

Tên	Đinh Phương My
MSSV	52100703
Nhóm thực hành	N101

BÀI BÁO CÁO THỰC HÀNH LAB 3

Lab 3.2

Câu 1: Viết chương trình để truyền đối vào số nguyên dương n vào và

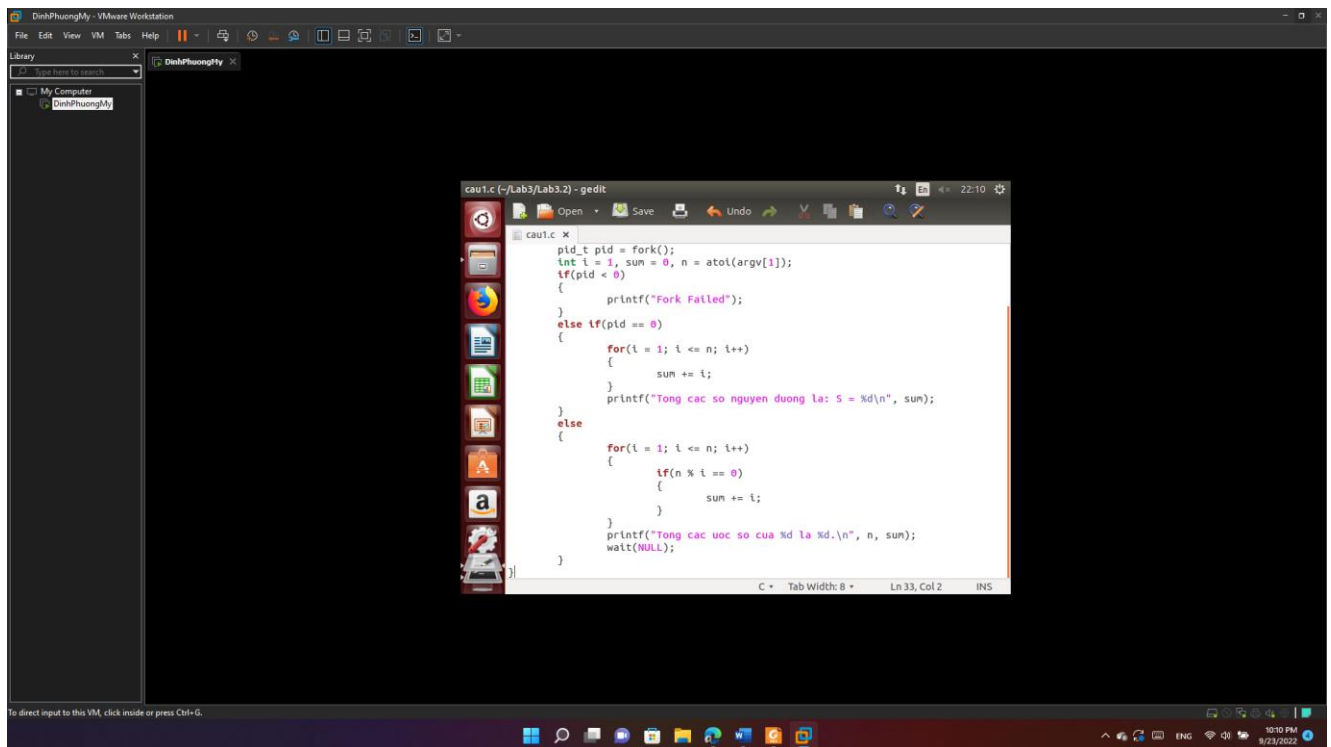
- Tiến trình cha tiếp tục tính rồi xuất ra tổng $S = 1 + 2 + \dots + n$
- Đồng thời tạo một tiến trình con tính tổng các ước số của n và in ra màn hình.

Code:

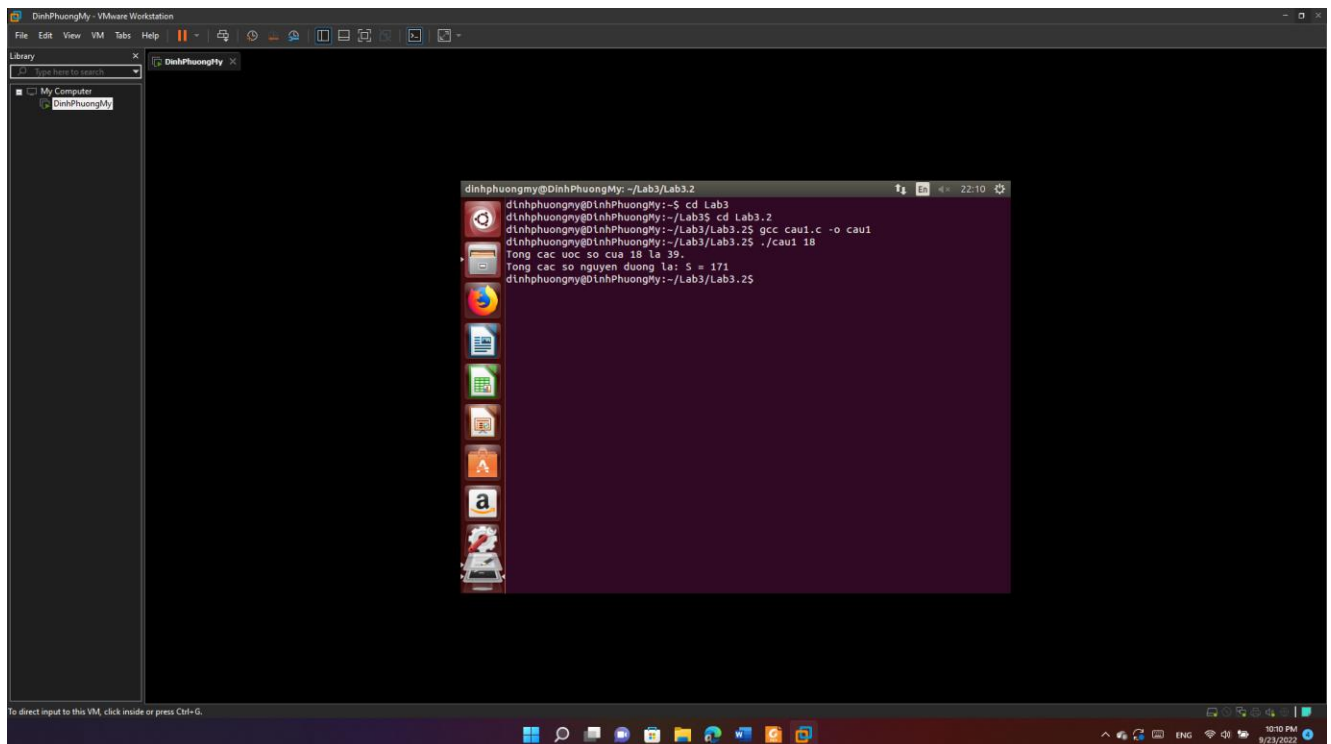
```

caul.c (-/Lab3/Lab3.2) - gedit
#include <stdio.h>
#include <stdlib.h>
#include <sys/wait.h>
#include <unistd.h>
void main(int argc, char **argv)
{
    pid_t pid = fork();
    int i = 1, sum = 0, n = atoi(argv[1]);
    if(pid < 0)
    {
        printf("Fork Failed");
    }
    else if(pid == 0)
    {
        for(i = 1; i <= n; i++)
        {
            sum += i;
        }
        printf("Tong cac so nguyen duong la: S = %d\n", sum);
    }
    else
    {
        for(i = 1; i <= n; i++)
        {
            if(n % i == 0)
            {
                sum += i;
            }
        }
        printf("Tong cac so nguyen duong la: S = %d\n", sum);
    }
}

```

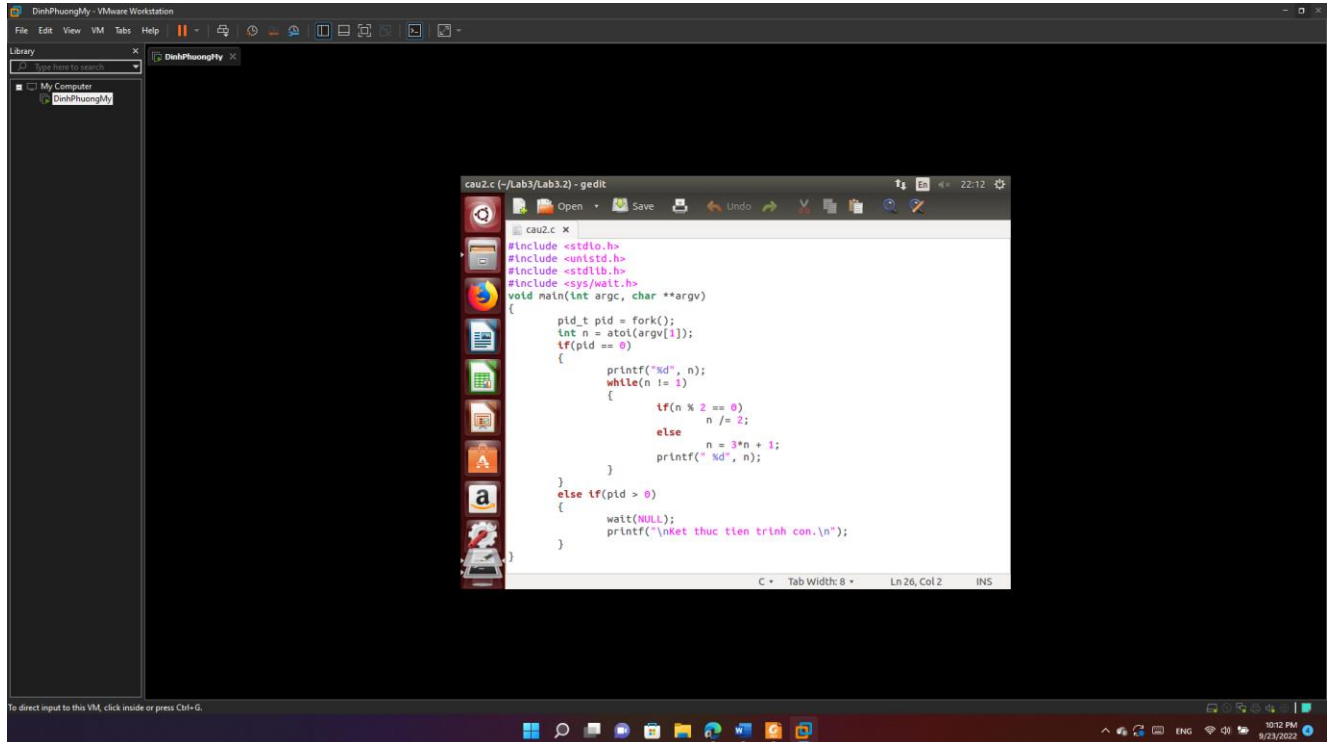


Run:



Câu 2: Viết một chương trình nhận số nguyên dương n vào thông qua đối số, kiểm tra tính đúng của giá trị này. Tạo ra một tiến trình con để tính và in ra dãy số, trong lúc tiến trình cha chờ tiến trình con hoàn thành thông qua lời gọi `wait()`.

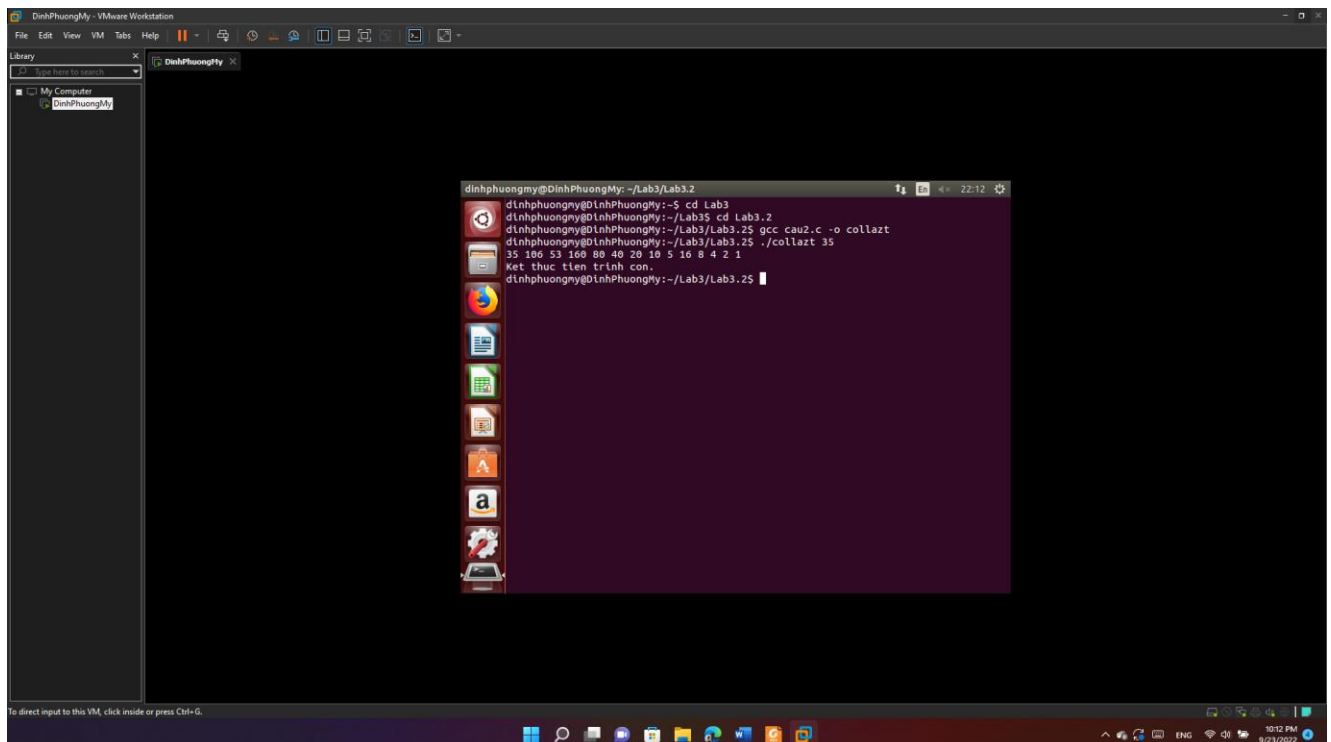
Code:

A screenshot of a VMware Workstation window showing a Linux virtual machine named 'DinhPhuongMy'. Inside the VM, a gedit text editor is open with a file named 'cau2.c'. The code in the editor is a C program that uses `fork()` to create a child process. The parent process prints the input number n and enters a `while` loop that continues until n is 1. Inside the loop, it checks if n is even or odd and updates n accordingly. The child process, created with `wait(NULL)`, prints the sequence of numbers generated by the parent process. The terminal output shows the sequence: 35 166 53 166 88 40 20 10 5 16 8 4 2 1.

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

void main(int argc, char **argv)
{
    pid_t pld = fork();
    int n = atoi(argv[1]);
    if(pld == 0)
    {
        printf("N", n);
        while(n != 1)
        {
            if(n % 2 == 0)
                n /= 2;
            else
                n = 3*n + 1;
            printf(" ", n);
        }
    }
    else if(pld > 0)
    {
        wait(NULL);
        printf("\nket thuc tien trinh con.\n");
    }
}
```

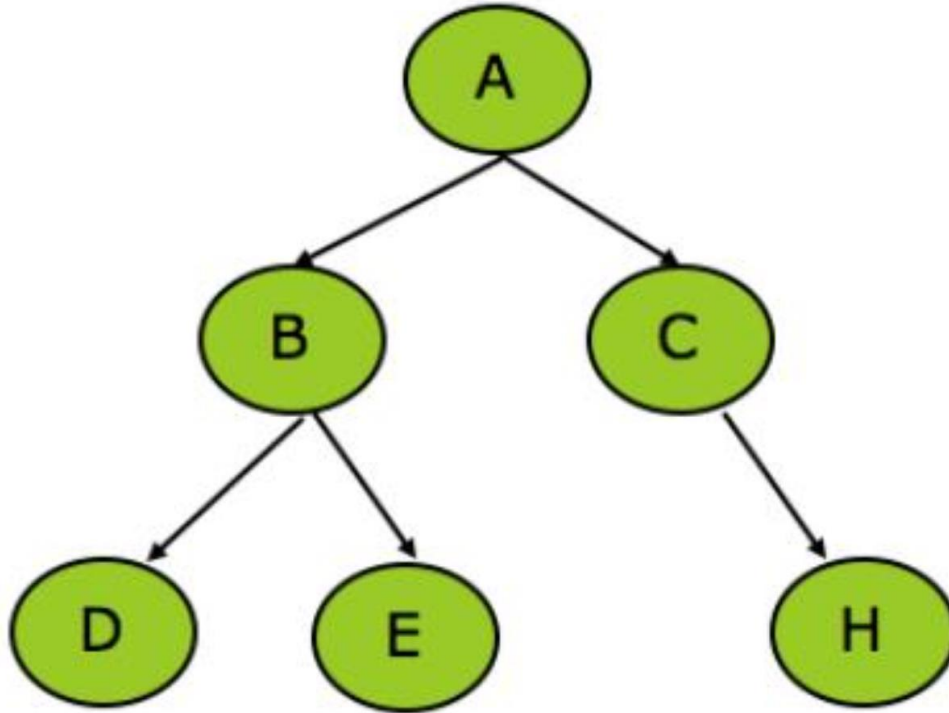
Run:

A screenshot of the same VMware Workstation window, but now showing a terminal window. The terminal displays the commands used to compile and run the program: `cd Lab3`, `cd Lab3.2`, `gcc cau2.c -o collatz`, and `./collatz 35`. The output of the program is shown: the sequence of numbers 35 166 53 166 88 40 20 10 5 16 8 4 2 1, followed by the message 'ket thuc tien trinh con.'.

```
dinhphuongmy@DinhPhuongMy:~/Lab3/Lab3.2$ cd Lab3
dinhphuongmy@DinhPhuongMy:~/Lab3$ cd Lab3.2
dinhphuongmy@DinhPhuongMy:~/Lab3/Lab3.2$ gcc cau2.c -o collatz
dinhphuongmy@DinhPhuongMy:~/Lab3/Lab3.2$ ./collatz 35
35 166 53 166 88 40 20 10 5 16 8 4 2 1
ket thuc tien trinh con.
```

Lab 3.3

Câu 1: Tạo ra cây tiến trình như sau, với mỗi tiến trình con, in ra ID của nó và ID của tiến trình cha.

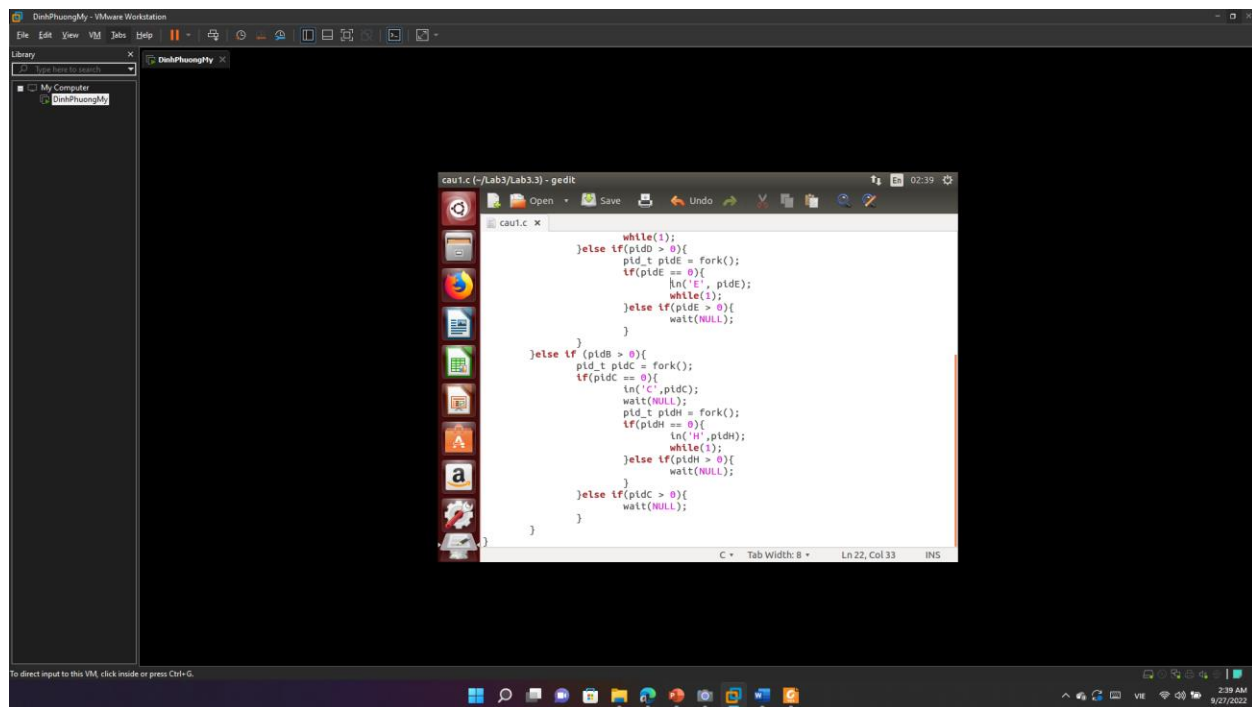


Code:

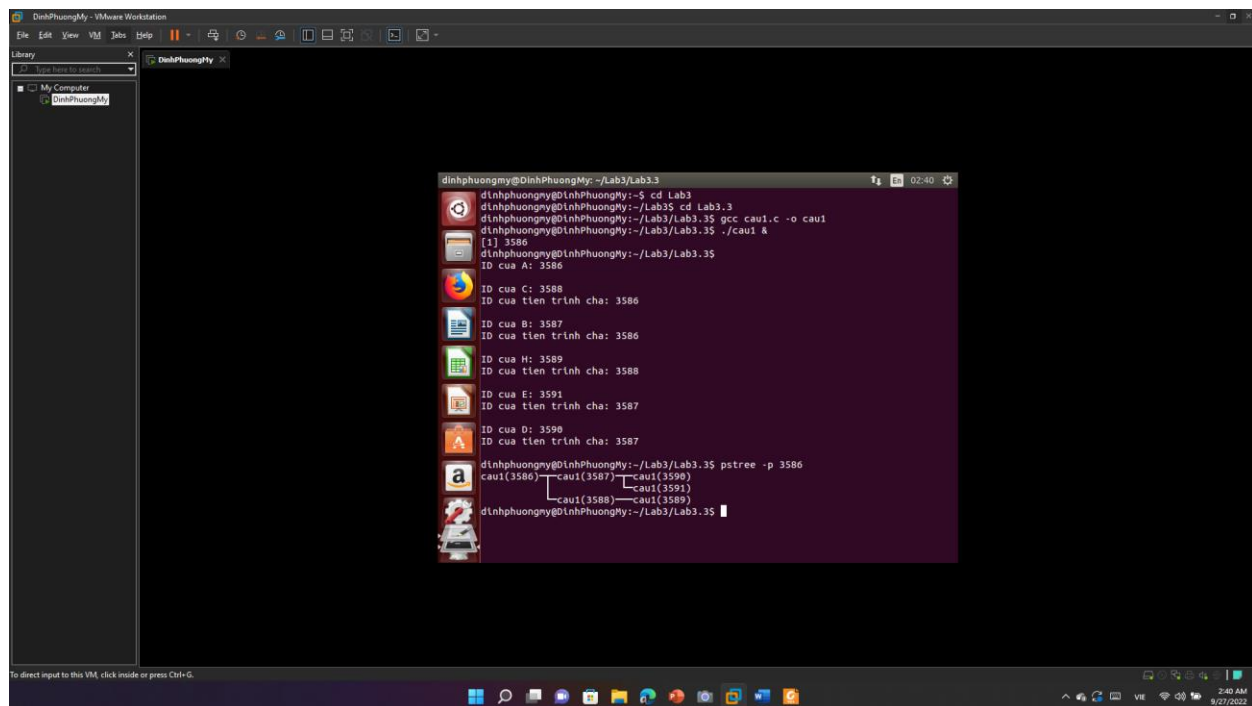
```
cau1.c (-/Lab3/Lab3.3) - gedit
#include <stdio.h>
#include <unistd.h>
#include <sys/wait.h>

void in(char t, pid_t pid){
    printf("\nID của %c: %d\n", t, getpid());
    printf("ID của tlen trình cha: %d\n", getppid());
}

void main(){
    printf("\nID của %c: %d\n", 'A', getpid());
    pid_t pidB = fork();
    if(pidB == 0){
        in('B', pidB);
        pid_t pidD = fork();
        if(pidD == 0){
            in('D', pidD);
        }
        while(1);
    }
    else if(pidB > 0){
        pid_t pidC = fork();
        if(pidC == 0){
            in('C', pidC);
        }
        else if(pidC > 0){
            wait(NULL);
        }
    }
}
```



Run:

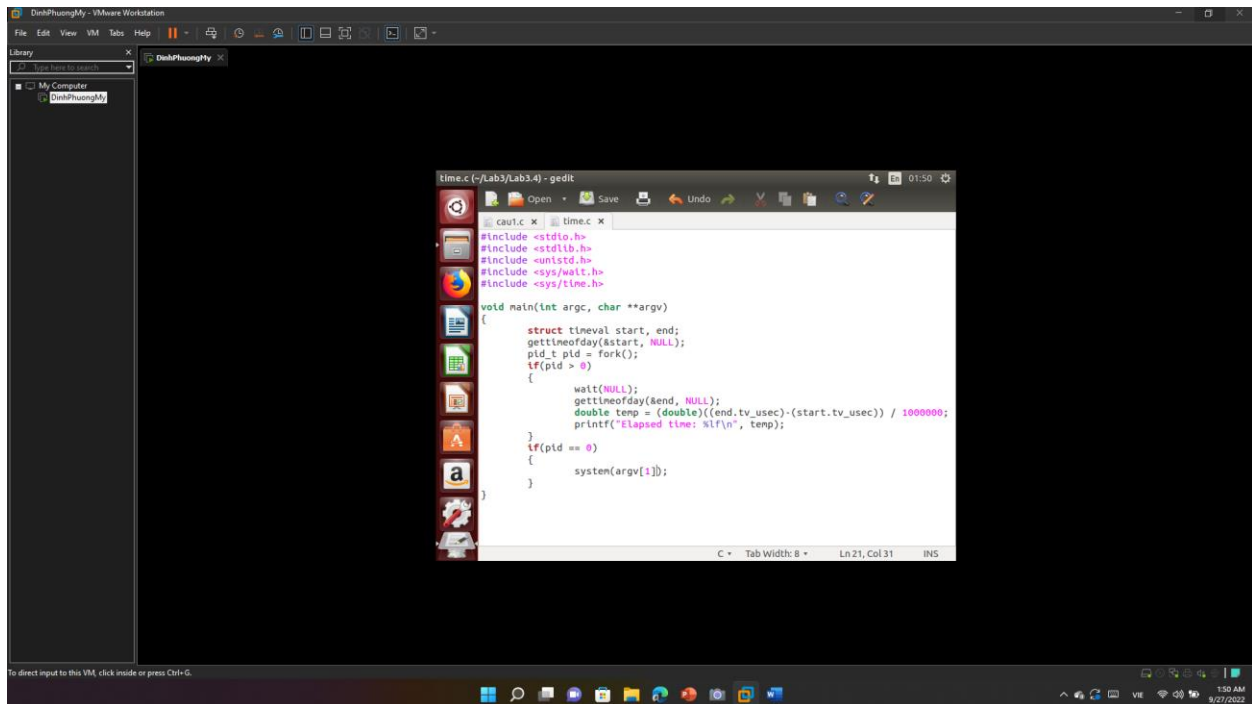


Lab 3.4

Câu 3: Viết chương trình xác định thời gian cần thiết để thực thi một lệnh, lệnh này truyền vào qua phần đối số như minh họa dưới đây. Hàm `gettimeofday()` có thể dùng để tính thời gian. Thân hàm chính nên được thiết kế như sau:

- Lấy thời gian hiện tại (bắt đầu)
- `fork()` để tạo tiến trình con và tiến trình con dùng `system()` để thực thi lệnh truyền vào.
- Khi tiến trình con kết thúc, tiến trình cha đợi bằng lời gọi `wait()`.
- Lấy thời gian hiện tại (kết thúc) và tính ra thời gian đã trôi qua (kết thúc – bắt đầu)

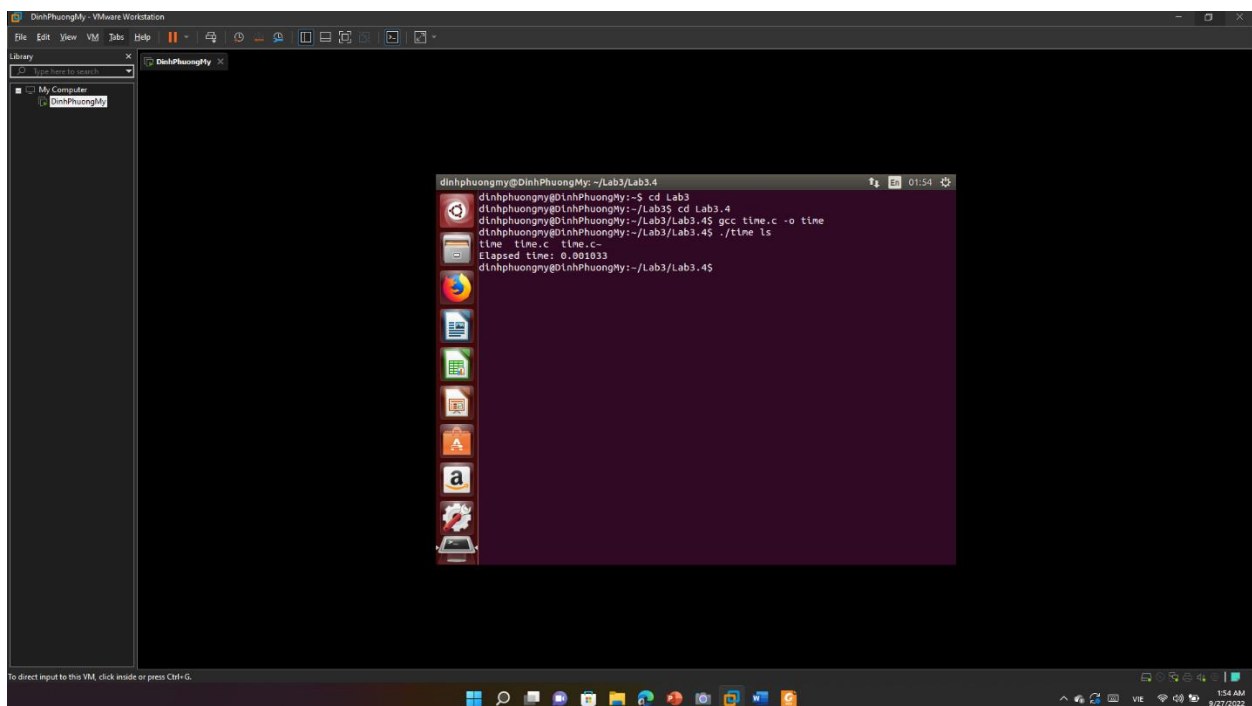
Code:



```
time.c (-/Lab3/Lab3.4) - gedit
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/wait.h>
#include <sys/time.h>

void main(int argc, char **argv)
{
    struct timeval start, end;
    gettimeofday(&start, NULL);
    pid_t pid = fork();
    if(pid > 0)
    {
        wait(NULL);
        gettimeofday(&end, NULL);
        double temp = (double)((end.tv_usec)-(start.tv_usec)) / 1000000;
        printf("Elapsed time: %f\n", temp);
    }
    if(pid == 0)
    {
        system(argv[1]);
    }
}
```

Run:



```
dinhphuongmy@DinhPhuongMy: ~/Lab3/Lab3.4
dinhphuongmy@DinhPhuongMy:~$ cd Lab3
dinhphuongmy@DinhPhuongMy:~/Lab3$ cd Lab3.4
dinhphuongmy@DinhPhuongMy:~/Lab3/Lab3.4$ gcc time.c -o time
dinhphuongmy@DinhPhuongMy:~/Lab3/Lab3.4$ ./time ls
time ./time.c time.c
Elapsed time: 0.001033
dinhphuongmy@DinhPhuongMy:~/Lab3/Lab3.4$
```