

    srjf(processes, n);
    print_process_info(processes, n);

    return 0;
}
```

```c
void srjf(Process p[], int n) {
    int completed = 0, current_time = 0;
    int is_completed[20] = {0};

    while (completed != n) {
        int idx = -1, min_remaining = 9999;

        for (int i = 0; i < n; i++) {
            if (p[i].arrival_time <= current_time && !is_completed[i] && p[i]
                .remaining_time > 0) {
                if (p[i].remaining_time < min_remaining) {
                    min_remaining = p[i].remaining_time;
                    idx = i;
                } else if (p[i].remaining_time == min_remaining && idx != -1) {
                    if (p[i].arrival_time < p[idx].arrival_time)
                        idx = i;
                }
            }
        }

        if (idx != -1) {
            p[idx].remaining_time--;
            current_time++;

            if (p[idx].remaining_time == 0) {
                p[idx].completion_time = current_time;
                p[idx].turnaround_time = p[idx].completion_time - p[idx].arrival_time;
                p[idx].waiting_time = p[idx].turnaround_time - p[idx].burst_time;
                is_completed[idx] = 1;
                completed++;
            }
        } else {
            current_time++;
```

```
}

void print_process_info(Process p[], int n) {
    int total_tat = 0, total_wt = 0;

    printf("\nPID\tArrival\tBurst\tCompletion\tTurnaround\tWaiting\n");
    for (int i = 0; i < n; i++) {
        printf("%d\t%d\t%d\t%d\t\t%d\t\t%d\n",
               p[i].pid, p[i].arrival_time, p[i].burst_time,
               p[i].completion_time, p[i].turnaround_time, p[i].waiting_time);
        total_tat += p[i].turnaround_time;
        total_wt += p[i].waiting_time;
    }

    printf("\nAverage Turnaround Time = %.2f\n", (float)total_tat / n);
    printf("Average Waiting Time = %.2f\n", (float)total_wt / n);
}

```

OUTPUT :

a. First come first serve

```
Enter number of processes: 5
Enter arrival time and burst time for process 1: 1
4
Enter arrival time and burst time for process 2: 2
3
Enter arrival time and burst time for process 3: 1
5
Enter arrival time and burst time for process 4: 5
2
Enter arrival time and burst time for process 5: 7
1

PID Arrival Burst   Completion  Turnaround  Waiting
1   1   4   5       4       0
3   1   5   10      9       4
2   2   3   13      11      8
4   5   2   15      10      8
5   7   1   16      9       8

Average Turnaround Time = 8.60
Average Waiting Time = 5.60
```

b. Round Robin Scheduling

```
Enter number of processes: 5
Enter arrival time and burst time for process 1: 1
5
Enter arrival time and burst time for process 2: 2
3
Enter arrival time and burst time for process 3: 1
5
Enter arrival time and burst time for process 4: 5
2
Enter arrival time and burst time for process 5: 7
1
Enter time quantum: 3

PID Arrival Burst   Completion  Turnaround  Waiting
1   1   5   12      11      6
3   1   5   17      16      11
2   2   3   10      8       5
4   5   2   14      9       7
5   7   1   15      8       7

Average Turnaround Time = 10.40
Average Waiting Time = 7.20
```

c. Shortest job first

```
Enter number of processes: 5
Enter arrival time and burst time for process 1: 1
4
Enter arrival time and burst time for process 2: 2
3
Enter arrival time and burst time for process 3: 1
5
Enter arrival time and burst time for process 4: 5
2
Enter arrival time and burst time for process 5: 7
1

PID Arrival Burst   Completion  Turnaround  Waiting
1   1   4   5       4       0
2   2   3   11      9       6
3   1   5   16      15      10
4   5   2   7       2       0
5   7   1   8       1       0

Average Turnaround Time = 6.20
Average Waiting Time = 3.20
```

d. Shortest Job remaining first

```
Enter number of processes: 5
Enter arrival time and burst time for process 1: 1
4
Enter arrival time and burst time for process 2: 2
3
Enter arrival time and burst time for process 3: 1
5
Enter arrival time and burst time for process 4: 5
2
Enter arrival time and burst time for process 5: 7
1

PID Arrival Burst   Completion  Turnaround  Waiting
1   1   4   5       4       0
2   2   3   11      9       6
3   1   5   16      15      10
4   5   2   7       2       0
5   7   1   8       1       0

Average Turnaround Time = 6.20
Average Waiting Time = 3.20
```

LEARNING OUTCOMES :

Experiment-5

Aim: Implementation of the following Memory Allocation Methods for fixed partition: a.

First Fit

b. Worst Fit

c. Best Fit

Theory:

Memory Allocation Methods

a. First Fit

b. Worst Fit

c. Best Fit

Code:

a. First Fit

C: > Users > vips415-11 > Desktop > C 5.c > main()

```c
#include <stdio.h>

int main()
{
    int blockSize[] = {100, 500, 200, 300, 600};
    int m = sizeof(blockSize) / sizeof(blockSize[0]);

    int processSize[] = {212, 417, 112, 426};
    int n = sizeof(processSize) / sizeof(processSize[0]);

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

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

    return 0;
}
```

```
First Fit Allocation:
Process No.     Process Size    Block No.
1               212             2
2               417             5
3               112             2
4               426             Not Allocated
```

b. Worst Fit

C: > Users > vips415-11 > Desktop > C 5.c > main()

```c
#include <stdio.h>
int main()
{
    int blockSize[] = {100, 500, 200, 300, 600};
    int m = sizeof(blockSize) / sizeof(blockSize[0]);
    int processSize[] = {212, 417, 112, 426};
    int n = sizeof(processSize) / sizeof(processSize[0]);
    int allocation[n];
    for (int i = 0; i < n; i++)
        allocation[i] = -1;
    for (int i = 0; i < n; i++)
    {
        int worstIdx = -1;
        for (int j = 0; j < m; j++)
        {
            if (blockSize[j] >= processSize[i])
            {
                if (worstIdx == -1 || blockSize[j] > blockSize[worstIdx])
                    worstIdx = j;
            }
        }
        if (worstIdx != -1)
        {
            allocation[i] = worstIdx;
            blockSize[worstIdx] -= processSize[i];
        }
    }
    printf("Worst Fit Allocation:\nProcess No.\tProcess Size\tBlock No.\n");
    for (int i = 0; i < n; i++)
    {
        printf("%d\t\t%d\t\t", i + 1, processSize[i]);
        if (allocation[i] != -1)
            printf("%d\n", allocation[i] + 1);
        else
            printf("Not Allocated\n");
    }
    return 0;
}
```

```
Worst Fit Allocation:
Process No.     Process Size    Block No.
1               212             5
2               417             2
3               112             5
4               426             Not Allocated
```

c. Best Fit

C: > Users > vips415-11 > Desktop > 5.c > main()

```c
#include <stdio.h>
int main()
{
    int blockSize[] = {100, 500, 200, 300, 600};
    int m = sizeof(blockSize) / sizeof(blockSize[0]);
    int processSize[] = {212, 417, 112, 426};
    int n = sizeof(processSize) / sizeof(processSize[0]);
    int allocation[n];
    for (int i = 0; i < n; i++)
        allocation[i] = -1;
    for (int i = 0; i < n; i++)
    {
        int bestIdx = -1;
        for (int j = 0; j < m; j++)
        {
            if (blockSize[j] >= processSize[i])
            {
                if (bestIdx == -1 || blockSize[j] < blockSize[bestIdx])
                    bestIdx = j;
            }
        }
        if (bestIdx != -1)
        {
            allocation[i] = bestIdx;
            blockSize[bestIdx] -= processSize[i];
        }
    }
    printf("Best Fit Allocation:\nProcess No.\tProcess Size\tBlock No.\n");
    for (int i = 0; i < n; i++)
    {
        printf("%d\t\t%d\t\t", i + 1, processSize[i]);
        if (allocation[i] != -1)
            printf("%d\n", allocation[i] + 1);
        else
            printf("Not Allocated\n");
    }
    return 0;
}
```

```
Best Fit Allocation:
Process No.     Process Size    Block No.
1               212             4
2               417             2
3               112             3
4               426             5
PS C:\Users\vips415-11\Desktop>
```

Learning Outcomes:

EXPERIMENT-6

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

Theory:

Code:

```c
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <semaphore.h>
#include <unistd.h>

sem_t mutex;
sem_t wrt;
int readcount = 0;

void *reader(void *arg) {
    int id = *((int *)arg);
    int count = 0;

    while (count < 5) {
        sem_wait(&mutex);
        readcount++;
        if (readcount == 1)
            sem_wait(&wrt);
        sem_post(&mutex);

        printf("Reader %d is reading\n", id);
        sleep(1);

        sem_wait(&mutex);
        readcount--;
        if (readcount == 0)
            sem_post(&wrt);
        sem_post(&mutex);

        sleep(1);
        count++;
    }
    printf("Reader %d has finished reading.\n", id);
    return NULL;
}

```

```c
void *writer(void *arg) {
    int id = *((int *)arg);
    int count = 0;

    while (count < 5) {
        sem_wait(&wrt);

        printf("Writer %d is writing\n", id);
        sleep(2);

        sem_post(&wrt);

        sleep(2);
        count++;
    }
    printf("Writer %d has finished writing.\n", id);
    return NULL;
}

```

```c
int main() {
    pthread_t rtid[5], wtid[2];
    int rids[5], wids[2];

    sem_init(&mutex, 0, 1);
    sem_init(&wrt, 0, 1);

    for (int i = 0; i < 5; i++) {
        rids[i] = i + 1;
        pthread_create(&rtid[i], NULL, reader, &rids[i]);
    }

    for (int i = 0; i < 2; i++) {
        wids[i] = i + 1;
        pthread_create(&wtid[i], NULL, writer, &wids[i]);
    }

    for (int i = 0; i < 5; i++) pthread_join(rtid[i], NULL);
    for (int i = 0; i < 2; i++) pthread_join(wtid[i], NULL);

    sem_destroy(&mutex);
    sem_destroy(&wrt);

    return 0;
}

```

Output:

```
Reader 3 is reading
Reader 2 is reading
Reader 5 is reading
Reader 4 is reading
Writer 2 is writing
Writer 1 is writing
Reader 5 is reading
Reader 1 is reading
Reader 2 is reading
Reader 3 is reading
Reader 4 is reading
Writer 2 is writing
Writer 1 is writing
Reader 5 is reading
Reader 1 is reading
Reader 4 is reading
Reader 3 is reading
Reader 2 is reading
Writer 2 is writing
Writer 1 is writing
Reader 5 is reading
Reader 2 is reading
Reader 4 is reading
Reader 1 is reading
Reader 3 is reading
Writer 2 is writing
Writer 1 has finished writing.
Reader 2 has finished reading.
Reader 5 has finished reading.
Reader 4 has finished reading.
Reader 1 has finished reading.
Reader 3 has finished reading.
Writer 2 has finished writing.
```

Learning Outcomes:

EXPERIMENT-7

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

Theory:

Code:

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

int main() {
    int n, m;  // n = number of processes, m = number of resources

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

    int allocation[n][m], max[n][m], need[n][m], available[m];
    int finish[n], safeSeq[n];

    // Input Allocation Matrix
    printf("\nEnter Allocation Matrix:\n");
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < m; j++) {
            scanf("%d", &allocation[i][j]);
        }
    }

    // Input Max Matrix
    printf("\nEnter Max Matrix:\n");
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < m; j++) {
            scanf("%d", &max[i][j]);
        }
    }

    // Input Available Resources
    printf("\nEnter Available Resources:\n");
    for (int i = 0; i < m; i++) {
        scanf("%d", &available[i]);
    }

    // Calculate Need Matrix
    for (int i = 0; i < n; i++) {
```

```
    // Calculate Need Matrix
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < m; j++) {
            need[i][j] = max[i][j] - allocation[i][j];
        }
    }

    // Initialize Finish array
    for (int i = 0; i < n; i++) {
        finish[i] = 0;
    }

    int count = 0;
    while (count < n) {
        bool found = false;
        for (int p = 0; p < n; p++) {
            if (finish[p] == 0) {
                int j;
                for (j = 0; j < m; j++) {
                    if (need[p][j] > available[j])
                        break;
                }

                if (j == m) {
                    for (int k = 0; k < m; k++) {
                        available[k] += allocation[p][k];
                    }
                    safeSeq[count++] = p;
                    finish[p] = 1;
                    found = true;
                }
            }
        }

        if (found == false) {
            printf("\nSystem is NOT in a safe state.\n");
            return 0;
        }
```

```
74          }
75      }
76
77      printf("\nSystem is in a SAFE state.\nSafe sequence: ");
78      for (int i = 0; i < n; i++) {
79          printf("P%d ", safeSeq[i]);
80      }
81      printf("\n");
82
83      return 0;
84  }
```

Output:

```
Enter number of processes: 5
Enter number of resources: 3

Enter Allocation Matrix:
0 1 0
2 0 0
3 0 2
2 1 1
0 0 2

Enter Max Matrix:
7 5 3
3 2 2
9 0 2
2 2 2
4 3 3

Enter Available Resources:
3 3 2

System is in a SAFE state.
Safe sequence: P1 P3 P4 P0 P2
```

Learning Outcomes:

EXPERIMENT-8

Aim: Process management

a) Write a program to implement the fork function using c.
b) Write a program to implement the execv function using c.
c) Write a program to implement the execlp function using c.
d) Write a program to implement the wait function using c.
e) Write a program to implement the sleep function using c.

Theory:

Code:

a) Fork

Users > chelsiaggarwal > C main.c > ...

```c
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>

int main() {
    int n = 3; // number of child processes
    pid_t pid;

    for (int i = 1; i <= n; i++) {
        pid = fork();

        if (pid < 0) {
            // If fork() fails
            printf("Fork failed at iteration %d\n", i);
            return 1;
        }
        else if (pid == 0) {
            // Child process
            printf("Child %d created. PID: %d, Parent PID: %d\n", i, getpid(), getppid());
            // Each child does some task
            sleep(1); // simulate work
            printf("Child %d (PID: %d) finished work.\n", i, getpid());
            return 0; // important to prevent child from creating more children
        }
        else {
            // Parent process continues the loop to create more children
            continue;
        }
    }

    // Parent waits to ensure all children complete
    sleep(2);
    printf("Parent process (PID: %d) created %d child processes.\n", getpid(), n);

    return 0;
}
```

b) execv

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

int main() {
    // Path to the new program
    char *path = "/bin/ls";

    // Arguments for execv
    char *args[] = {"ls", "-l", "/usr", NULL};

    printf("Before execv() call\n");

    // Replace current process with /bin/ls
    execv(path, args);

    // Only runs if execv fails
    perror("execv failed");

    return 1;
}
```

c) ***execlp***

```c
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>

int main() {
    printf("Before execlp() call\n");

    // execlp searches for the command in system PATH
    // Command: ls -l /usr
    execlp("ls", "ls", "-l", "/usr", NULL);

    // Only executes if execlp fails
    perror("execlp failed");
    exit(1);
}
```

d) ***wait***

```c
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <stdlib.h>

int main() {
    pid_t pid;
    int status;

    pid = fork(); // Create child process

    if (pid < 0) {
        perror("fork failed");
        return 1;
    }
    else if (pid == 0) {
        // Child process
        printf("Child process PID = %d\n", getpid());
        sleep(2); // simulate work
        exit(5);  // child exits with status 5
    }
    else {
        // Parent process waits for child
        printf("Parent waiting for child to finish...\n");
        pid_t child_pid = wait(&status);

        printf("Child with PID %d finished.\n", child_pid);

        if (WIFEXITED(status)) {
            printf("Child exited with status = %d\n", WEXITSTATUS(status));
        }
    }

    return 0;
}
```

e) sleep

```c
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <stdlib.h>

int main() {
    pid_t pid;

    pid = fork(); // Create child process

    if (pid < 0) {
        perror("fork failed");
        return 1;
    }
    else if (pid == 0) {
        // Child process
        printf("Child starts. PID = %d\n", getpid());
        for (int i = 3; i > 0; i--) {
            printf("Child counting: %d\n", i);
            sleep(1); // Child sleeps 1 second
        }
        printf("Child finished.\n");
        exit(0);
    }
    else {
        // Parent process
        printf("Parent starts. PID = %d\n", getpid());
        for (int i = 5; i > 0; i--) {
            printf("Parent counting: %d\n", i);
            sleep(2); // Parent sleeps 2 seconds
        }
        wait(NULL); // Wait for child to finish
        printf("Parent finished after waiting for child.\n");
    }

    return 0;
}
```

Output:

a) fork

```
Child 1 created. PID: 7214, Parent PID: 7212
Child 2 created. PID: 7215, Parent PID: 7212
Child 3 created. PID: 7216, Parent PID: 7212
Child 2 (PID: 7215) finished work.
Child 3 (PID: 7216) finished work.
Child 1 (PID: 7214) finished work.
Parent process (PID: 7212) created 3 child processes.
```

b) execv

```
Before execv() call
total 0
lrwxr-xr-x    1 root  wheel     25 Sep  9 12:45 X11 -> ../private/var/select/X11
lrwxr-xr-x    1 root  wheel     25 Sep  9 12:45 X11R6 -> ../private/var/select/X11
drwxr-xr-x  918 root  wheel  29376 Sep  9 12:45 bin
drwxr-xr-x   32 root  wheel   1024 Sep  9 12:45 lib
drwxr-xr-x  420 root  wheel  13440 Sep  9 12:45 libexec
drwxr-xr-x    7 root  wheel    224 Sep 17 10:56 local
drwxr-xr-x  230 root  wheel   7360 Sep  9 12:45 sbin
drwxr-xr-x   43 root  wheel   1376 Sep  9 12:45 share
drwxr-xr-x    5 root  wheel    160 Sep  9 12:45 standalone
```

c) execlp

```
Before execlp() call
total 0
lrwxr-xr-x    1 root  wheel     25 Sep  9 12:45 X11 -> ../private/var/select/X11
lrwxr-xr-x    1 root  wheel     25 Sep  9 12:45 X11R6 -> ../private/var/select/X11
drwxr-xr-x  918 root  wheel  29376 Sep  9 12:45 bin
drwxr-xr-x   32 root  wheel   1024 Sep  9 12:45 lib
drwxr-xr-x  420 root  wheel  13440 Sep  9 12:45 libexec
drwxr-xr-x    7 root  wheel    224 Sep 17 10:56 local
drwxr-xr-x  230 root  wheel   7360 Sep  9 12:45 sbin
drwxr-xr-x   43 root  wheel   1376 Sep  9 12:45 share
drwxr-xr-x    5 root  wheel    160 Sep  9 12:45 standalone
```

d) wait

```
Parent waiting for child to finish.
Child process PID = 7382
Child with PID 7382 finished.
Child exited with status = 5
```

e) sleep

```
Parent starts. PID = 7441
Parent counting: 5
Child starts. PID = 7442
Child counting: 3
Child counting: 2
Parent counting: 4
Child counting: 1
Child finished.
Parent counting: 3
Parent counting: 2
Parent counting: 1
Parent finished after waiting for child.
```

Learning Outcomes:

EXPERIMENT-9

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

Theory:

Code:

```c
// IPC using Message Queues
// One process sends message; another receives it.

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/ipc.h>
#include <sys/msg.h>
#include <unistd.h>

// Structure for message
struct msg_buffer {
    long msg_type;
    char msg_text[100];
} message;

int main() {
    key_t key;
    int msgid;

    // Generate unique key
    key = ftok("progfile", 65); // 'progfile' can be any existing file

    // Create message queue and return id
    msgid = msgget(key, 0666 | IPC_CREAT);
    if (msgid == -1) {
        perror("msgget failed");
        exit(1);
    }

    pid_t pid = fork();

    if (pid < 0) {
        perror("Fork failed");
        exit(1);
    }

    // Child process → sender
    if (pid == 0) {
        message.msg_type = 1;

        printf("Enter a message to send: ");
        fgets(message.msg_text, sizeof(message.msg_text), stdin);
        message.msg_text[strcspn(message.msg_text, "\n")] = '\0'; // remove newline

        // Send message
        if (msgsnd(msgid, &message, sizeof(message.msg_text), 0) == -1) {
            perror("msgsnd failed");
            exit(1);
        }

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

    // Parent process → receiver
    else {
        sleep(2); // ensure sender sends first

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

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

        // Destroy the message queue
        msgctl(msgid, IPC_RMID, NULL);
    }

    return 0;
}
```

Output:

```
Enter a message to send: Hello! OS-Lab
Message sent: Hello! OS-Lab
Message received: Hello! OS-Lab
```

Learning Outcomes:

EXPERIMENT-10

Aim: Write a program to implement IPC using pipes.

Theory:

Code:

```c
#include <stdio.h>
#include <unistd.h>
#include <string.h>
#include <stdlib.h>

int main() {
    int pipefd[2];
    pid_t pid;
    char write_msg[100], read_msg[100];

    if (pipe(pipefd) == -1) {
        perror("Pipe failed");
        exit(1);
    }

    pid = fork();

    if (pid < 0) {
        perror("Fork failed");
        exit(1);
    }

    if (pid == 0) {
        close(pipefd[1]);
        read(pipefd[0], read_msg, sizeof(read_msg));
        printf("Child Process received: %s\n", read_msg);
        close(pipefd[0]);
    } else {
        close(pipefd[0]);
        printf("Enter a message to send: ");
        fgets(write_msg, sizeof(write_msg), stdin);
        write_msg[strcspn(write_msg, "\n")] = '\0';
        write(pipefd[1], write_msg, strlen(write_msg) + 1);
        printf("Parent Process sent: %s\n", write_msg);
        close(pipefd[1]);
    }

    return 0;
}
```

Output:

```
Enter a message to send: Hello! OS-Lab
Parent Process sent: Hello! OS-Lab
Child Process received: Hello! OS-Lab
```

Learning Outcomes:

EXPERIMENT-11

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

Theory:

Code:

```c
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <ctype.h>

void* count_words(void* filename) {
    FILE* fp = fopen((char*)filename, "r");
    if (fp == NULL) {
        perror("Error opening file");
        pthread_exit(NULL);
    }

    int words = 0;
    char ch, prev = ' ';
    while ((ch = fgetc(fp)) != EOF) {
        if (isspace(ch) && !isspace(prev))
            words++;
        prev = ch;
    }
    fclose(fp);

    printf("Child Thread: Number of words = %d\n", words);
    pthread_exit(NULL);
}

int main() {
    pthread_t tid;
    char filename[100];

    printf("Enter filename: ");
    scanf("%s", filename);

    FILE* fp = fopen(filename, "r");
    if (fp == NULL) {
        perror("Error opening file");
        return 1;
    }

    pthread_create(&tid, NULL, count_words, (void*)filename);

    int lines = 0;
    char ch;
    while ((ch = fgetc(fp)) != EOF) {
        if (ch == '\n')
            lines++;
    }
    fclose(fp);

    printf("Main Thread: Number of lines = %d\n", lines);

    pthread_join(tid, NULL);

    return 0;
}
```

Output:

```
Enter filename: sample.txt
Main Thread: Number of lines = 2
Child Thread: Number of words = 14
```

Learning Outcomes:

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

BTECH Programme: AIML

Course Title: Operating Systems lab

Course Code: AIML351

Submitted To

Dr. Shivanka

Submitted By

Name: Rony Mandal
Enrollment No: 08817711623

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.

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 Name	Date	Marks			Remark	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/<yourname>". <yourname> should be the name of the student performing this experiment. j) Write command to reach to "yourname" Directory. k) Create a txt file named "TodaysMsg.txt" and write a greeting message in it.							

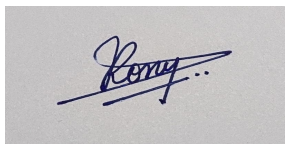
	l) Copy this file "TodaysMsg.txt" to OSLab directory m) Delete the file "TodaysMsg.txt" from the 'yourname' Folder n) Delete the directory 'yourname' o) Create a text file named "Hello.txt" and write a suitable message in it. p) Using touch command create files with names mon.txt, tues.txt, and wed.txt q) Copy these newly created files to a folder named "dupfolder" after creating it. r) Move Hello.txt to dupfolder s) Count 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. Shortes 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() and 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.							

IIT Madras BS in Data Science and Applications

Hall Ticket for	2025 Nov23: IIT M AN EXAM QDF2	Number of Answer Booklets	0

Candidate Name	RONY MANDAL				
Roll No	DS23F2002480			**Seating Number**	34705369
Subject	PDSA, Appdev2				
Date of Birth	05-04-2004				
PwD Status	No	**Compensatory Time Required**	No	**Scribe Required**	No
Exam Date	**Sunday, 23 November, 2025**				
Reporting Time	**01:00 PM IST**			**Gate Closure**	**02:30 PM IST**
Exam Timing	**Refer to Table in Page 2**			**Shift**	**Afternoon**
Test Centre Name	iON Digital Zone iDZ 1 Mathura Road				
Test Centre Address	A 27, Mohan Co-Op Industrial Estate, Near Sarita Vihar Metro Station, Near Metro Pillar No. 292-293, New Delhi, Delhi, India - 110044				



IIT Madras BS Degree Coordinator

You are not allowed to write the exam at any center other than the one mentioned in your hall ticket.

IITM BS Degree Exams
General Instructions for candidates
(All timings mentioned here are in IST)

DRESS CODE: Candidates are expected to come in professional attire to write the exams.
Candidates wearing SHORTS will NOT be permitted inside the exam hall
AT THE EXAM CENTRE: If you encounter any issues with respect to the computer or exam officials, kindly contact the IIT MADRAS exam representative, who will be available at the centre.

Hall Ticket and Entry:

1. The Hall Ticket must be presented for verification along with one original printed photo identification containing your Date of Birth (not photocopy or scanned copy). Examples of acceptable photo identification documents are BS degree ID card, College ID, Employee ID, Driving License, Passport, PAN card, Voter ID, Aadhaar-ID. Printed copy of the hall ticket and original photo id card should be brought to the exam centre. Hall ticket and id card copies on the phone will not be permitted.
2. This Hall Ticket is valid only if the candidate's photograph and signature images are visible. To ensure this, print the Hall Ticket on A4 sized paper using a laser printer, preferably a colour photo printer.
3. ON THE TOP PART OF YOUR SCREEN, YOU WILL SEE THE EXAM SUBJECTS SHOWN AS ONE OR MORE SECTIONS WITHIN THE QUESTION PAPER. CLICK ON THE ARROWS AT BOTH ENDS TO LOCATE YOUR SUBJECT(S) FOR THE EXAM. SELECT ONLY THE SECTIONS/SUBJECTS RELEVANT TO YOU AND ANSWER. IF YOU ANSWER ANY OTHER SUBJECTS NOT ELIGIBLE FOR YOU, IT WILL NOT BE CONSIDERED OR COUNTED.
4. Candidates will be permitted to appear for the examination ONLY after their credentials are verified by center officials.
5. Candidates are advised to locate the examination center at least a day prior to the examination, so that they can reach the center on time for the examination.