

PRACTICAL NO. 1

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

Steps to Install Ubuntu Terminal Environment on Windows with Windows Subsystem for Linux (WSL):

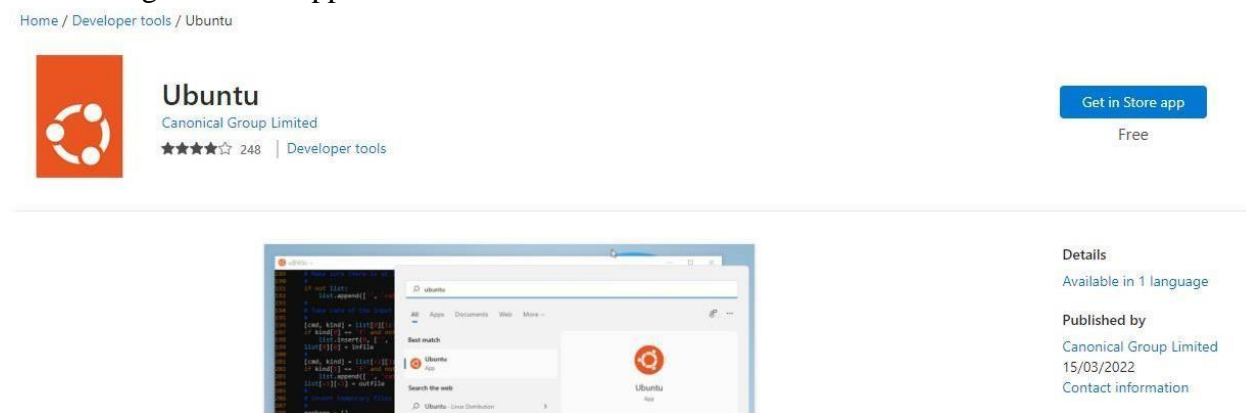
- 1) Go to following URL <https://ubuntu.com/desktop/wsl> and click download from the Microsoft Store.



- 2) It will take you to

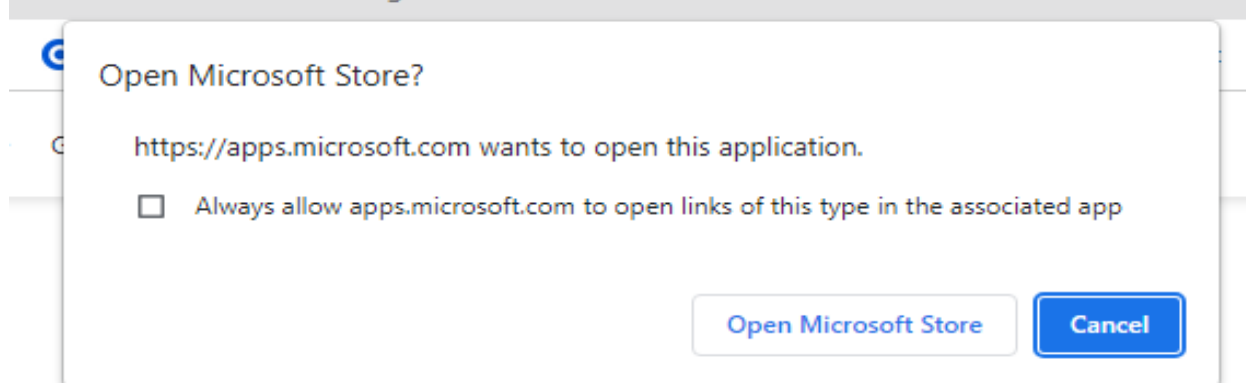
<https://apps.microsoft.com/store/detail/ubuntu/9PDXGNCFCSCZV?hl=en-in&gl=in&rtc=1>

then click get in Store app

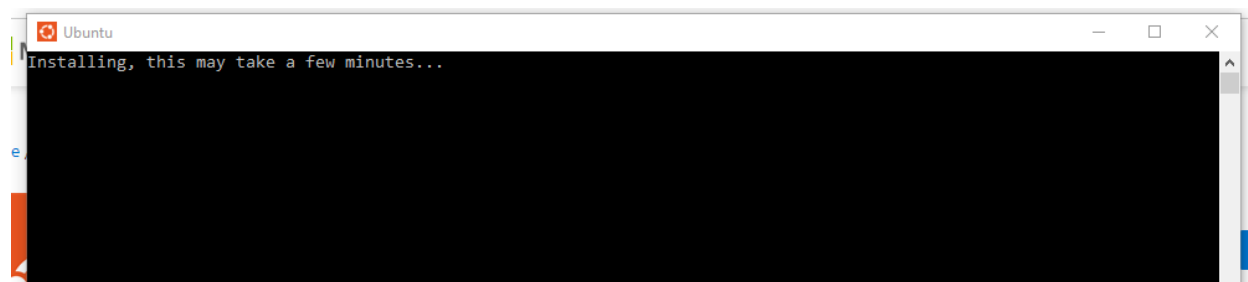
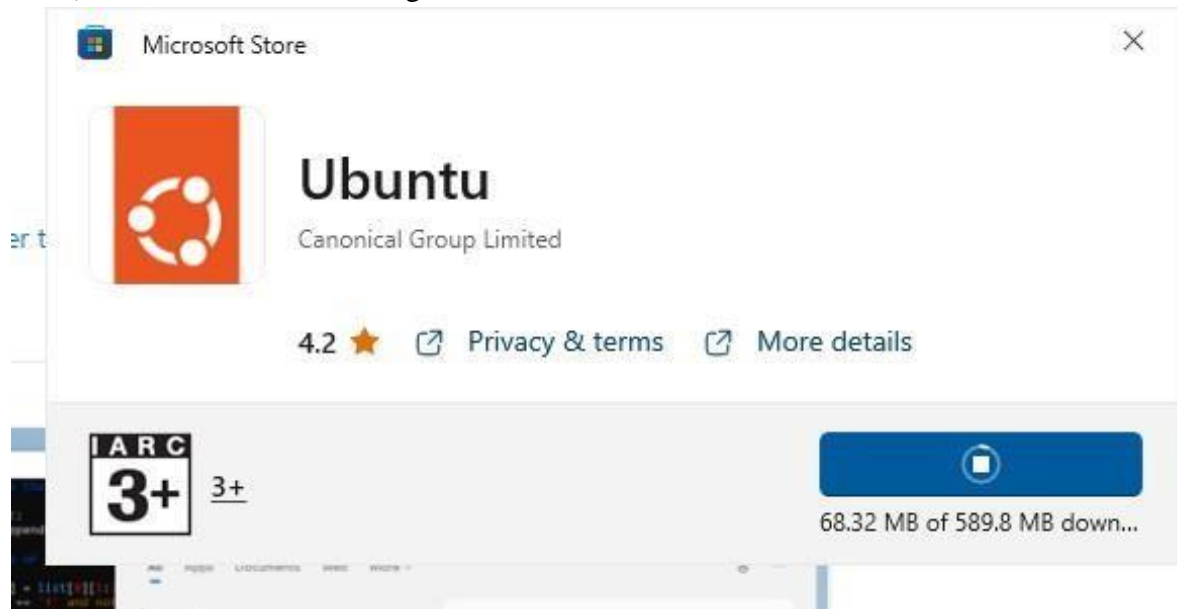


- 3) Click open Microsoft Store

[9PDXGNCFCSCZV?hl=en-in&gl=in&rtc=1](https://apps.microsoft.com/store/detail/ubuntu/9PDXGNCFCSCZV?hl=en-in&gl=in&rtc=1)



- 4) It will start downloading and once downloaded, it will start installation



- 5) Setting up username and password. After installation it will ask for a new username and password.

```
Installing, this may take a few minutes...
Please create a default UNIX user account. The username does not need to match your Windows username.
For more information visit: https://aka.ms/wslusers
Enter new UNIX username: anshul
New password:
Retype new password:
passwd: password updated successfully
Installation successful!
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

Welcome to Ubuntu 24.04.1 LTS (GNU/Linux 5.15.153.1-microsoft-standard-WSL2 x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/pro

System information as of Thu Oct 24 16:14:04 UTC 2024

System load:  0.08          Processes:            46
Usage of /:   0.1% of 1006.85GB Users logged in:       0
Memory usage: 6%           IPv4 address for eth0: 172.21.187.165
Swap usage:   0%

This message is shown once a day. To disable it please create the
/home/anshul/.hushlogin file.
anshul@LAPTOP-0GQIM1QM:~$
```

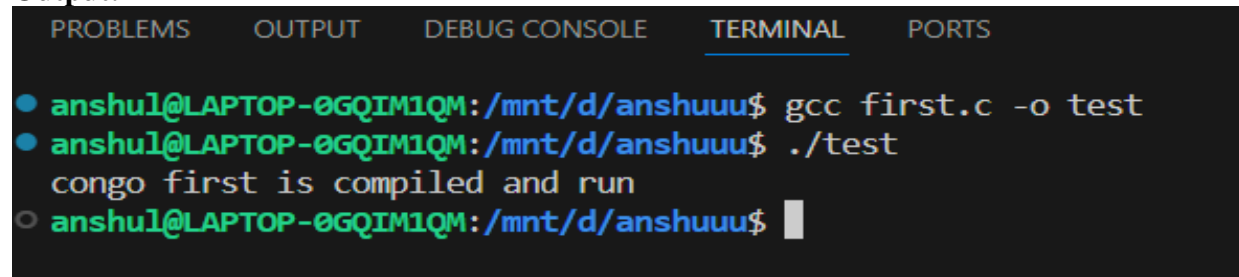
First C program, compile and run using gcc is:

Theory: The provided C code is a simple program that demonstrates the fundamental structure of a C application. It begins with the inclusion of the standard input/output library using `#include <stdio.h>`, which is essential for utilizing functions like `printf`. The `main` function serves as the entry point of the program, where execution begins. Inside this function, the `printf` statement outputs the message "Congratulations First is compiled and run," followed by a newline character to move the cursor to the next line. This output confirms that the program has compiled and run successfully. Finally, the `return 0;` statement indicates the successful completion of the program, with `0` signifying no errors occurred during execution. Overall, this code serves as a basic introduction to C programming, illustrating how to structure a program and perform output operations, making it an ideal starting point for beginners.

Source Code:

```
#include <stdio.h>
int main(){
    printf("Congratulations First is compiled and run\n");
    return 0;
}
```

Output:



The screenshot shows a terminal window with a dark background and light-colored text. At the top, there are five tabs: "PROBLEMS", "OUTPUT", "DEBUG CONSOLE", "TERMINAL" (which is selected and underlined), and "PORTS". Below the tabs, there are three lines of terminal output, each preceded by a colored bullet point (blue, green, and grey respectively). The first line shows the command `gcc first.c -o test` being executed. The second line shows the command `./test` being executed, followed by the output `congo first is compiled and run`. The third line shows the prompt `anshu1@LAPTOP-0GQIM1QM:/mnt/d/anshuuu$` with a cursor.

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
● anshu1@LAPTOP-0GQIM1QM:/mnt/d/anshuuu$ gcc first.c -o test
● anshu1@LAPTOP-0GQIM1QM:/mnt/d/anshuuu$ ./test
  congo first is compiled and run
○ anshu1@LAPTOP-0GQIM1QM:/mnt/d/anshuuu$
```

Learning Outcomes:

PRACTICAL NO. 2

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

Theory: The Ubuntu terminal is a vital component of the Linux operating system, offering a powerful command-line interface for system interaction. Basic commands like `pwd`, `ls`, and `mkdir` form the core of terminal functionality. The `pwd` command displays the current working directory, helping users navigate the filesystem. With `ls`, users can list files and directories, gaining insight into their contents. The `mkdir` command allows for creating new directories, promoting organized file management. Other essential commands include `cd` for changing directories, `cp` for copying files, `mv` for moving or renaming them, and `rm` for removing files or directories. These commands illustrate the efficiency of the command line, where complex tasks can be executed quickly and easily. Mastering these basic commands empowers users to manage their systems effectively, paving the way for more advanced operations and deeper engagement with the Linux environment.

- A. Command to know your current working directory.

```
● anshul@LAPTOP-0GQIM1QM:/mnt/d/anshuuu$ pwd  
/mnt/d/anshuuu
```

- B. List all the files in the current directory.

```
● anshul@LAPTOP-0GQIM1QM:/mnt/d/anshuuu$ ls  
first.c first.exe test
```

- C. List all the files in the order of their file size.

```
● anshul@LAPTOP-0GQIM1QM:/mnt/d/anshuuu$ ls -laShr  
total 56K  
-rwxrwxrwx 1 anshul anshul 100 Oct 24 16:52 first.c  
drwxrwxrwx 1 anshul anshul 4.0K Oct 24 16:31 .  
drwxrwxrwx 1 anshul anshul 4.0K Oct 24 16:53 ..  
-rwxrwxrwx 1 anshul anshul 16K Oct 24 16:53 test  
-rwxrwxrwx 1 anshul anshul 40K Oct 24 16:32 first.exe
```

- D. List only directories in the current folder.

```
anshul@LAPTOP-0GQIM1QM:~$ ls -d */
OSLab/  last/  new/
```

- E. List only files starting with “N” alphabet.

```
anshul@LAPTOP-0GQIM1QM:~$ ls n*
new.c

new:
```

- F. Using help command find the help on ‘man’ command.

```
anshul@LAPTOP-0GQIM1QM:~$ help -m
GNU bash, version 5.1.16(1)-release (x86_64-pc-linux-gnu)
These shell commands are defined internally.  Type 'help' to see this list.
Type 'help name' to find out more about the function 'name'.
Use 'info bash' to find out more about the shell in general.
Use 'man -k' or 'info' to find out more about commands not in this list.

A star (*) next to a name means that the command is disabled.

job_spec [&]                                history [-c] [-d offset] [n] or hi>
(( expression ))                            if COMMANDS; then COMMANDS; [ elif>
. filename [arguments]                      jobs [-lnprs] [jobspec ...] or job>
:                                             kill [-s sigspec | -n signum | -si>
[ arg... ]                                  let arg [arg ...]
```

- G. Display the content of manual pages on 'ls' command.

```
LS(1)                                User Commands                                LS(1)

NAME
    ls - list directory contents

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

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

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

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

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

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

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

- H. Demonstrate the usage of "whatis" command.

```
anshul@LAPTOP-0GQIM1QM:~$ whatis ls
ls (1) - list directory contents
```

- I. Make directory named "OSLab/<yourname>". <yourname> should be the name of the student performing this experiment.

```
● anshul@LAPTOP-0GQIM1QM:/mnt/d$ mkdir OSLab
● anshul@LAPTOP-0GQIM1QM:/mnt/d$ mkdir OSLab/anshul
```

- J. Write command to reach to "yourname" Directory.

```
● anshul@LAPTOP-0GQIM1QM:/mnt/d$ cd OSLab/anshul
● anshul@LAPTOP-0GQIM1QM:/mnt/d/OSLab/anshul$ pwd
/mnt/d/OSLab/anshul
```

- K. Create a txt file named "TodayMsg.txt" and write a greeting message in it.

```
● anshul@LAPTOP-0GQIM1QM:/mnt/d/OSLab/anshul$ echo "HELLOOO THEREE" > todayMsg.txt
● anshul@LAPTOP-0GQIM1QM:/mnt/d/OSLab/anshul$ cat todayMsg.txt
HELLOOO THEREE
```

- L. Copy this file "TodayMsg.txt" to OSLab directory.

```
● anshul@LAPTOP-0GQIM1QM:/mnt/d/OSLab/anshul$ cp todayMsg.txt /home/anshul/OSLab
```

- M. Delete the file “TodaysMsg.txt” from the ‘yourname’ Folder.

```
anshul@LAPTOP-0GQIM1QM:/mnt/d/anshuuu$ rm TodaysMsg.txt
```

- N. Delete the directory ‘yourname’.

```
anshul@LAPTOP-0GQIM1QM:/mnt/d/anshuuu$ cd ..  
anshul@LAPTOP-0GQIM1QM:/mnt/d$ rmdir anshul
```

- O. Create a text file named “Hello.txt” and write a suitable message in it.

```
anshul@LAPTOP-0GQIM1QM:~/OSLab$ echo "Hello there! WELCOME." > Hello.txt  
anshul@LAPTOP-0GQIM1QM:~/OSLab$ cat Hello.txt  
Hello there! WELCOME.
```

- P. Using touch command create files with names mon.txt, tues.txt, and wed.txt.

```
anshul@LAPTOP-0GQIM1QM:~/OSLab$ touch mon.txt  
anshul@LAPTOP-0GQIM1QM:~/OSLab$ touch tues.txt  
anshul@LAPTOP-0GQIM1QM:~/OSLab$ touch wed.txt  
anshul@LAPTOP-0GQIM1QM:~/OSLab$ ls  
Hello.txt  TodaysMsg.txt  mon.txt  tues.txt  wed.txt
```

- Q. Copy these newly created files to a folder named “dupfolder” after creating it.

```
● anshul@LAPTOP-0GQIM1QM:/mnt/d$ sudo mkdir /duplfolder  
● anshul@LAPTOP-0GQIM1QM:/mnt/d$ sudo cp mon.txt /duplfolder  
● anshul@LAPTOP-0GQIM1QM:/mnt/d$ sudo cp tues.txt /duplfolder  
● anshul@LAPTOP-0GQIM1QM:/mnt/d$ sudo cp wed.txt /duplfolder
```

- R. Move Hello.txt to dupfolder.

```
● anshul@LAPTOP-0GQIM1QM:/mnt/d$ sudo mv Hello.txt /duplfolder
```

- S. Count number of words in the Hello.txt file.

```
● anshul@LAPTOP-0GQIM1QM:/mnt/d/OSLab$ wc -w Hello.txt  
3 Hello.txt
```

- T. Create two files with identical content, change one alphabet in one of these and compare them using cmp command.

```
anshul@LAPTOP-0GQIM1QM:~/OSLab/dupfolder$ echo "This is a sample file." > file1.txt  
anshul@LAPTOP-0GQIM1QM:~/OSLab/dupfolder$ echo "This is a Sample file." > file2.txt  
anshul@LAPTOP-0GQIM1QM:~/OSLab/dupfolder$ cmp file1.txt file2.txt  
file1.txt file2.txt differ: byte 11, line 1
```

Learning Outcomes:

PRACTICAL NO. 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: Shell script-based programs are essential tools within the Linux environment, enabling users to automate tasks and enhance productivity. A shell script is essentially a text file containing a series of command-line instructions that the shell interprets and executes. This allows users to perform complex sequences of operations with minimal effort, encapsulating commands that would otherwise need to be typed individually in the terminal. Shell scripts can perform a wide array of tasks, such as file manipulation, system monitoring, and program execution, making them indispensable for system administrators and developers alike.

What sets shell scripting apart is its ability to incorporate control structures like loops, conditionals, and functions, enabling dynamic decision-making and complex logic. This flexibility allows users to tailor scripts to meet specific needs, whether automating backups, managing user accounts, or processing data. Additionally, shell scripts can accept user input and handle errors gracefully, further enhancing their robustness. Moreover, shell scripting fosters a deeper understanding of the operating system's inner workings, as users learn to manipulate the command line effectively. Ultimately, shell scripts are a powerful means of increasing efficiency and reducing repetitive manual work, making them a cornerstone of effective system management in the Linux ecosystem.

Source Code:

A.

```
GNU nano 6.2 swap.sh
#!/bin/bash
# Read two integers
echo "Enter first integer: "
read a
echo "Enter second integer: "
read b
# Swap the integers
temp=$a
a=$b
b=$temp
# Display swapped values
echo "After swapping: "
echo "First integer: $a"
echo "Second integer: $b"
```


B.

```
GNU nano 6.2 backup.sh
#!/bin/bash
# Variables for directory and backup location
SOURCE_DIR="/home/kunal/OSLab"
BACKUP_DIR="/home/kunal/backup"
# Check if the source directory exists
if [ -d "$SOURCE_DIR" ]; then
    # Create backup directory if it doesn't exist
    mkdir -p "$BACKUP_DIR"
    # Copy files to backup directory
    cp -r "$SOURCE_DIR/*" "$BACKUP_DIR/"
    # List all files in the backup directory
    echo "Files successfully copied to backup directory:"
    ls "$BACKUP_DIR"
else
    # Print error message if directory doesn't exist
    echo "Error: The directory $SOURCE_DIR does not exist."
fi
```

C.

```
GNU nano 6.2 prime.sh
#!/bin/bash
echo "Enter a number: " # Read a number from user
read num
is_prime() { # Function to check if a number is prime
    if [ $num -le 1 ]; then
        echo "$num is not a prime number."
        return
    fi
    for ((i=2; i*i<=$num; i++)); do
        if [ $((num % i)) -eq 0 ]; then
            echo "$num is not a prime number."
            return
        fi
    done
    echo "$num is a prime number."
}
is_prime # Call the function
```

D.

```
GNU nano 6.2 greet.sh
#!/bin/bash
hour=$(date +%H) # Get current hour
# Determine the greeting based on the hour
if [ $hour -lt 12 ]; then
    echo "Good Morning, $(whoami)!"
elif [ $hour -lt 18 ]; then
    echo "Good Afternoon, $(whoami)!"
else
    echo "Good Evening, $(whoami)!"
fi
```

Output:

A.

```
anshul@LAPTOP-0GQIM1QM:~/ScriptShell$ nano swap.sh
anshul@LAPTOP-0GQIM1QM:~/ScriptShell$ chmod +x swap.sh
anshul@LAPTOP-0GQIM1QM:~/ScriptShell$ ./swap.sh
Enter first integer:
10
Enter second integer:
20
After swapping:
First integer: 20
Second integer: 10
```

B.

```
anshul@LAPTOP-0GQIM1QM:~$ nano backup.sh
anshul@LAPTOP-0GQIM1QM:~$ chmod +x backup.sh
anshul@LAPTOP-0GQIM1QM:~$ ./backup.sh
Files successfully copied to backup directory:
TodavsMsg.txt backup.sh dupfolder mon.txt tues.txt wed.txt
anshul@LAPTOP-0GQIM1QM:~$ cd backup
anshul@LAPTOP-0GQIM1QM:~/backup$ ls
TodavsMsg.txt backup.sh dupfolder mon.txt tues.txt wed.txt
```

C.

```
anshu@LAPTOP-0GQIM1QM:~$ nano prime.sh
anshu@LAPTOP-0GQIM1QM:~$ chmod +x prime.sh
anshu@LAPTOP-0GQIM1QM:~$ ./prime.sh
Enter a number:
3
3 is a prime number.
anshu@LAPTOP-0GQIM1QM:~$ ./prime.sh
Enter a number:
4
4 is not a prime number.
```

D.

```
anshu@LAPTOP-0GQIM1QM:~$ nano greet.sh
anshu@LAPTOP-0GQIM1QM:~$ ./greet.sh
Good Evening, anshu
```

Learning Outcomes:

PRACTICAL NO. 1