Họ và tên: Nguyễn Trọng Đạt

MSSV: 52100176

Lớp: 21050301

Lab 5

Bài 1:

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

int main(int argc, char* argv[]){
        int fp1[2], fp2[2];
        int buffer;
        int pid;
        if(argc < 2){
                return -1;
        }
        if(pipe(fp1) == 0 && pipe(fp2) == 0){
                pid = fork();
                if(pid < 0) {printf("Failed \n"); return -1;}
                else if(pid == 0){
                        close(fp1[1]);
                        read(fp1[0], &buffer, sizeof(buffer));
                        printf("Read form parents: %d\n", buffer);
                        close(fp1[0]);
                        int n = buffer;
                        printf("Data send to parents: %d\n", n);
                        close(fp2[0]);
                        write(fp2[1], &n, sizeof(n));
                        close(fp2[1]);
                }
                else {
                        close(fp1[0]);
                        printf("Data from parents: %s\n", argv[1]);
                        int temp = atoi(argv[1]);
                        write(fp1[1], &temp, sizeof(temp));
                        close(fp1[1]);
                        printf("Da viet\n");
                        close(fp2[1]);
                        int tam;
                        read(fp2[0], &tam, sizeof(tam));
                        printf("Data get from chils %d\n", tam);
                        close(fp2[0]);
                }
        }
        else {printf("Pipe failed\n"); return -2;}
```

```
                }
        }
        else {printf("Pipe failed\n"); return -2;}
}
```

```
trongdat1108@ubuntu:~/lab5.1$ gcc -c bai1.c
bai1.c: In function ‘main’:
bai1.c:29:15: warning: implicit declaration of function ‘atoi’ [-Wimplicit-function-declaration]
    int temp = atoi(argv[1]);
               ^
trongdat1108@ubuntu:~/lab5.1$ gcc -o bai1.out bai1.o
trongdat1108@ubuntu:~/lab5.1$ ./bai1.out 12
Data from parents: 12
Da viet
Read form parents: 12
Data send to parents: 12
Data get from chils 12
trongdat1108@ubuntu:~/lab5.1$
```

Bài 1 name:

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/errno.h>
#define FIFO1 "/tmp/ff.1"
#define FIFO2 "/tmp/ff.2"
#define PM 0666
extern int errno;
#define PIPE_BUF 4096
int main(int argc, char *argv[])
{
    char s1[PIPE_BUF], s2[PIPE_BUF];
    int childpid, readfd, writefd;
    if ((mknod(FIFO1, S_IFIFO | PM, 0) < 0) && (errno != EEXIST))
    {
        printf("Fail to create FIFO 1. Aborted.\n");
        return -1;
    }
    if ((mknod(FIFO2, S_IFIFO | PM, 0) < 0) && (errno != EEXIST))
    {
        unlink(FIFO1);
        printf("Fail to create FIFO 2. Aborted.\n");
        return -1;
    }
    childpid = fork();
    if (childpid == 0)
    { // child
        if ((readfd = open(FIFO1, 0)) < 0)
            perror("Child cannot open readFIFO.\n");
        while(read(readfd, s2, PIPE_BUF)){
            printf("%s\n", s2);
        }
        close(readfd);
        return 1;
    }
    else if (childpid > 0)
    { // parent
        if ((writefd = open(FIFO1, 1)) < 0)
            perror("Parent cannot open writeFIFO.\n");
        int i;
        for (i = 1; i < argc; i++)
        {
            gets(s1);
            write(writefd, s1, PIPE_BUF);
        }
        while (wait((int *)0) != childpid);
        close(writefd);
        if (unlink(FIFO1) < 0)
            perror("Cannot remove FIFO1.\n");
        return 1;
    }
    else
    {
        printf("Fork failed\n");
        return -1;
    }
}
```

Bài 2:

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

int main(int argc, char* argv[]){
    int fp1[2],fp2[2];
    int buffer;
    int pid;
    if(argc<2  && argv[1]<=3) {
        return -1;
    }
    if(pipe(fp1)==0 && pipe(fp2)==0){
        pid = fork();
        if(pid<0) {printf("Failed \n"); return -1;}
        else if(pid==0) {
            close(fp1[1]);
            read(fp1[0], &buffer, sizeof(buffer));
            printf("Read from parents: %d\n", buffer);
            close(fp1[0]);
            int n= 1;
            int c;
            for(c=1;c<=buffer;c++) {n *= c;}
            printf("data send to parent: %d \n",n);
            close(fp2[0]);
            write(fp2[1], &n, sizeof(n));
            close(fp2[1]);
        }
        else {
            close(fp1[0]);
            printf("Data from parents: %s\n", argv[1]);
            int temp =atoi(argv[1]);
            write(fp1[1], &temp, sizeof(temp));
            close(fp1[1]);
            printf("da viet \n");
            close(fp2[1]);
            int tam;
            read(fp2[0], &tam, sizeof(tam));
            printf("data get from child %d \n",tam);
            close(fp2[0]);
        }
    }
    else {printf("Pipe falied \n"); return -2;}
}
```

```
trongdat1108@ubuntu:~/lab5.1$ gcc -c bai2.c
bai2.c: In function ‘main’:
bai2.c:9:26: warning: comparison between pointer and integer
     if(argc<2  && argv[1]<=3) {
                          ^
bai2.c:31:23: warning: implicit declaration of function ‘atoi’ [-Wimplicit-function-declarati
on]
             int temp =atoi(argv[1]);
                       ^
trongdat1108@ubuntu:~/lab5.1$ gcc -o bai2.out bai2.o
trongdat1108@ubuntu:~/lab5.1$ ./bai2.out 12
Data from parents: 12
da viet
Read from parents: 12
data send to parent: 479001600
data get from child 479001600
trongdat1108@ubuntu:~/lab5.1$ ./bai2.out 5
Data from parents: 5
da viet
Read from parents: 5
data send to parent: 120
data get from child 120
trongdat1108@ubuntu:~/lab5.1$
```

Bài 2 name:

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/errno.h>
#define FIFO1 "/tmp/ff.1"
#define FIFO2 "/tmp/ff.2"
#define PM 0666
extern int errno;
#define PIPE_BUF 4096
int main(int argc, char *argv[])
{
    char s1[PIPE_BUF], s2[PIPE_BUF];
    int childpid, readfd, writefd;
    if ((mknod(FIFO1, S_IFIFO | PM, 0) < 0) && (errno != EEXIST))
    {
        printf("Fail to create FIFO 1. Aborted.\n");
        return -1;
    }
    if ((mknod(FIFO2, S_IFIFO | PM, 0) < 0) && (errno != EEXIST))
    {
        unlink(FIFO1);
        printf("Fail to create FIFO 2. Aborted.\n");
        return -1;
    }
    childpid = fork();
    if (childpid == 0)
    { // child
        if ((readfd = open(FIFO1, 0)) < 0)
            perror("Child cannot open readFIFO.\n");
        fflush(stdin);
        read(readfd, s2, PIPE_BUF);
        int cnt = 1;
        int i;
        for (i = 1; i <= atoi(s2); i++)
        {

        for (i = 1; i <= atoi(s2); i++)
        {
            cnt *= i;
        }
        printf("%d!=%d\n", atoi(s2), cnt);
        close(readfd);
        return 1;
    }
    else if (childpid > 0)
    { // parent
        if ((writefd = open(FIFO1, 1)) < 0)
            perror("Parent cannot open writeFIFO.\n");
        fflush(stdin);
        scanf("%s",s1);
        write(writefd, s1,strlen(s1));
        while (wait((int *)0) != childpid)
            ;
        close(writefd);
        if (unlink(FIFO1) < 0)
            perror("Cannot remove FIFO1.\n");
        return 1;
    }
    else
    {
        printf("Fork failed\n");
        return -1;
    }
}
```

```
trongdat1108@ubuntu:~/lab5.1$ gcc -c bai2_name.c
bai2_name.c: In function ‘main’:
bai2_name.c:31:23: warning: implicit declaration of function ‘open’ [-Wimplicit-function-decl
aration]
         if ((readfd = open(FIFO1, 0)) < 0)
                       ^
bai2_name.c:52:16: warning: implicit declaration of function ‘wait’ [-Wimplicit-function-decl
aration]
         while (wait((int *)0) != childpid)
                ^
trongdat1108@ubuntu:~/lab5.1$ gcc -o bai2_name.out bai2_name.o
trongdat1108@ubuntu:~/lab5.1$ ./bai2_name.out 5

12
12!=479001600
trongdat1108@ubuntu:~/lab5.1$ ./bai2_name.out
12
12!=479001600
trongdat1108@ubuntu:~/lab5.1$ ./bai2_name.out
5
5!=120
trongdat1108@ubuntu:~/lab5.1$
```

Bài 3:

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


int main(int argc, char** argv){
        int fp1[2],fp2[2],fp3[2],fp4[2];
        int pid;
        if(argc!=4){
                printf("Thieu doi so\n");
                return 1;
        }
        if(pipe(fp1)==0&&pipe(fp2)==0&&pipe(fp3)==0&&pipe(fp4)==0){
                pid=fork();
                if(pid,0){
                        printf("Fork failed\n");
                        return 1;
                }
                else if(pid==0){
                        int a,b;
                        char c;
                        close(fp1[1]);
                        read(fp1[0],&a,sizeof(a));
                        close(fp2[1]);
                        read(fp2[0],&b,sizeof(b));
                        close(fp3[1]);
                        read(fp3[0],&c,sizeof(c));
                        printf("Child da nhan duoc du lieu tu parent\n");
                        close(fp1[0]);
                        close(fp2[0]);
                        close(fp3[0]);
                        float res;
                        switch(c){
                        case '+':
                                res=a+b;
                                break;
                        case'-':
```

```
                        break;
                case'-':
                        res=a-b;
                        break;
                case '*':
                        res=(float)a*b;
                        break;
                case '/':
                        res=(float)a/b;
                        break;
                }
                close(fp4[0]);
                printf("Gui ket qua den parent......\n");
                write(fp4[1],&res,sizeof(res));
                close(fp4[1]);
        }
        else{
                int a=atoi(argv[1]);
                int b=atoi(argv[2]);
                char c=argv[3][0];
                close(fp1[0]);
                close(fp2[0]);
                close(fp3[0]);
                write(fp1[1],&a,sizeof(a));
                write(fp2[1],&b,sizeof(b));
                write(fp3[1],&c,sizeof(c));
                close(fp1[1]);
                close(fp2[1]);
                close(fp3[1]);
                float res;
                close(fp4[1]);
                read(fp4[0],&res,sizeof(res));
                printf("Parent nhan duoc ket qua = %.2f\n",res);
                close(fp4[0]);
                wait(NULL);
        }
    }
    else{
        printf("Pipe failed\n");
```

```
        }
    }
    else{
        printf("Pipe failed\n");
        return -1;
    }
    return 0;
}
```

```
trongdat1108@ubuntu:~/lab5.1$ gcc -c bai3.c
trongdat1108@ubuntu:~/lab5.1$ gcc -o bai3.out bai3.o
trongdat1108@ubuntu:~/lab5.1$ ./bai3.out 12 2 +
Child da nhan duoc du lieu tu parent
Gui ket qua den parent......
Parent nhan duoc ket qua = 14.00
trongdat1108@ubuntu:~/lab5.1$
```

Bài 3 name:

```c
#include <stdio.h>;
#include <stdlib.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/errno.h>
#include <unistd.h>
#include <string.h>
#define FIFO1 "/tmp/ff.1"
#define FIFO2 "/tmp/ff.2"
#define PM 0666
extern int errno;
#define PIPE_BUF 4096
int main(int argc, char *argv[]){
        char s1[PIPE_BUF], s2[PIPE_BUF];
        int childpid, readfd, writefd;
        if ((mknod(FIFO1, S_IFIFO | PM, 0) < 0) && (errno != EEXIST)){
                printf("Fail to create FIFO 1. Aborted.\n");
                return -1;
        }
        if ((mknod(FIFO2, S_IFIFO | PM, 0) < 0) && (errno != EEXIST)){
                unlink(FIFO1);
                printf("Fail to create FIFO 2. Aborted.\n");
                return -1;
        }
        childpid = fork();
        if (childpid == 0)
        { // child
                if ((readfd = open(FIFO1, 0)) < 0)
                        perror("Child cannot open readFIFO.\n");
                char arr[4];
                int cnt=0;
                fflush(stdin);
                while (read(readfd, s1, PIPE_BUF) > 0){
                        arr[cnt++] = s1[0];
                }
                printf("%s",s1[0]);
                close(readfd);
                return 1;
```

```
        }
        else if (childpid > 0)
        { // parent
                if ((writefd = open(FIFO1, 1)) < 0)
                        perror("Parent cannot open writeFIFO.\n");
                int i;
                for (i = 0; i < argc; i++)
                {
                        fflush(stdin);
                        scanf("%s",s1);
                        write(writefd, s1, PIPE_BUF);
                }
                while (wait((int *)0) != childpid);
                close(writefd);
                if (unlink(FIFO1) < 0)
                perror("Cannot remove FIFO1.\n");
                return 1;
        }
        else
        {
        printf("Fork failed\n");
        return -1;
        }
}
```