

Tên	Đinh Phương My
MSSV	52100703
Nhóm thực hành	N101

BÀI BÁO CÁO THỰC HÀNH LAB 5

Lab 5.1

Câu 1: Tiến trình cha nhận vào các đối số thông qua lời gọi thực thi. Và lần lượt ghi vào đường ống. Tiến trình con nhận dữ liệu rồi in ra màn hình.

Theo Unname pipe:

Code:

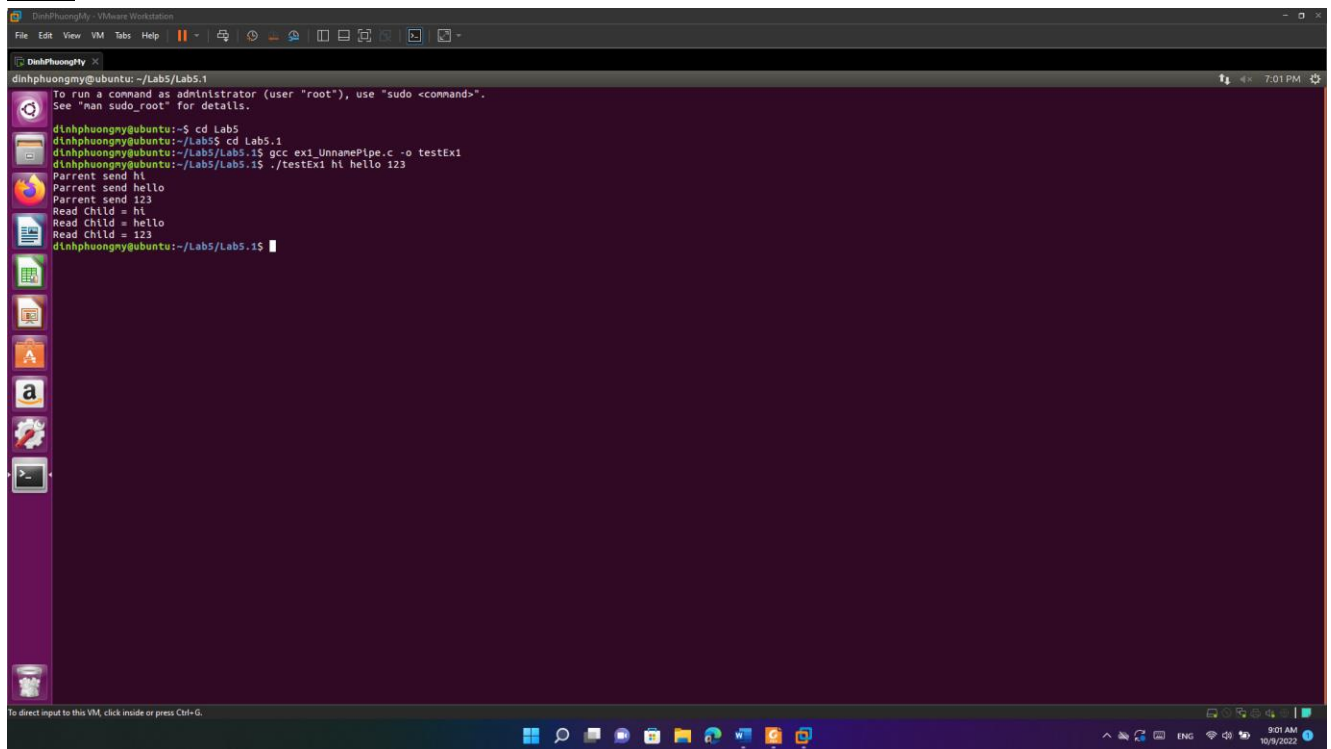
```

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

int main(int argc, char *argv[])
{
    int n = argc - 1, fp[2];
    if(pipe(fp) == -1)
    {
        printf("Pipe Failed!");
        return 0;
    }
    pid_t pid = fork();
    if(pid < 0)
    {
        printf("Fork Failed!");
        exit(1);
    }
    if(pid == 0)
    {
        sleep(1);
        char buff[256];
        close(fp[1]);
        while(read(fp[0], buff, sizeof(buff)) > 0)
        {
            printf("Read Child = %s\n", buff);
        }
        close(fp[0]);
        exit(0);
    }
    else
    {
        close(fp[0]);
        int i = 0;
        for(i = 0; i < n; i++)
        {
            close(fp[1]);
            write(fp[1], argv[i + 1], 256);
            printf("Parent send %s\n", argv[i + 1]);
        }
        close(fp[1]);
        wait(NULL);
    }
    return 0;
}

```

Run:

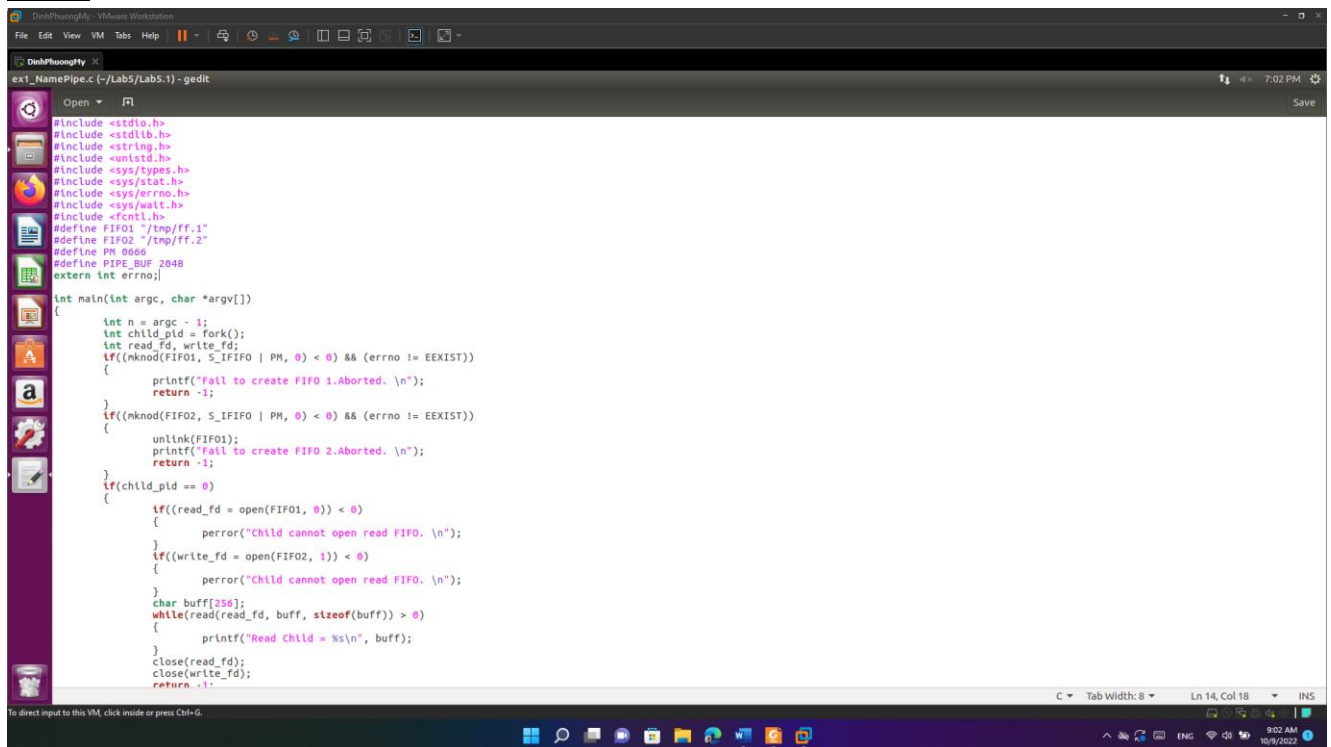


```
dinhphuongmy@ubuntu: ~/Lab5/Lab5.1
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

dinhphuongmy@ubuntu:~/Lab5$ cd Lab5
dinhphuongmy@ubuntu:~/Lab5$ cd Lab5.1
dinhphuongmy@ubuntu:~/Lab5/Lab5.1$ gcc ex1_UnnamePipe.c -o testEx1
dinhphuongmy@ubuntu:~/Lab5/Lab5.1$ ./testEx1 hi hello 123
Parent send hi
Parent send hello
Parent send 123
Read Child = hi
Read Child = hello
Read Child = 123
dinhphuongmy@ubuntu:~/Lab5/Lab5.1$
```

Theo Name pipe:

Code:



```
ex1_NamePipe.c (-/Lab5/Lab5.1) - gedit
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/errno.h>
#include <sys/wait.h>
#include <fcntl.h>
#define FIFO1 "/tmp/ff.1"
#define FIFO2 "/tmp/ff.2"
#define PM 0666
#define PIPE_BUF 2048
extern int errno;

int main(int argc, char *argv[])
{
    int n = argc - 1;
    int child_pid = fork();
    int read_fd, write_fd;
    if((mkfifo(FIFO1, S_IFIFO | PM, 0) < 0) && (errno != EEXIST))
    {
        printf("Fail to create FIFO 1.Aborted. \n");
        return -1;
    }
    if((mkfifo(FIFO2, S_IFIFO | PM, 0) < 0) && (errno != EEXIST))
    {
        unlink(FIFO1);
        printf("Fail to create FIFO 2.Aborted. \n");
        return -1;
    }
    if(child_pid == 0)
    {
        if((read_fd = open(FIFO1, 0)) < 0)
        {
            perror("Child cannot open read FIFO. \n");
        }
        if((write_fd = open(FIFO2, 1)) < 0)
        {
            perror("Child cannot open read FIFO. \n");
        }
        char buff[256];
        while(read(read_fd, buff, sizeof(buff)) > 0)
        {
            printf("Read Child = %s\n", buff);
        }
        close(read_fd);
        close(write_fd);
        return -1;
    }
}
```

The screenshot shows a Windows desktop with a virtual machine named 'DishPhuongThy' running Ubuntu. The terminal window displays the execution of a C program named 'ex1_NamePipe.c' which demonstrates named pipes (FIFOs). The program creates two FIFOs, FIFO1 and FIFO2. It then forks a child process. The parent process opens FIFO1 for writing and FIFO2 for reading. The child process opens FIFO2 for reading and FIFO1 for writing. The parent writes 'Parent send' to FIFO1, and the child reads it. The child writes 'Child cannot open read FIFO.' to FIFO1, and the parent reads it. The program ends with 'Fork Failed.' and returns -1.

```

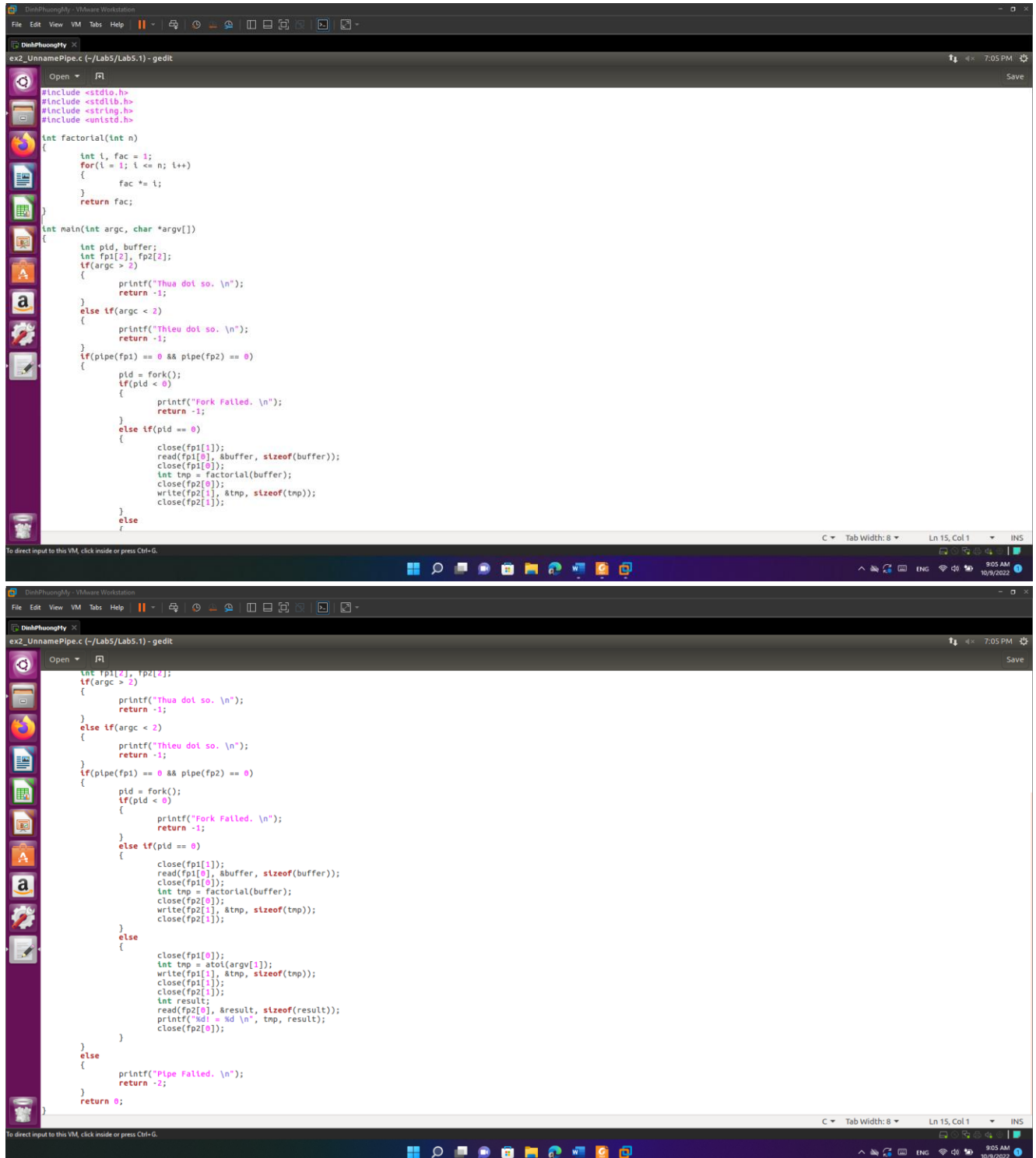
}
{
    if((write_fd = open(FIFO2, 1)) < 0)
    {
        perror("Child cannot open read FIFO. \n");
    }
    char buff[256];
    while(read(read_fd, buff, sizeof(buff)) > 0)
    {
        printf("Read Child = %s\n", buff);
    }
    close(read_fd);
    close(write_fd);
    return -1;
}
else if(chld_pid > 0)
{
    if((write_fd = open(FIFO1, 1)) < 0)
    {
        perror("Parent cannot open read FIFO. \n");
    }
    if((read_fd = open(FIFO2, 0)) < 0)
    {
        perror("Child cannot open read FIFO. \n");
    }
    int i = 0;
    for(i = 0; i < n; i++)
    {
        write(write_fd, argv[i + 1], 256);
        printf("Parent send %s\n", argv[i + 1]);
    }
    close(read_fd);
    close(write_fd);
    if(unlink(FIFO1) < 0)
    {
        printf("Cannot remove FIFO1.\n");
    }
    if(unlink(FIFO2) < 0)
    {
        printf("Cannot remove FIFO2.\n");
    }
    wait(NULL);
    return i;
}
else
{
    printf("Fork Failed.\n");
    return -1;
}
}

```

Câu 2: Tiến trình cha chuyển đổi số đầu tiên (argv [1]) là một số nguyên lớn hơn 3 cho tiến trình con thông qua đường ống. Tiến trình con nhận, tính giá trị $n! = 1 * 2 * \dots * n$ và ghi nó vào đường ống. Tiến trình cha nhận và xuất dữ liệu ra màn hình.

Theo Unname pipe:

Code:

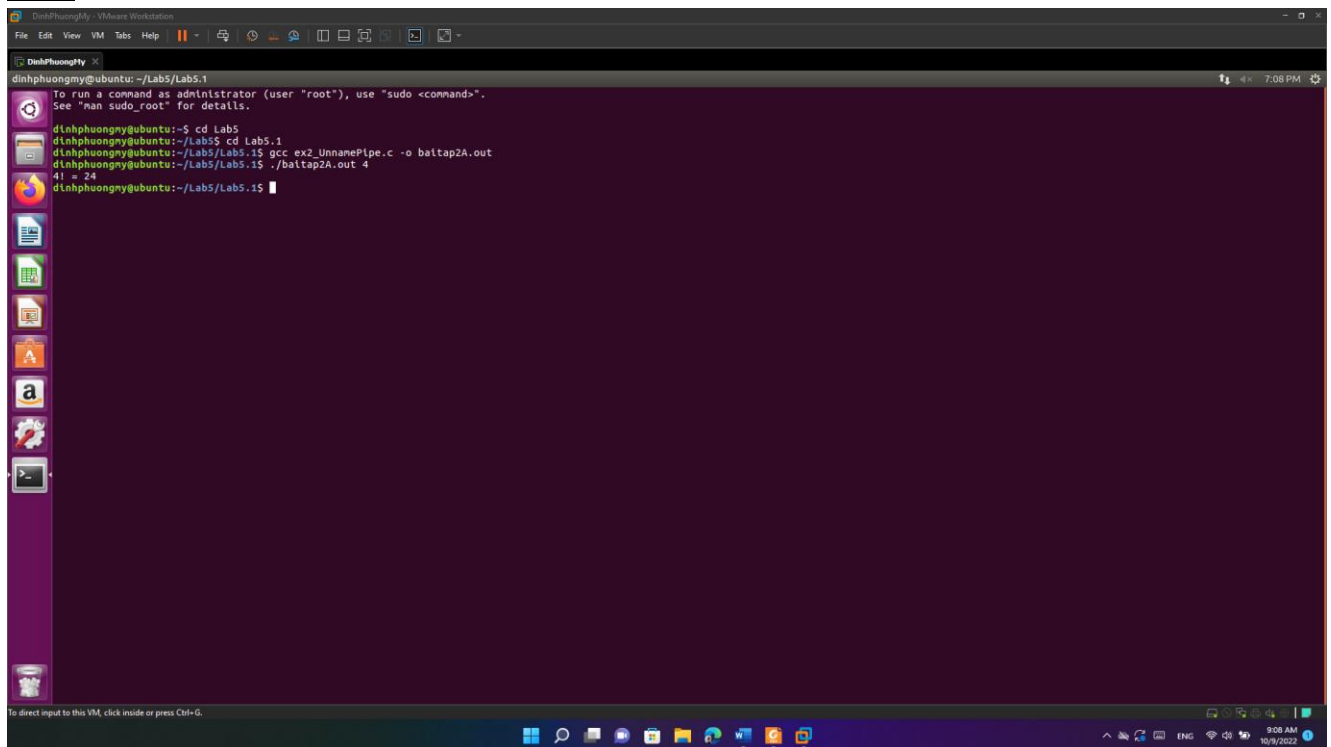


```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>

int factorial(int n)
{
    int i, fac = 1;
    for(i = 1; i <= n; i++)
    {
        fac *= i;
    }
    return fac;
}

int main(int argc, char *argv[])
{
    int pid, buffer;
    int fp1[2], fp2[2];
    if(argc > 2)
    {
        printf("Thua doi so. \n");
        return -1;
    }
    else if(argc < 2)
    {
        printf("Thieu doi so. \n");
        return -1;
    }
    if(pipe(fp1) == 0 && pipe(fp2) == 0)
    {
        pid = fork();
        if(pid < 0)
        {
            printf("Fork Failed. \n");
            return -1;
        }
        else if(pid == 0)
        {
            close(fp1[1]);
            read(fp1[0], &buffer, sizeof(buffer));
            close(fp1[0]);
            int tmp = factorial(buffer);
            close(fp2[0]);
            write(fp2[1], &tmp, sizeof(tmp));
            close(fp2[1]);
        }
        else
        {
            close(fp1[0]);
            int tmp = atoi(argv[1]);
            write(fp1[1], &tmp, sizeof(tmp));
            close(fp1[1]);
            close(fp2[1]);
            int result;
            read(fp2[0], &result, sizeof(result));
            printf("Id = %d \n", tmp, result);
            close(fp2[0]);
        }
    }
    else
    {
        printf("Pipe Failed. \n");
        return -2;
    }
    return 0;
}
```

Run:

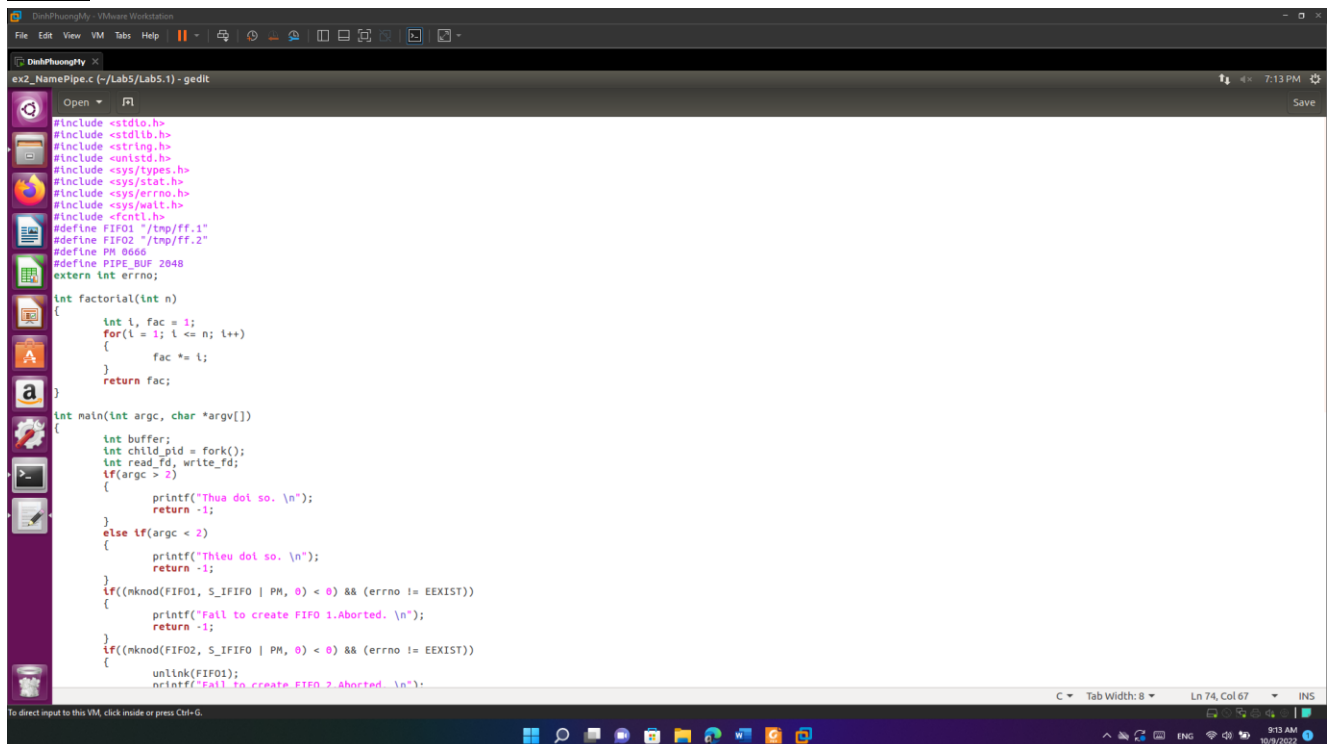


```
dinhphuongmy@ubuntu: ~/Lab5/Lab5.1
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

dinhphuongmy@ubuntu:~/Lab5$ cd Lab5
dinhphuongmy@ubuntu:~/Lab5$ gcc ex2_UnnamePipe.c -o baitap2A.out
dinhphuongmy@ubuntu:~/Lab5$ ./baitap2A.out 4
41 = 24
dinhphuongmy@ubuntu:~/Lab5$
```

Theo Name pipe:

Code:

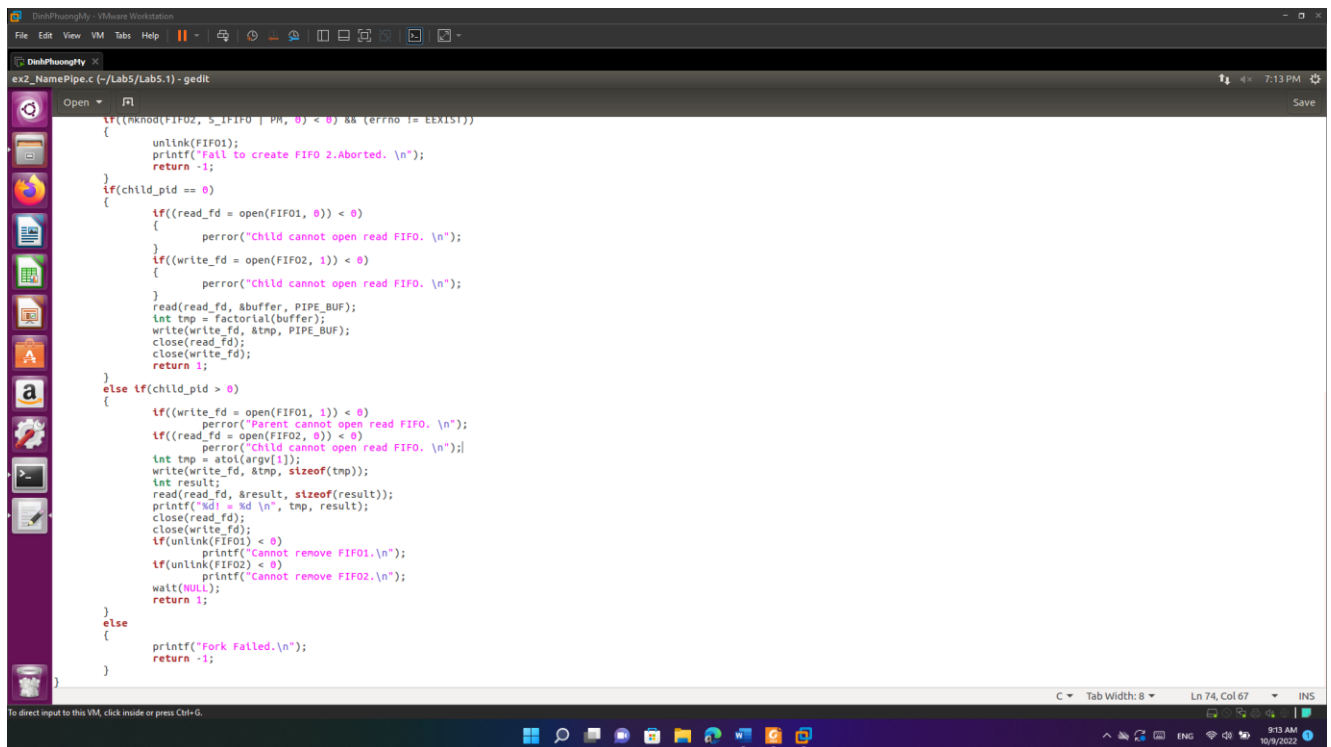


```
ex2_NamePipe.c (-/Lab5/Lab5.1) - gedit
Open  [icon] Save

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/errno.h>
#include <sys/wait.h>
#include <fcntl.h>
#define FIFO1 "/tmp/ff.1"
#define FIFO2 "/tmp/ff.2"
#define PM 0666
#define PIPE_BUF 2048
extern int errno;

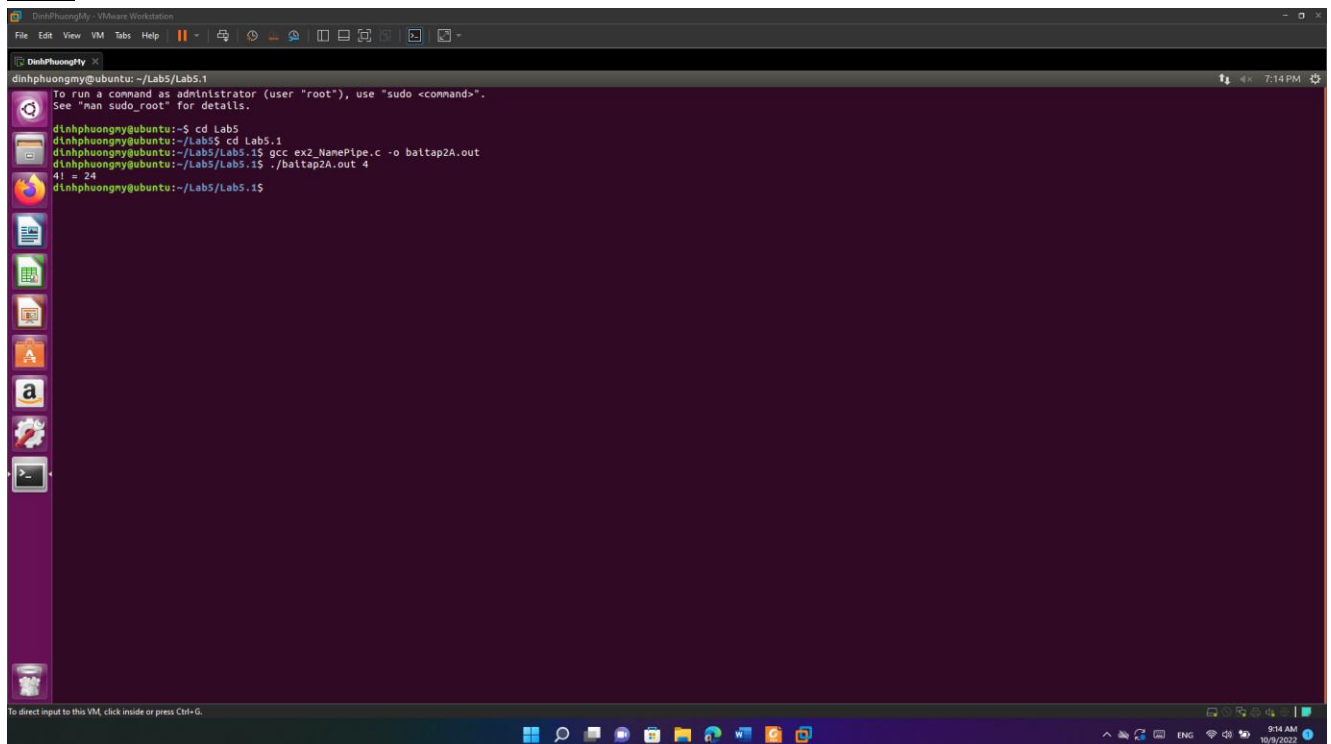
int factorial(int n)
{
    int i, fac = 1;
    for(i = 1; i <= n; i++)
    {
        fac *= i;
    }
    return fac;
}

int main(int argc, char *argv[])
{
    int buffer;
    int child_pid = fork();
    int read_fd, write_fd;
    if(argc > 2)
    {
        printf("Thua doi so. \n");
        return -1;
    }
    else if(argc < 2)
    {
        printf("Thieu doi so. \n");
        return -1;
    }
    if((mknode(FIFO1, S_IFIFO | PM, 0) < 0) && (errno != EEXIST))
    {
        printf("Fail to create FIFO 1.Aborted. \n");
        return -1;
    }
    if((mknode(FIFO2, S_IFIFO | PM, 0) < 0) && (errno != EEXIST))
    {
        unlink(FIFO1);
        printf("Fail to create FIFO 2.Aborted. \n");
    }
}
```



```
ex2_NamePipe.c (~/.Lab5/Lab5.1) - gedit
1  if((mkfifo(FIFO2, S_IFIFO | 0600) < 0) && (errno != EEXIST))
2  {
3      unlink(FIFO1);
4      printf("Fail to create FIFO 2.Aborted. \n");
5      return -1;
6  }
7  if(child_pid == 0)
8  {
9      if((read_fd = open(FIFO1, 0)) < 0)
10     {
11         perror("Child cannot open read FIFO. \n");
12     }
13     if((write_fd = open(FIFO2, 1)) < 0)
14     {
15         perror("Child cannot open read FIFO. \n");
16     }
17     read(read_fd, &buffer, PIPE_BUF);
18     int tmp = factorial(buffer);
19     write(write_fd, &tmp, PIPE_BUF);
20     close(read_fd);
21     close(write_fd);
22     return 1;
23 }
24 else if(child_pid > 0)
25 {
26     if((write_fd = open(FIFO1, 1)) < 0)
27     {
28         perror("Parent cannot open read FIFO. \n");
29     }
30     if((read_fd = open(FIFO2, 0)) < 0)
31     {
32         perror("Child cannot open read FIFO. \n");
33     }
34     int tmp = atoi(argv[1]);
35     write(write_fd, &tmp, sizeof(tmp));
36     int result;
37     read(read_fd, &result, sizeof(result));
38     printf("%d! = %d \n", tmp, result);
39     close(read_fd);
40     close(write_fd);
41     if(unlink(FIFO1) < 0)
42     {
43         printf("Cannot remove FIFO1.\n");
44     }
45     if(unlink(FIFO2) < 0)
46     {
47         printf("Cannot remove FIFO2.\n");
48     }
49     wait(NULL);
50     return 1;
51 }
52 else
53 {
54     printf("Fork Failed.\n");
55     return -1;
56 }
57 }
```

Run:



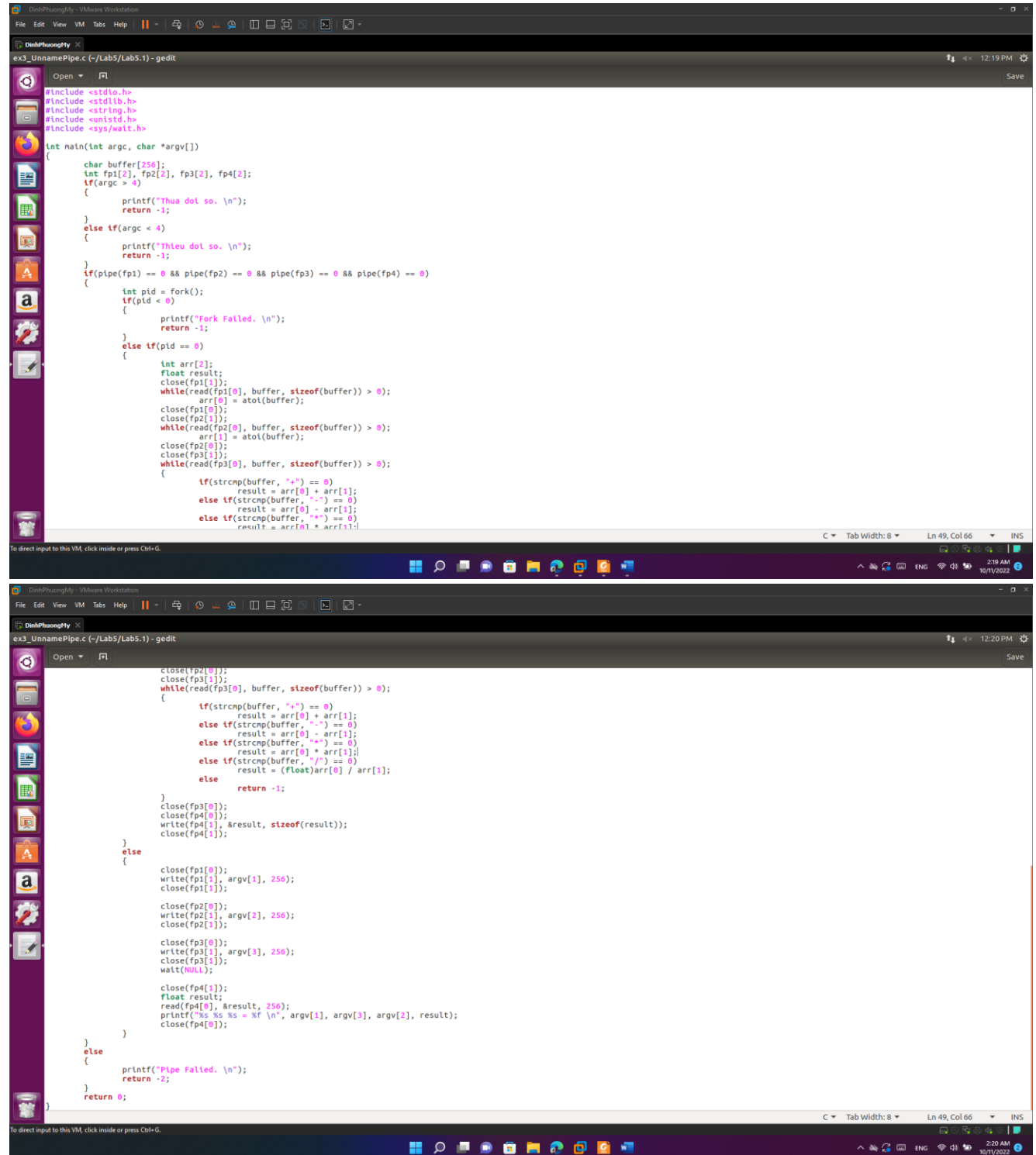
```
dinhphuongmy@ubuntu:~/Lab5/Lab5.1
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

dinhphuongmy@ubuntu:~/Lab5/Lab5.1$ cd Lab5
dinhphuongmy@ubuntu:~/Lab5/Lab5.1$ gcc ex2_NamePipe.c -o baltap2A.out
dinhphuongmy@ubuntu:~/Lab5/Lab5.1$ ./baltap2A.out 4
4! = 24
dinhphuongmy@ubuntu:~/Lab5/Lab5.1$
```

Câu 3: Tiến trình cha đọc hai số nguyên và một thao tác +, -, *, / và chuyển tất cả cho tiến trình con. Tiến trình con tính toán kết quả và trả về cho tiến trình cha để in ra màn hình.

Theo Unname pipe:

Code:



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

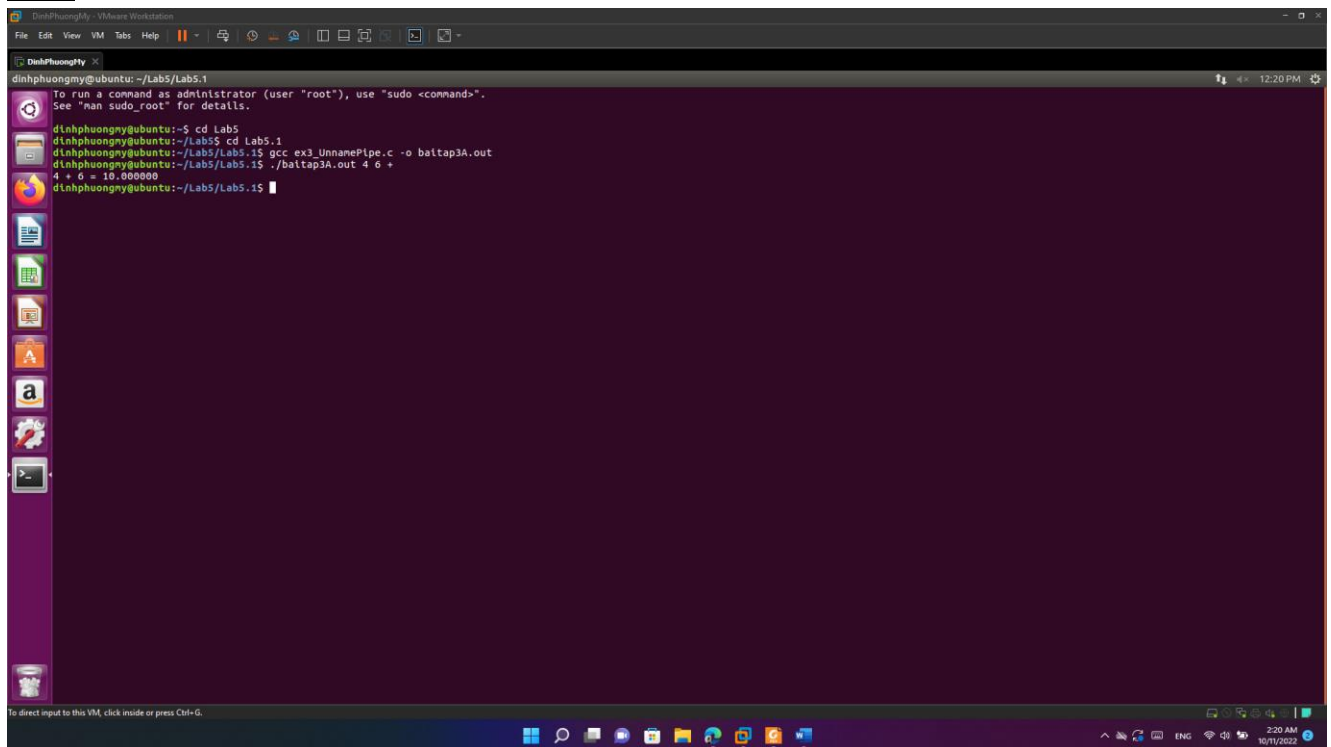
int main(int argc, char *argv[])
{
    char buffer[256];
    int fp1[2], fp2[2], fp3[2], fp4[2];
    if(argc > 4)
    {
        printf("Thua doi so. \n");
        return -1;
    }
    else if(argc < 4)
    {
        printf("Thieu doi so. \n");
        return -1;
    }
    if(pipe(fp1) == 0 && pipe(fp2) == 0 && pipe(fp3) == 0 && pipe(fp4) == 0)
    {
        int pid = fork();
        if(pid < 0)
        {
            printf("Fork Failed. \n");
            return -1;
        }
        else if(pid == 0)
        {
            int arr[2];
            float result;
            close(fp1[1]);
            while(read(fp1[0], buffer, sizeof(buffer)) > 0);
            arr[0] = atoi(buffer);
            close(fp2[1]);
            while(read(fp2[0], buffer, sizeof(buffer)) > 0);
            arr[1] = atoi(buffer);
            close(fp2[0]);
            close(fp3[1]);
            while(read(fp3[0], buffer, sizeof(buffer)) > 0);
            {
                if(strcmp(buffer, "+") == 0)
                    result = arr[0] + arr[1];
                else if(strcmp(buffer, "-") == 0)
                    result = arr[0] - arr[1];
                else if(strcmp(buffer, "*") == 0)
                    result = arr[0] * arr[1];
                else if(strcmp(buffer, "/") == 0)
                    result = (float)arr[0] / arr[1];
                else
                    return -1;
            }
            close(fp3[0]);
            close(fp4[0]);
            write(fp4[1], &result, sizeof(result));
            close(fp4[1]);
        }
        else
        {
            close(fp1[0]);
            write(fp1[1], argv[1], 256);
            close(fp1[1]);

            close(fp2[0]);
            write(fp2[1], argv[2], 256);
            close(fp2[1]);

            close(fp3[0]);
            write(fp3[1], argv[3], 256);
            close(fp3[1]);
            wait(NULL);

            close(fp4[1]);
            float result;
            read(fp4[0], &result, 256);
            printf("%s %s %s = %f \n", argv[1], argv[3], argv[2], result);
            close(fp4[0]);
        }
    }
    else
    {
        printf("Pipe Failed. \n");
        return -2;
    }
    return 0;
}
```

Run:

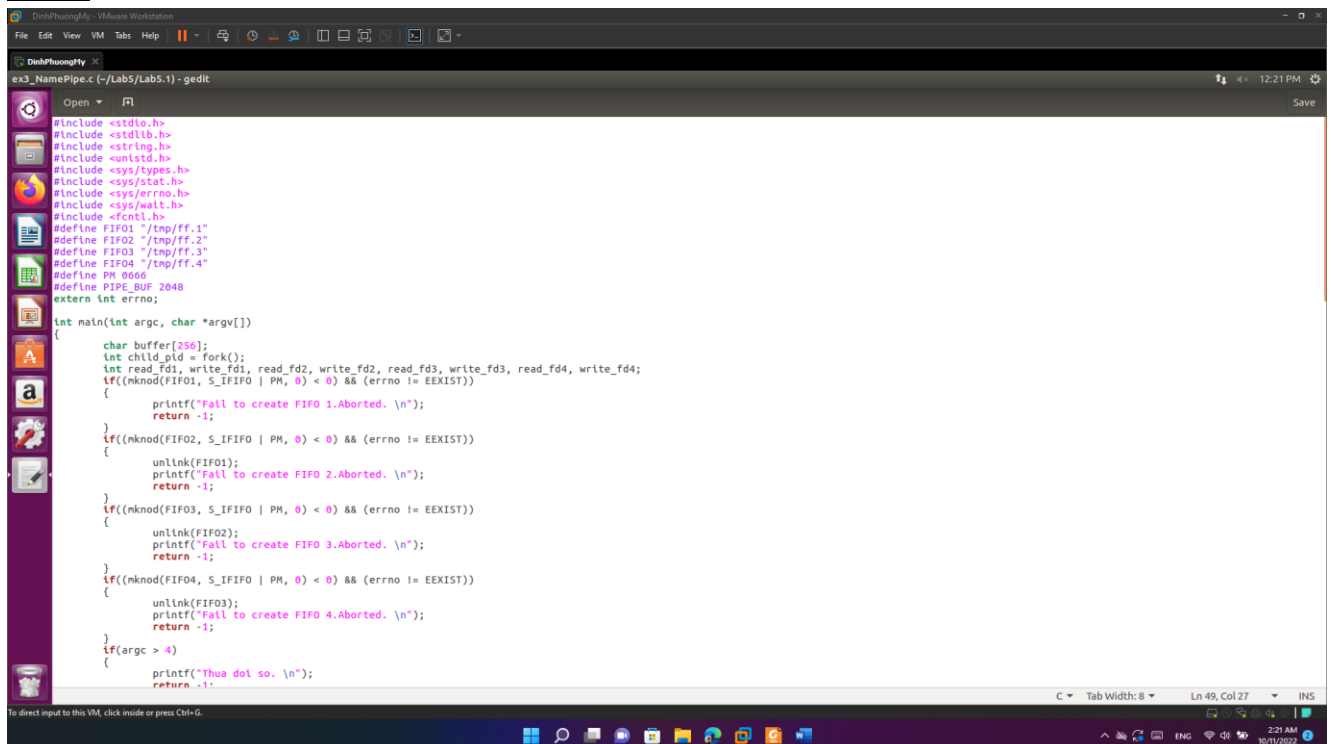


```
dinhphuongmy@ubuntu: ~/Lab5/Lab5.1
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

dinhphuongmy@ubuntu:~/Lab5$ cd Lab5
dinhphuongmy@ubuntu:~/Lab5$ gcc ex3_UnnamePipe.c -o baitap3A.out
dinhphuongmy@ubuntu:~/Lab5$ ./baitap3A.out 4 6
4 + 6 = 10.000000
dinhphuongmy@ubuntu:~/Lab5$
```

Theo Name pipe:

Code:



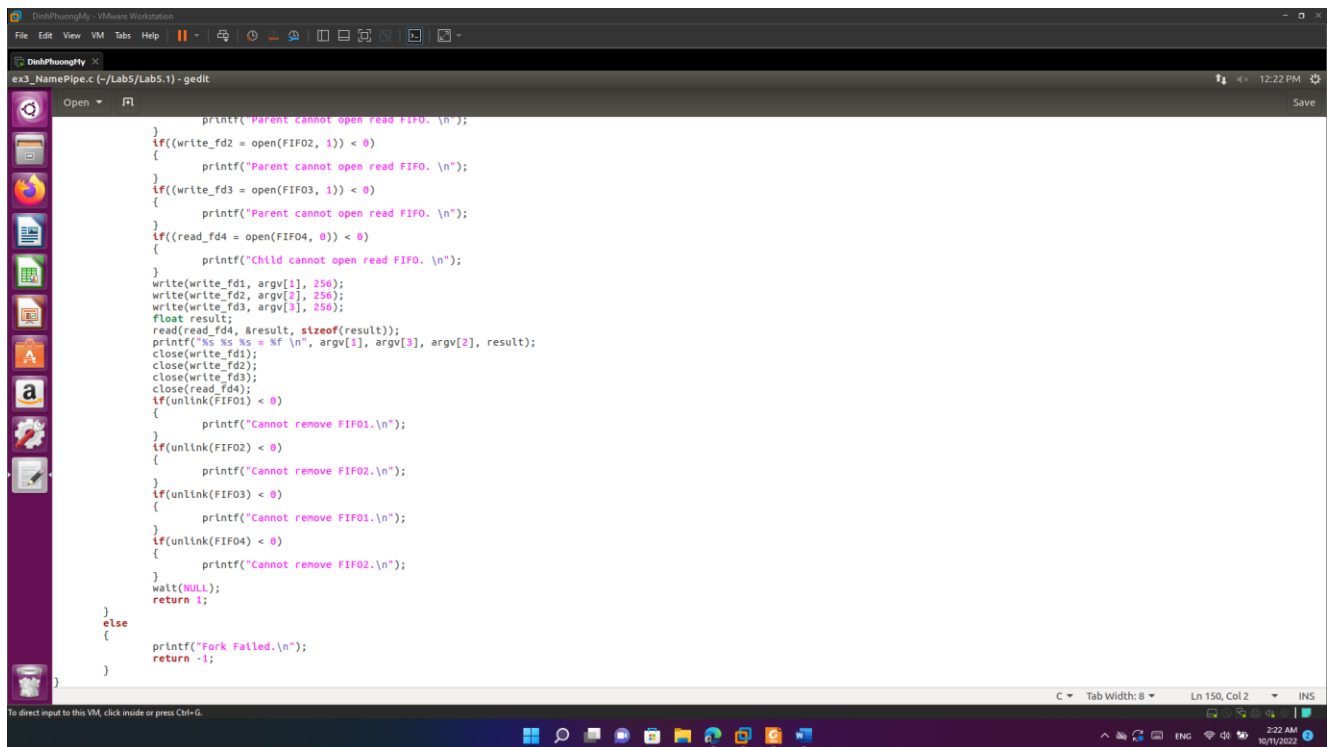
```
ex3_NamePipe.c (-/Lab5/Lab5.1) - gedit
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/errno.h>
#include <sys/wait.h>
#include <fcntl.h>
#define FIFO1 "/tmp/ff.1"
#define FIFO2 "/tmp/ff.2"
#define FIFO3 "/tmp/ff.3"
#define FIFO4 "/tmp/ff.4"
#define PH 0666
#define PIPE_BUF 2048
extern int errno;

int main(int argc, char *argv[])
{
    char buffer[256];
    int chld_pid = fork();
    int read_fd1, write_fd1, read_fd2, write_fd2, read_fd3, write_fd3, read_fd4, write_fd4;
    if((mknod(FIFO1, S_IFIFO | PH, 0) < 0) && (errno != EEXIST))
    {
        printf("Fail to create FIFO 1.Aborted. \n");
        return -1;
    }
    if((mknod(FIFO2, S_IFIFO | PH, 0) < 0) && (errno != EEXIST))
    {
        unlink(FIFO1);
        printf("Fail to create FIFO 2.Aborted. \n");
        return -1;
    }
    if((mknod(FIFO3, S_IFIFO | PH, 0) < 0) && (errno != EEXIST))
    {
        unlink(FIFO2);
        printf("Fail to create FIFO 3.Aborted. \n");
        return -1;
    }
    if((mknod(FIFO4, S_IFIFO | PH, 0) < 0) && (errno != EEXIST))
    {
        unlink(FIFO3);
        printf("Fail to create FIFO 4.Aborted. \n");
        return -1;
    }
    if(argc > 4)
    {
        printf("Thua doi so. \n");
        return -1;
    }
}
```



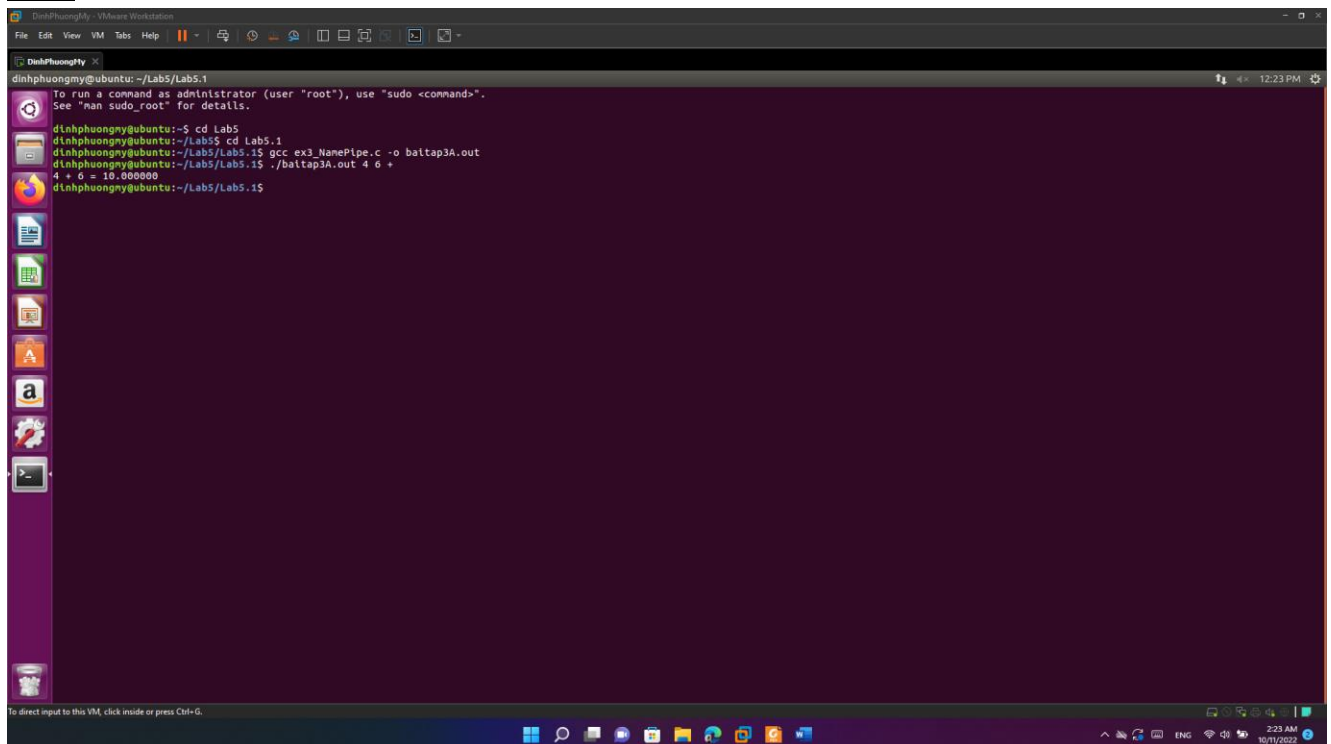
```
DinhPhuongMy - VMware Workstation
File Edit View VM Tabs Help
DinhPhuongMy
ex3_NamePipe.c (~\Lab5\Lab5.1) - gedit
Open
return -1;
}
else if(argc < 4)
{
    printf("Thieu doi so. \n");
    return -1;
}
if(chld_pid == 0)
{
    if((read_fd1 = open(FIFO1, 0)) < 0)
    {
        perror("Child cannot open read FIFO. \n");
    }
    if((read_fd2 = open(FIFO2, 0)) < 0)
    {
        perror("Child cannot open read FIFO. \n");
    }
    if((read_fd3 = open(FIFO3, 0)) < 0)
    {
        perror("Child cannot open read FIFO. \n");
    }
    if((write_fd4 = open(FIFO4, 1)) < 0)
    {
        perror("Child cannot open read FIFO. \n");
    }
    int arr[2];
    float result;
    read(read_fd1, buffer, sizeof(buffer));
    arr[0] = atoi(buffer);
    read(read_fd2, buffer, sizeof(buffer));
    arr[1] = atoi(buffer);
    read(read_fd3, buffer, sizeof(buffer));
    if(strcmp(buffer, "+") == 0)
    {
        result = arr[0] + arr[1];
    }
    else if(strcmp(buffer, "-") == 0)
    {
        result = arr[0] - arr[1];
    }
    else if(strcmp(buffer, "*") == 0)
    {
        result = arr[0] * arr[1];
    }
    else if(strcmp(buffer, "/") == 0)
    {
        result = (float)arr[0] / arr[1];
    }
    else
    {
        return -1;
    }
    write(write_fd4, &result, PIPE_BUF);
    close(read_fd1);
    close(read_fd2);
    close(read_fd3);
    close(write_fd4);
    return 1;
}
}
```

```
DinhPhuongMy - VMware Workstation
File Edit View VM Tabs Help
DinhPhuongMy
ex3_NamePipe.c (~\Lab5\Lab5.1) - gedit
Open
else if(chld_pid > 0)
{
    if((write_fd1 = open(FIFO1, 1)) < 0)
    {
        printf("Parent cannot open read FIFO. \n");
    }
    if((write_fd2 = open(FIFO2, 1)) < 0)
    {
        printf("Parent cannot open read FIFO. \n");
    }
    if((write_fd3 = open(FIFO3, 1)) < 0)
    {
        printf("Parent cannot open read FIFO. \n");
    }
    if((read_fd4 = open(FIFO4, 0)) < 0)
    {
        printf("Child cannot open read FIFO. \n");
    }
    write(write_fd1, argv[1], 256);
    write(write_fd2, argv[2], 256);
    write(write_fd3, argv[3], 256);
    float result;
    read(read_fd4, &result, sizeof(result));
    printf("%s %s %s = %f \n", argv[1], argv[3], argv[2], result);
    close(write_fd1);
    close(write_fd2);
    close(write_fd3);
    close(read_fd4);
    if(unlink(FIFO1) < 0)
    {
        printf("Cannot remove FIFO1.\n");
    }
    if(unlink(FIFO2) < 0)
    {
        printf("Cannot remove FIFO2.\n");
    }
    if(unlink(FIFO3) < 0)
    {
        printf("Cannot remove FIFO1.\n");
    }
    if(unlink(FIFO4) < 0)
    {
        printf("Cannot remove FIFO2.\n");
    }
    wait(NULL);
    return 1;
}
else
```



```
ex3_NamePipe.c (~/.Lab5/Lab5.1) - gedit
printf("Parent cannot open read FIFO. \n");
if((write_fd2 = open(FIFO2, 1)) < 0)
{
    printf("Parent cannot open read FIFO. \n");
}
if((write_fd3 = open(FIFO3, 1)) < 0)
{
    printf("Parent cannot open read FIFO. \n");
}
if((read_fd4 = open(FIFO4, 0)) < 0)
{
    printf("Child cannot open read FIFO. \n");
}
write(write_fd1, argv[1], 256);
write(write_fd2, argv[2], 256);
write(write_fd3, argv[3], 256);
float result;
read(read_fd4, &result, sizeof(result));
printf("%s %s %s = %f \n", argv[1], argv[3], argv[2], result);
close(write_fd1);
close(write_fd2);
close(write_fd3);
close(read_fd4);
if(unlink(FIFO1) < 0)
{
    printf("Cannot remove FIFO1.\n");
}
if(unlink(FIFO2) < 0)
{
    printf("Cannot remove FIFO2.\n");
}
if(unlink(FIFO3) < 0)
{
    printf("Cannot remove FIFO1.\n");
}
if(unlink(FIFO4) < 0)
{
    printf("Cannot remove FIFO2.\n");
}
wait(NULL);
return 1;
}
else
{
    printf("Fork Failed.\n");
    return -1;
}
```

Run:



```
dinhphuongmy@ubuntu:~/Lab5/Lab5.1
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

dinhphuongmy@ubuntu:~/Lab5$ cd Lab5
dinhphuongmy@ubuntu:~/Lab5$ gcc ex3_NamePipe.c -o baltap3A.out
dinhphuongmy@ubuntu:~/Lab5$ ./baltap3A.out 4 6 +
4 * 6 = 10.000000
dinhphuongmy@ubuntu:~/Lab5$
```