Subject Name: **Operating System**

Subject Code: **22CS005**

Session: **2022-23**

Department: **DCSE**

****

|  |  |  |
| --- | --- | --- |
| **Submitted By:**  Bhuvesh Mittal  2210991450  G5-E |  | **Submitted To:**  Dr. Rajwinder Kaur |

**Index**

|  |  |  |  |
| --- | --- | --- | --- |
| **S. No.** | **Experiments** | **Page Number** | **Remarks** |
| 1 | **Installation**: Configuration & Customizations of Linux  **Introduction to GCC compiler:** Basics of GCC, Compilation of program, Execution of program, Time stamping, Automating the execution using Make file. |  |  |
| 2 | Implement Process concepts using C language by Printing process Id, Execute Linux command as sub process, Creating and executing process using fork and exec system calls. |  |  |
| 3 | Implement FCFS, SJF, priority scheduling, and RR scheduling algorithms in C language. |  |  |
| 4 | Implement the basic and user status commands like: su, sudo, man, help, history, who, whoami, id, uname, uptime, free, tty, cal, date, hostname, reboot, clear |  |  |
| 5 | Implement deadlock in C by using shared variable. |  |  |
| 6 | File system: Introduction to File system, File system Architecture and File Types. |  |  |
| 7 | Implement the commands that is used for Creating and Manipulating files: cat, cp, mv, rm, ls and its options, touch and their options, which is, where is, what is |  |  |
| 8 | Implement Directory oriented commands: cd, pwd, mkdir, rmdir |  |  |
| 9 | Implement File system commands: Comparing Files using diff, cmp, comm |  |  |

**EXPERIMENT 1**

PART 1:

**Installation**: Configuration & Customizations of Linux

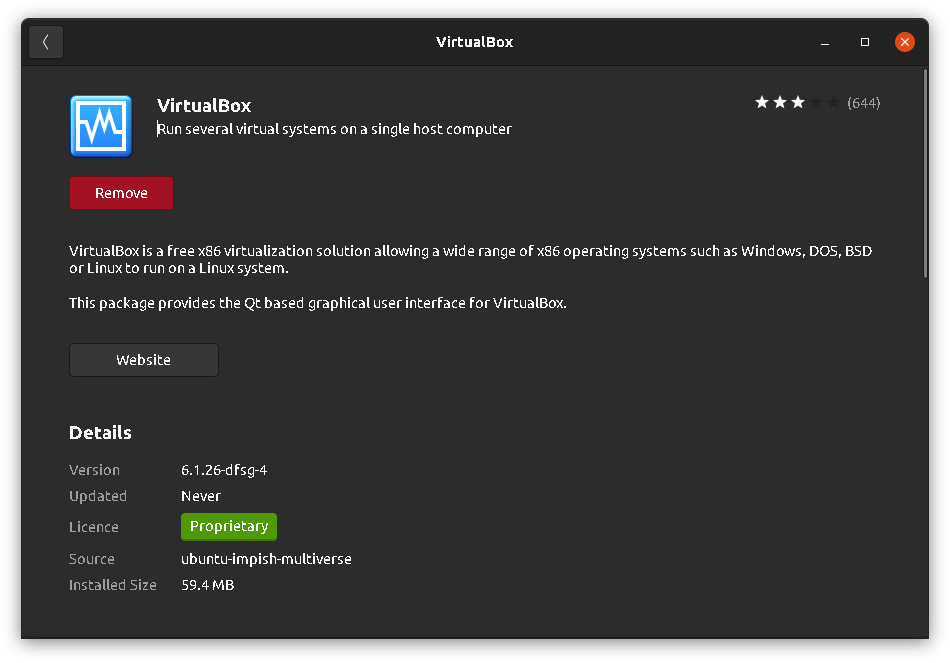
First Download an Ubuntu Image

• You can download an Ubuntu image

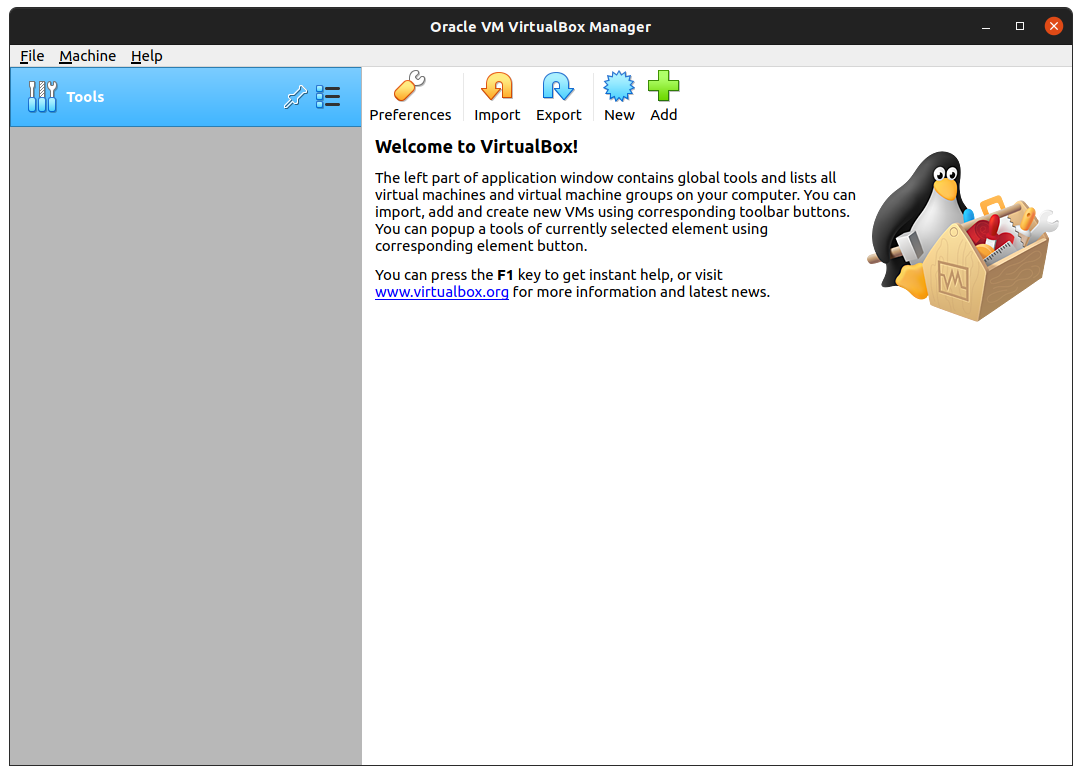
• https://ubuntu.com/download/desktop • Make sure to save it to a memorable location on your PC! For this tutorial, we will use the Ubuntu 20.04 LTS release.

• On Mac OS or Windows, you can download VirtualBox from the downloads page <https://www.virtualbox.org/wiki/Downloads>

• This page also includes instructions to download VirtualBox for Linux. However, on Ubuntu, you can find VirtualBox by simply searching for it in the Ubuntu Software app.



* Once you have completed the installation, go ahead and run VirtualBox.

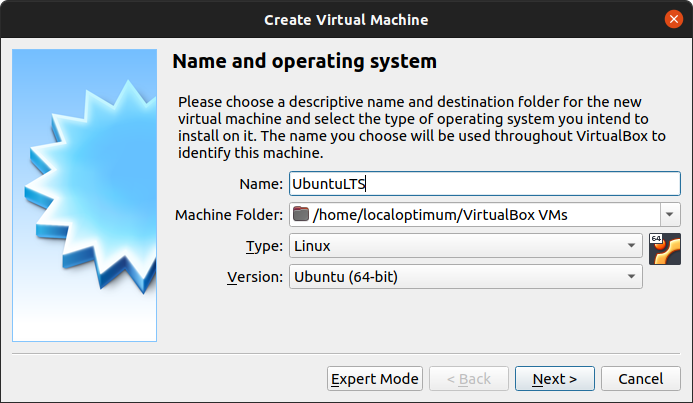


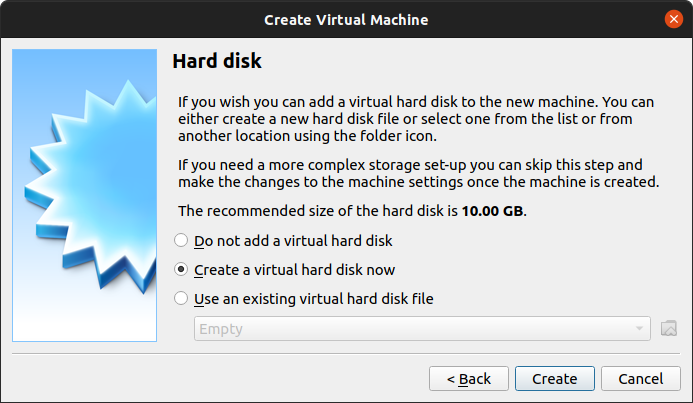
Step 2. Create a new virtual machine

* Click New to create a new virtual machine. Fill in the appropriate details
* Name: If you include the word Ubuntu in your name the Type and Version will

auto-update.

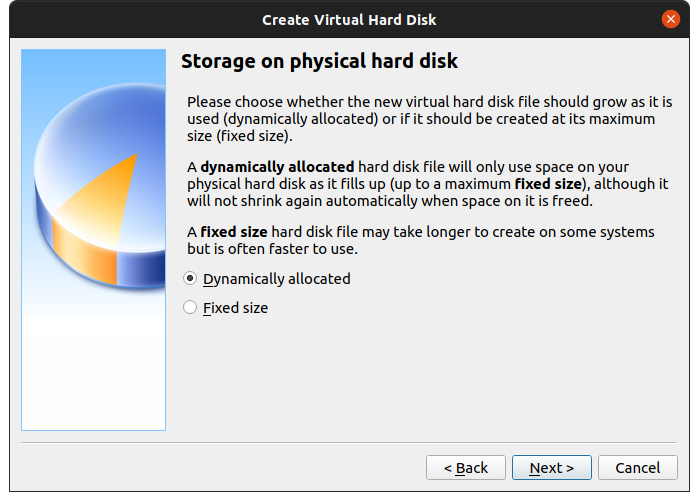
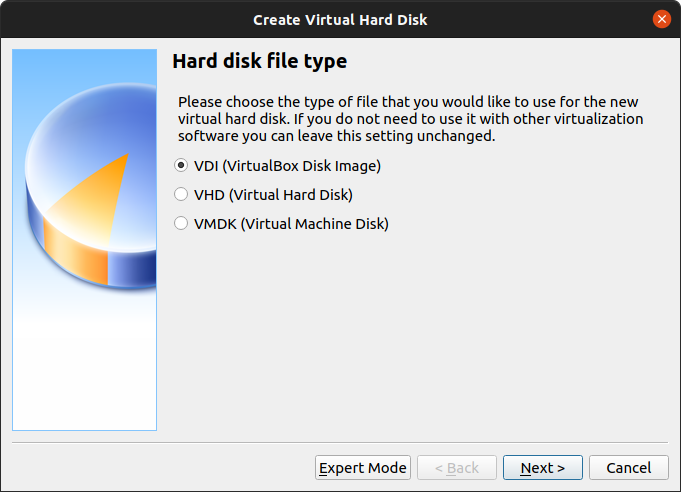
* Machine Folder: This is where your virtual machines will be stored so you can resume working on them whenever you like.
* Type: Linux
* Version: Ubuntu (64-bit)



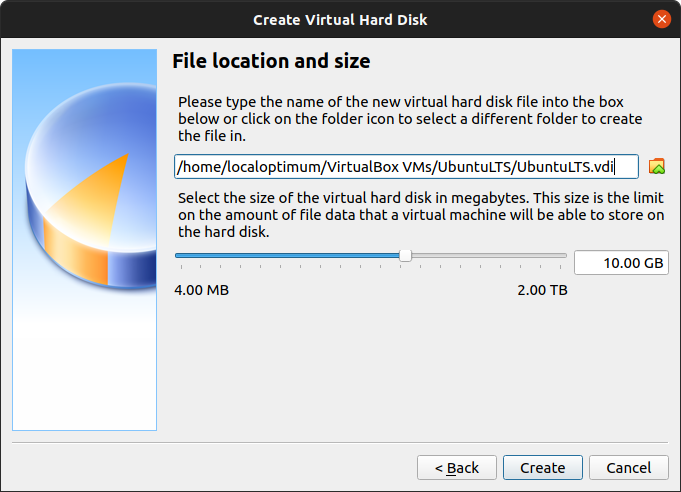


Then you can choose whether the hard disk is dynamically allocated (up to the limit we will set on the next screen), filling up as the VM requires it. Otherwise, we can tell it to allocate the full amount of memory right from the start. This will improve performance but may take up unnecessary space. We’ll leave it as dynamically allocated for this installation.

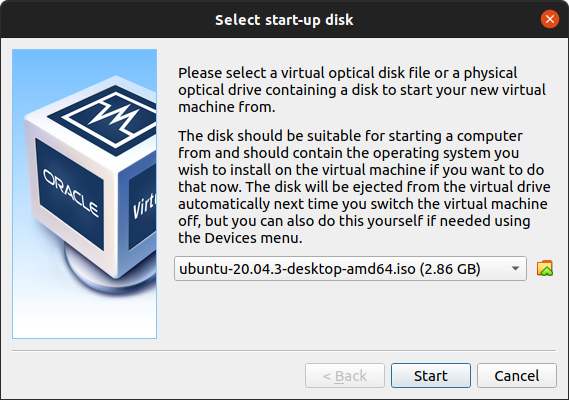
**,**

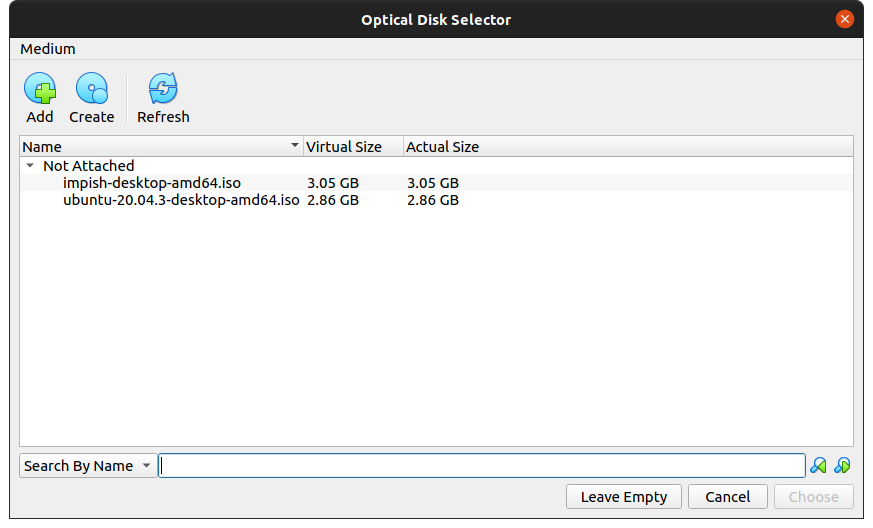


* Finally, you can set the maximum amount of memory your VM can access.



* Click Start to launch the virtual machine. You will be prompted to select the start-up disk. Use the file icon to open the Optical disc selector and click Add to find your .iso file

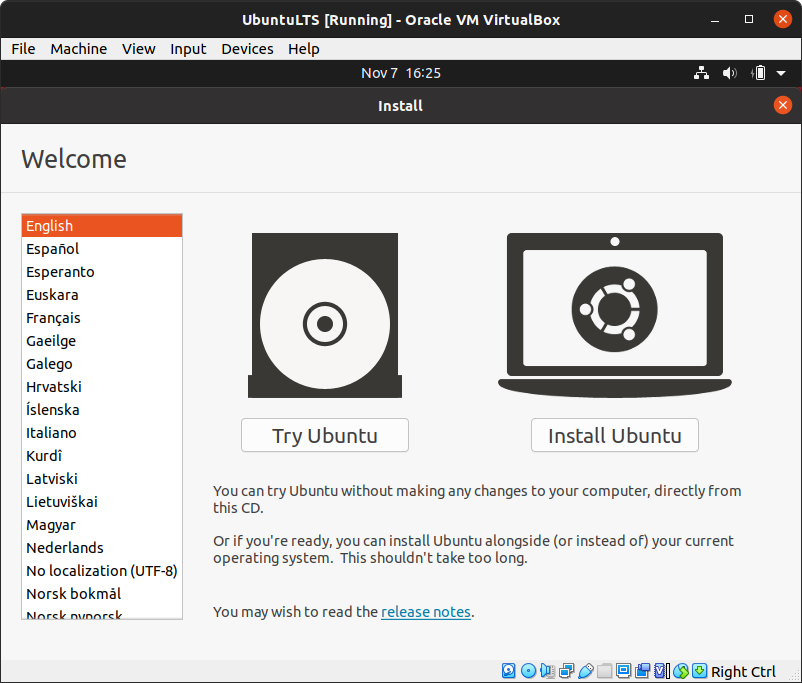




Choose the disc image you want to use, then click Start on the start-up disc window.

Note: If you close this window before selecting an image you can still do so from the Devices menu at the top of the VM window. Select Devices > Optical Drives > Choose/Create a disc image…

* Ubuntu desktop should now boot and display the installation menu.



Now follow instructions to complete installation.

PART 2:

**Introduction to GCC compiler:** Basics of GCC, Compilation of program, Execution of program, Time stamping, Automating the execution using Make file.

**Major features of GCC**

* First of all, GCC is a portable compiler—it runs on most platforms available today.
* GCC is not only a native compiler—it can also cross-compile any program, producing executable ﬁles for different system.
* GCC has a modular design, allowing support for new languages.
* Most importantly, GCC is free software.

**Compiling a C program**

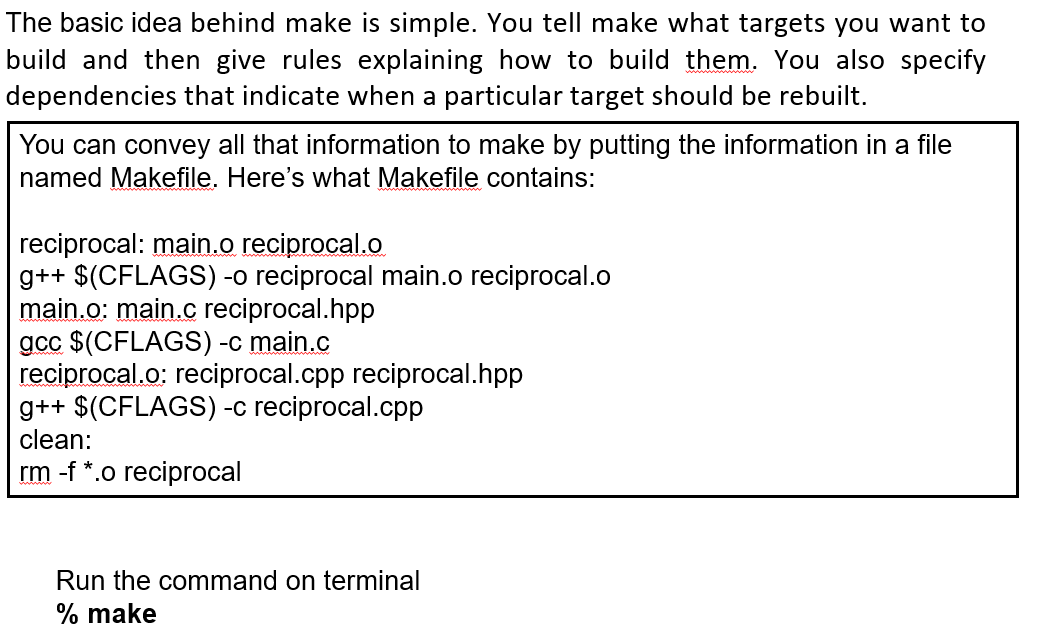
* There are two ways of compiling a C program
* 1). $ gcc bad.c // compiling the C program.

$ ./a.out // executing the object file.

* 2). $ gcc bad.c -o bad // compilation with different object file name.

$. /bad // executing the object file.

**Make File**



**Experiment 2**

Implement Process concepts using C language by Printing process Id, Execute Linux command as sub process, Creating and executing process using fork and exec system calls.

**Program 1 :**To write some data on the standard output device.

Code :

#include<stdio.h>

#include<unistd.h>

#include<sys/types.h>

#include<sys/stat.h>

#include<sys/wait.h>

#include<fcntl.h>

#include<unistd.h>

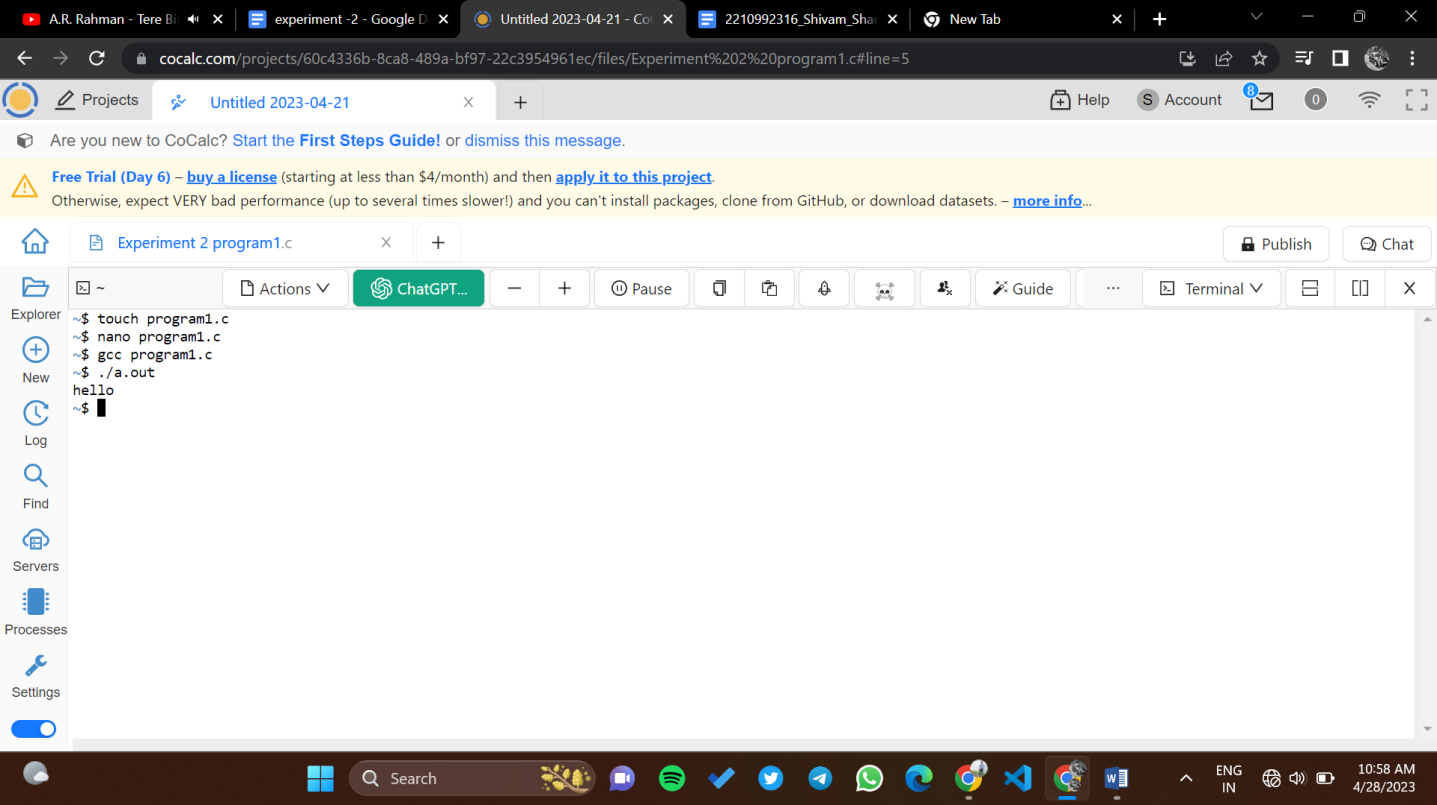
int main()

{

write(1,"hello\n",6);

}

**Output :**



**Program 2:** To read data from the standard input device and write it on the screen.

Code :

#include<stdio.h>

#include<unistd.h>

#include<sys/types.h>

#include<sys/stat.h>

#include<sys/wait.h>

#include<fcntl.h>

#include<unistd.h>

int main()

{

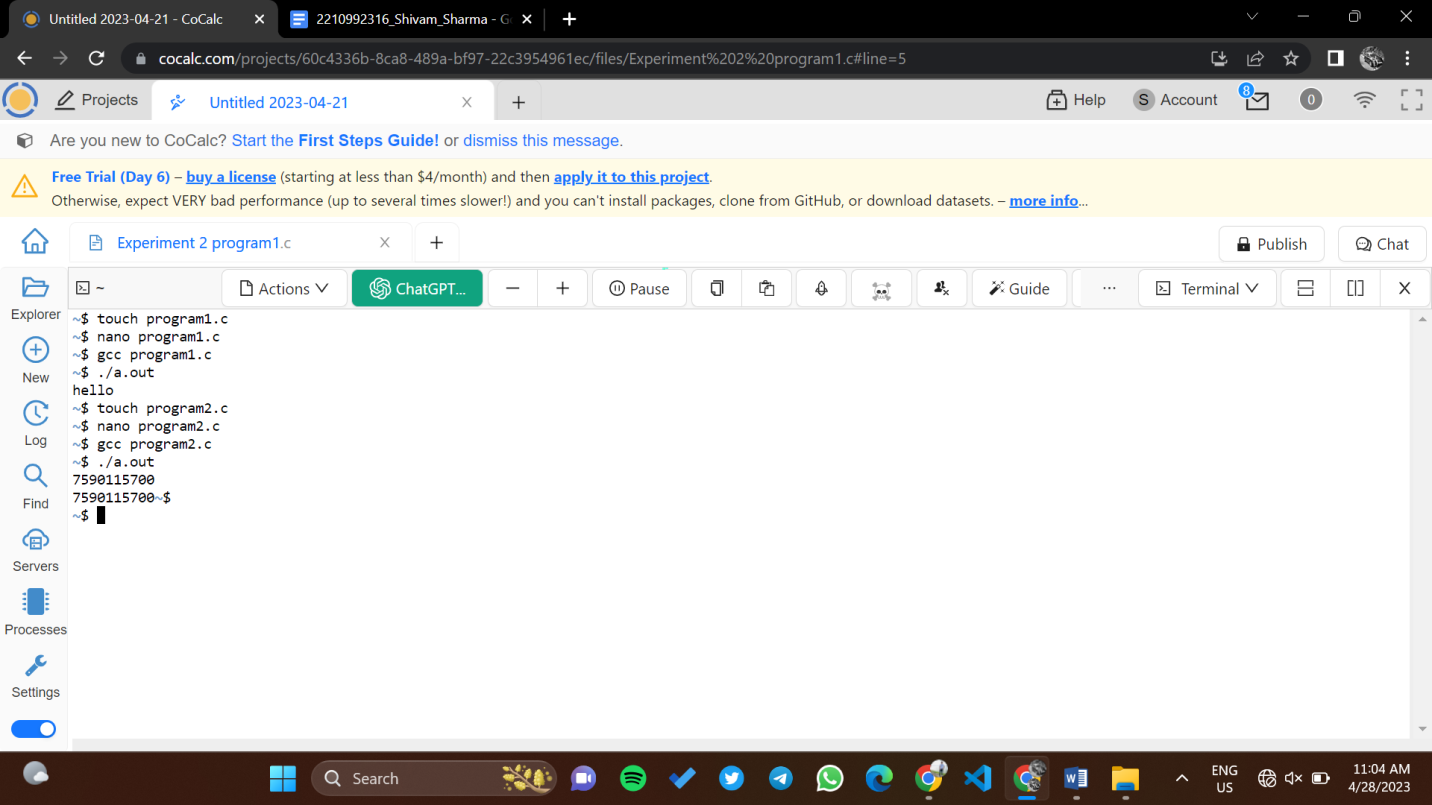
char buff[20];

read(0,buff,10);

write(1,buff,10);

}

**Output :**

****

**Program 3:** Write a program using open () system call to read the first 10 characters of an existing file “test.txt” and print them on screen.

Code :

#include<stdio.h>

#include<unistd.h>

#include<sys/types.h>

#include<sys/stat.h>

#include<fcntl.h>

int main()

{

int n,fd;

char buff[50];

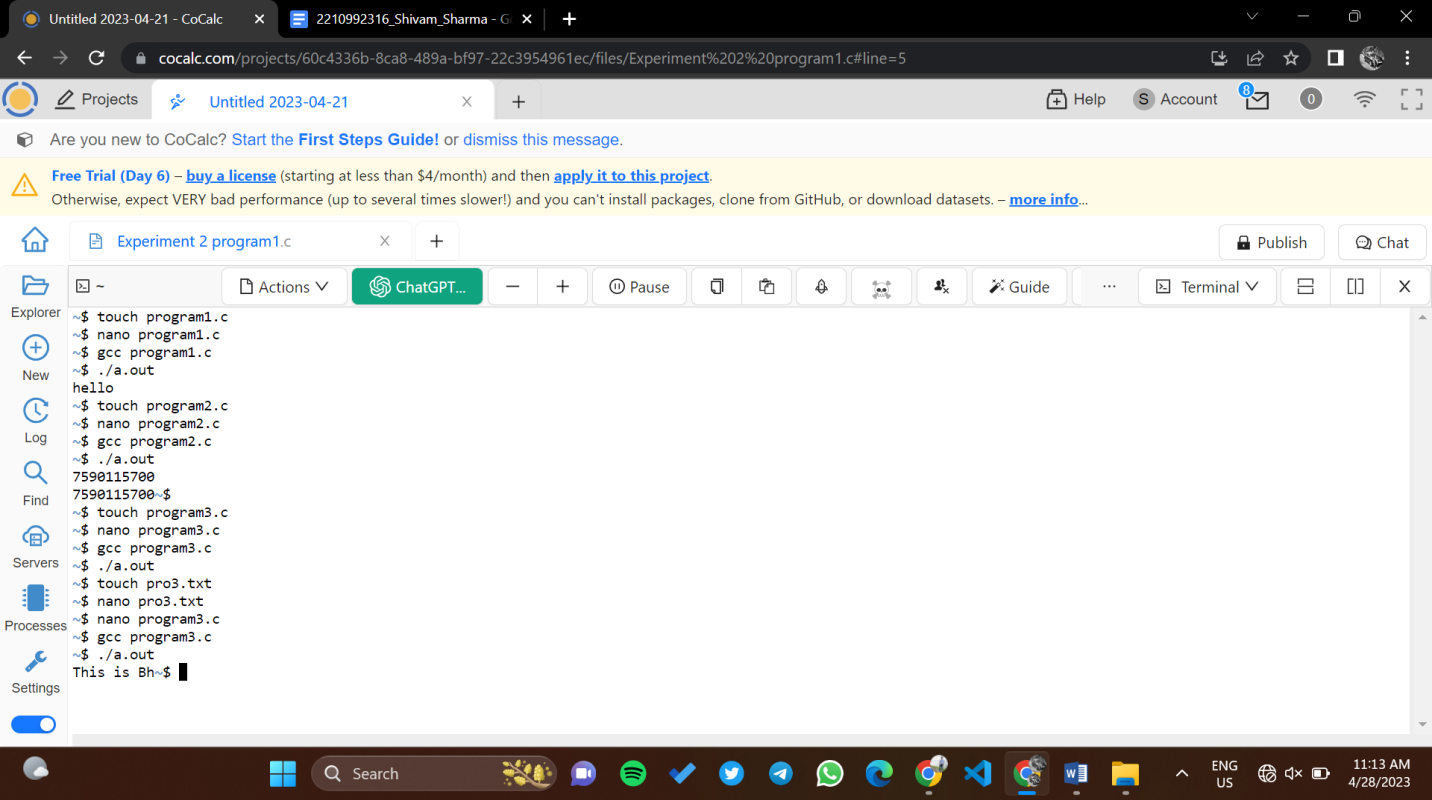
fd=open("test.txt",O\_RDONLY);

n=read(fd,buff,10);

write(1,buff,n);

}

**Output:**



**Program 4:** To read 10 characters from file “test.txt” and write them into non-existing file “towrite.txt”.

Code :

#include<unistd.h>

#include<sys/types.h>

#include<sys/stat.h>

#include<fcntl.h>

int main()

{

int n,fd,fd1;

char buff[50];

fd=open("test.txt",O\_RDONLY);

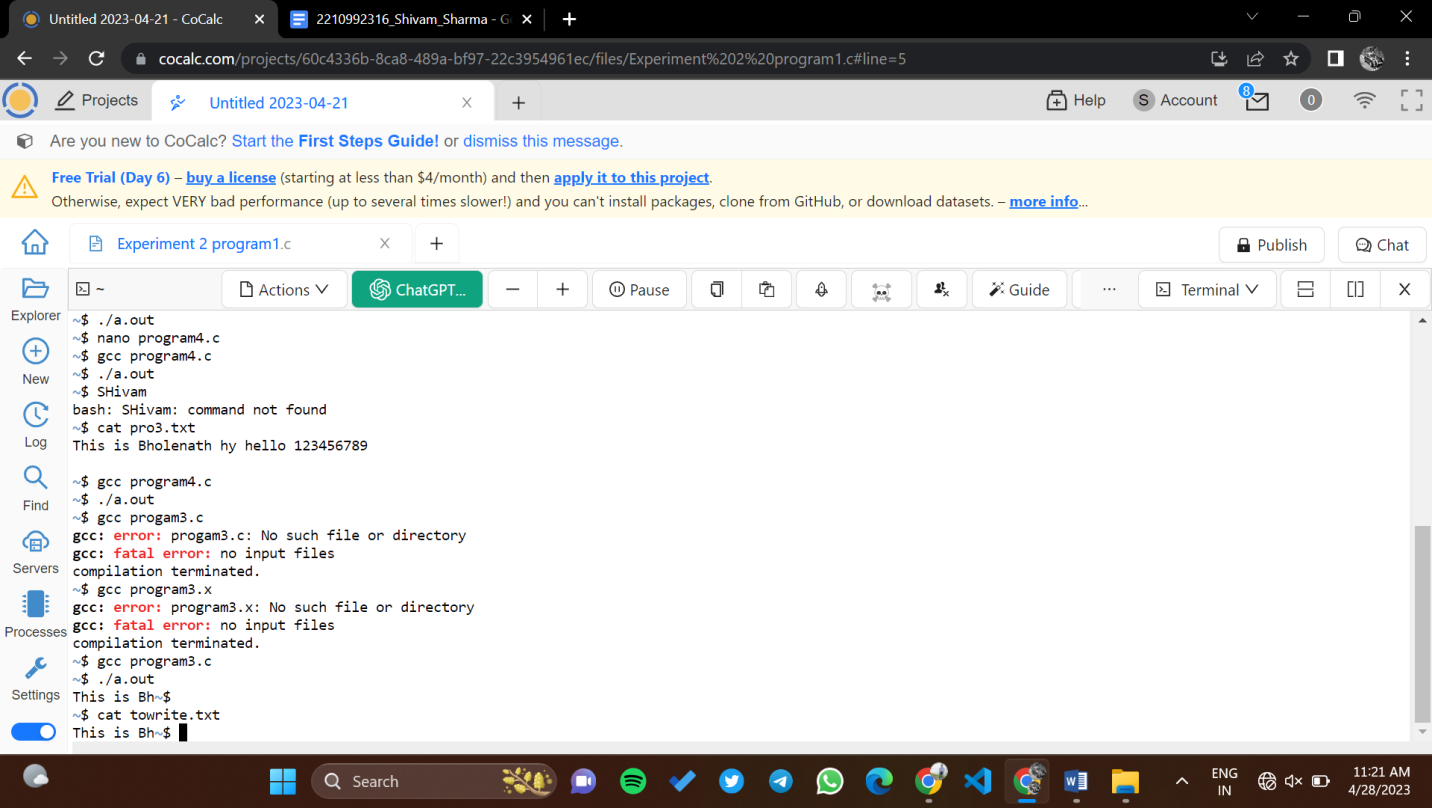
n=read(fd,buff,10);

fd1=open("towrite.txt",O\_WRONLY|O\_CREAT,0642);

write(fd1,buff,n);

}

**Output :**

****

**Program 5:** fork () Command

#include<stdio.h>

#include<unistd.h>

#include<sys/types.h>

#include<sys/stat.h>

#include<sys/wait.h>

#include<fcntl.h>

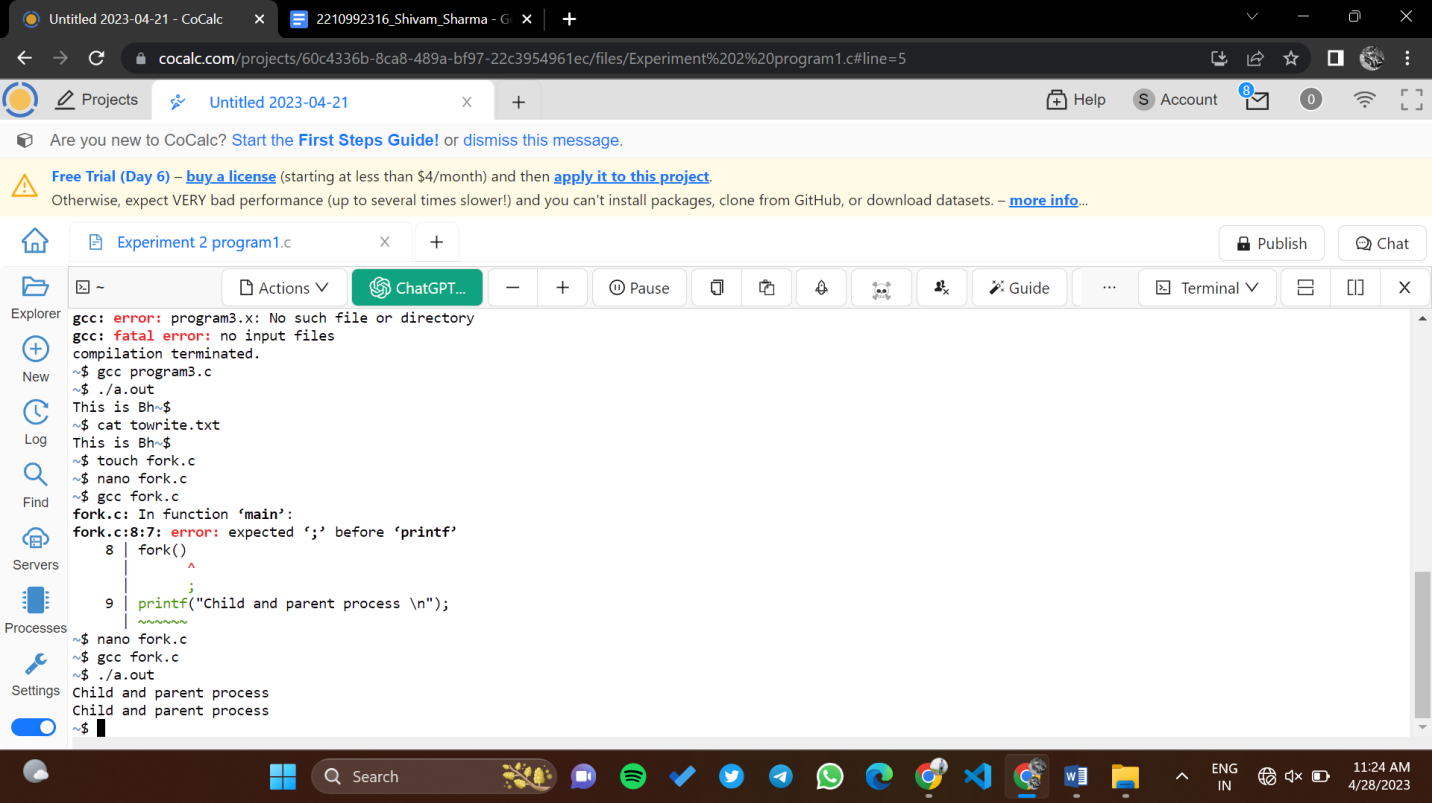
int main(){

fork();

printf("Child and parent process \n");

}

**Output :**

****

**Program 6:** Printing Parent and child process id through if and else block.

Code :

#include<stdio.h>

#include<unistd.h>

#include<sys/types.h>

#include<sys/stat.h>

#include<sys/wait.h>

#include<fcntl.h>

void main(){

pid\_t q;

q=fork();

if(q==-1){

printf("error");}

if(q==0){

printf("Child processid=%d \n",getpid());

printf("parent processid=%d\n",getppid());}

else{

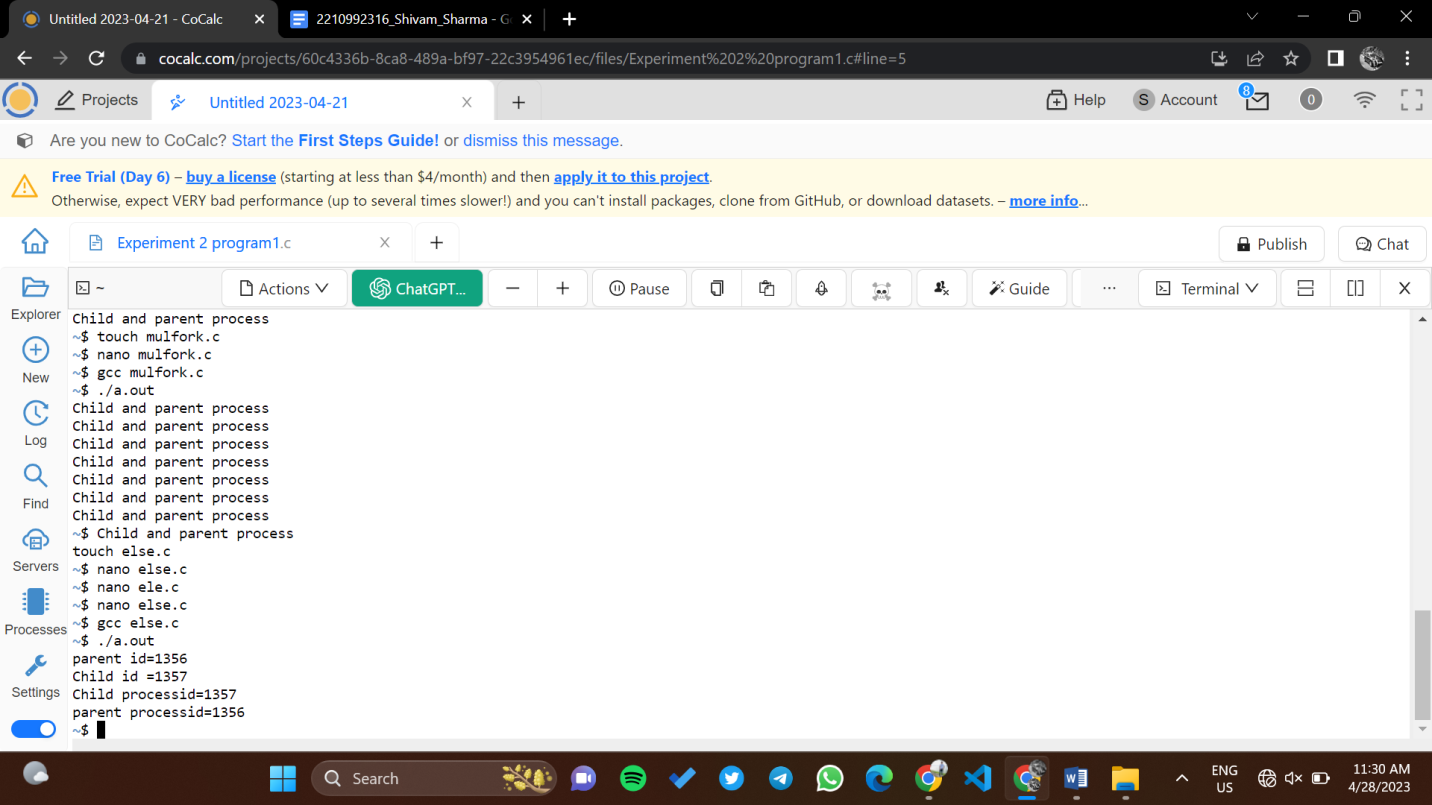
printf("parent id=%d \n",getpid());

printf("Child id =%d \n" ,q);

}

}

Output :



**Program 7 :** Printing Parent and child process id through keeping else block on wait().

Code :

#include<stdio.h>

#include<unistd.h>

#include<sys/types.h>

#include<sys/stat.h>

#include<sys/wait.h>

#include<fcntl.h>

void main(){

pid\_t q;

q=fork();

if(q==-1){

printf("error");}

if(q==0){

printf("Child processid=%d \n",getpid());

printf("parent processid=%d\n",getppid());}

else{

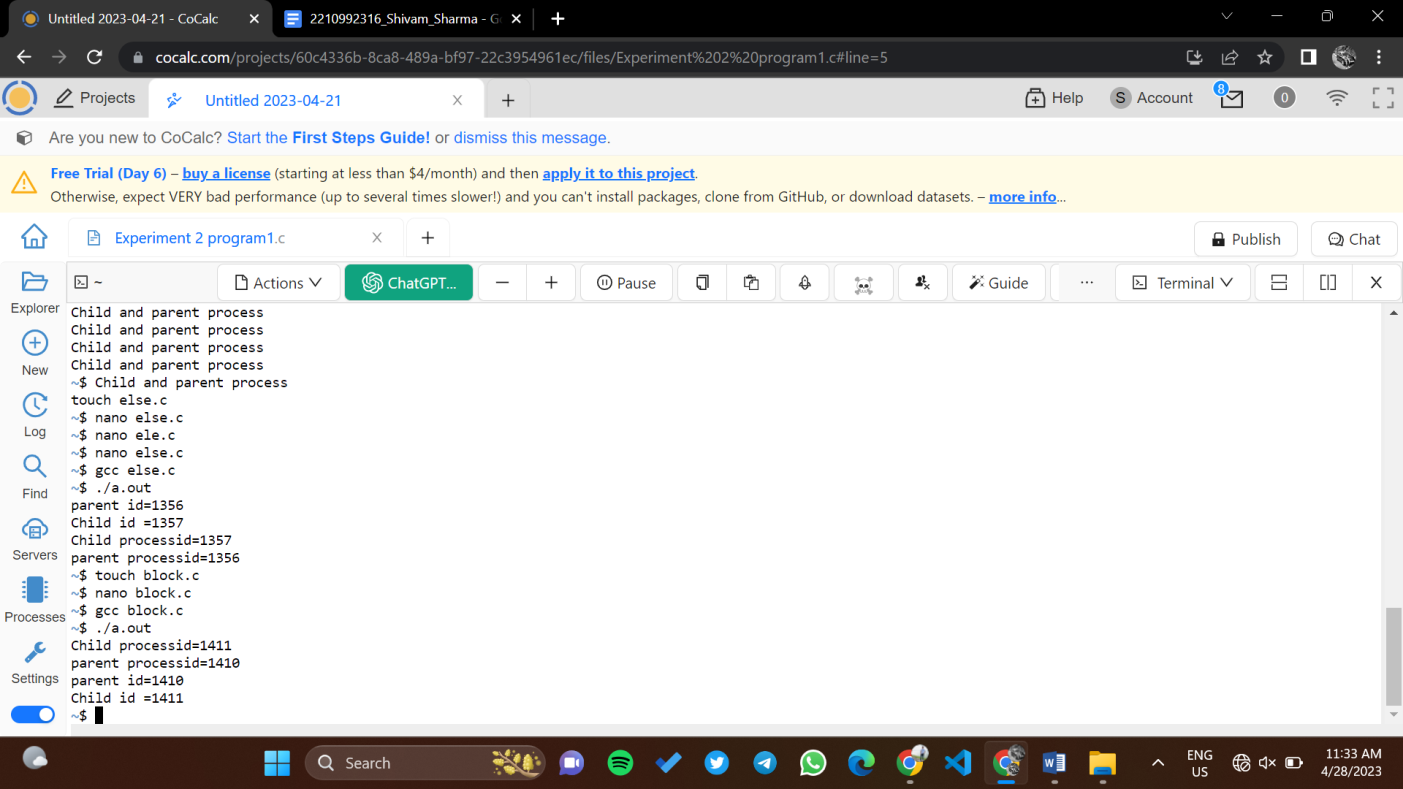
wait(NULL);

printf("parent id=%d \n",getpid());

printf("Child id =%d \n" ,q);

}

}

Output : 

**Program 8:** Printing Parent and child process id through keeping if block on sleep().

Code :

#include<stdio.h>

#include<unistd.h>

#include<sys/types.h>

#include<sys/stat.h>

#include<sys/wait.h>

#include<fcntl.h>

void main(){

pid\_t q;

q=fork();

if(q==-1){

printf("error");}

if(q==0){

sleep(10);

printf("Child processid=%d \n",getpid());

printf("parent processid=%d\n",getppid());}

else{

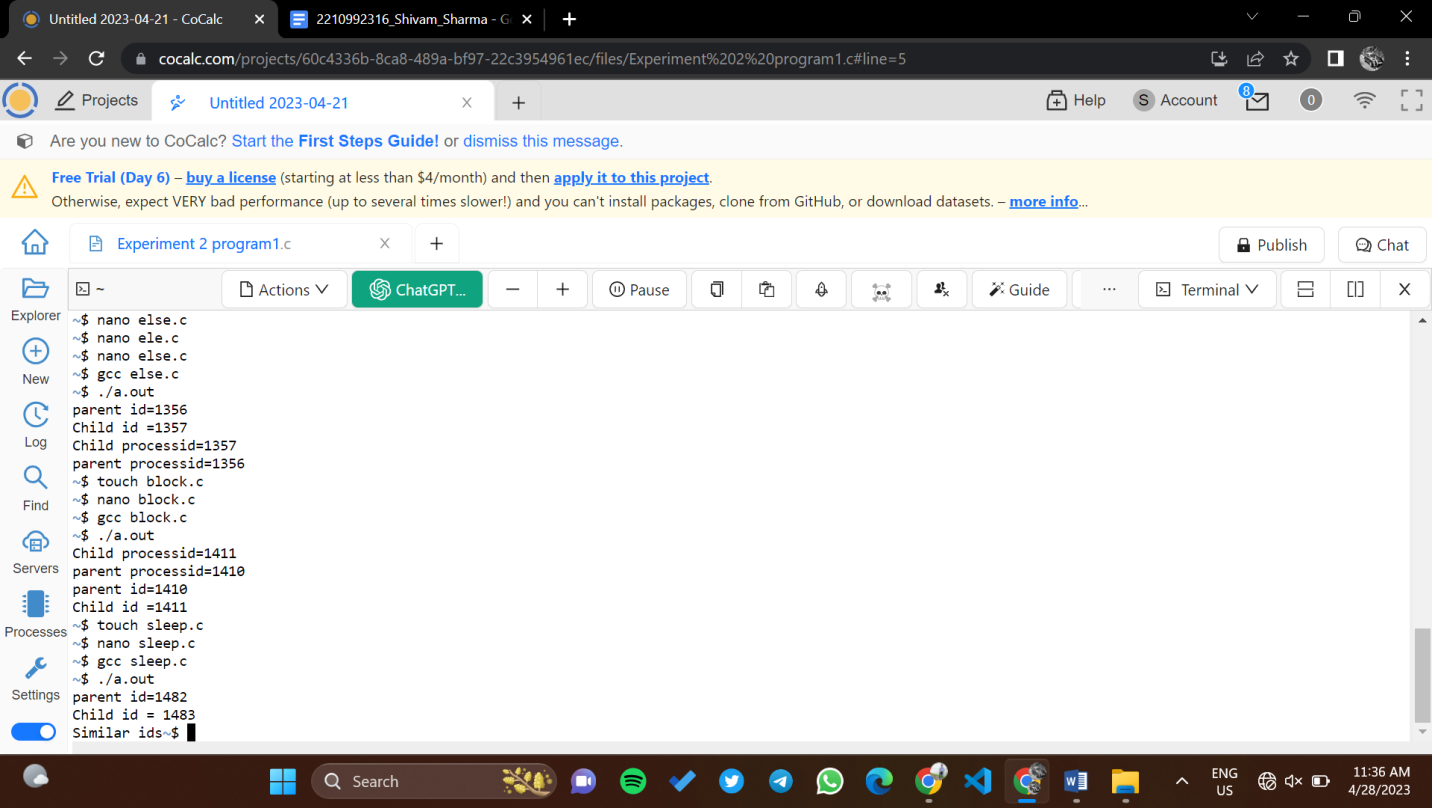
printf("parent id=%d \n",getpid());

printf("Child id = %d \n",q);}

printf("Similar ids");

}

Output :



**Program 9 :** Creating and executing process using fork and exec system calls.

Code :

#include<stdio.h>

#include<unistd.h>

#include<sys/types.h>

#include<sys/stat.h>

#include<sys/wait.h>

#include<fcntl.h>

int main()

{

printf("I am in execl.c \n");

printf("PID of execl is = %d \n" ,getpid());

char \*args[]={"./Hello",NULL};

execv(args[0],args);

printf("Coming back to main program \n");

return 0;

}

Output :

