**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**BÁO CÁO BÀI TẬP NMHĐH**

**HK2, 2021-2022**

**Lab 6**

**Nhóm:** 05 **Tổ:** 01

Thành Viên Nhóm:

1: Nguyễn Trần Quang Huy (MSSV: 52000668)

**Muc lục**

[A. PHẦN THỰC HÀNH 3](#_Toc98914391)

[Ví Dụ 1: 3](#_Toc98914392)

[1.1: Code Chương Trình 3](#_Toc98914393)

[1.2: Kết Quả Demo 3](#_Toc98914394)

[Ví Dụ 2: 5](#_Toc98914395)

[2.1: Code chương trình 5](#_Toc98914396)

[2.2 Kết quả Demo 6](#_Toc98914397)

[Ví Dụ 3: 6](#_Toc98914398)

[3.1 Code chương trình 6](#_Toc98914399)

[3.2 Kết quả demo 7](#_Toc98914400)

[Ví Dụ 4: 8](#_Toc98914401)

[4.1 Code chương trình 8](#_Toc98914402)

[4.2 Kết quả demo 10](#_Toc98914403)

[Ví Dụ 5: 11](#_Toc98914404)

[5.1 Code chương trình 11](#_Toc98914405)

[5.2 Kết quả demo 13](#_Toc98914406)

[B. PHẦN BÀI TẬP 14](#_Toc98914407)

[BÀI 1: 14](#_Toc98914408)

[A: Code Chương Trình: 14](#_Toc98914409)

[B: Kết Quả Demo: 16](#_Toc98914410)

[BÀI 2: 16](#_Toc98914411)

[A: Code Chương Trình: 16](#_Toc98914412)

[B: Kết Quả Demo: 18](#_Toc98914413)

[BÀI 3: 20](#_Toc98914414)

[A: Code Chương Trình: 20](#_Toc98914415)

[B: Kết Quả Demo: 23](#_Toc98914416)

[BÀI 4: 23](#_Toc98914417)

[A: Code Chương Trình: 23](#_Toc98914418)

[B: Kết Quả Demo: 25](#_Toc98914419)

[BÀI 5: 26](#_Toc98914420)

[A: Code Chương Trình: 26](#_Toc98914421)

[Kết Quả Demo: 28](#_Toc98914422)

[BÀI 6: 29](#_Toc98914423)

[A: Code Chương Trình: 29](#_Toc98914424)

[B: Kết Quả Demo: 33](#_Toc98914425)

[KẾT LUẬN 35](#_Toc98914426)

[KẾT QUẢ LÀM VIỆC CÁC THÀNH VIÊN TRONG NHÓM 36](#_Toc98914427)

# A. PHẦN THỰC HÀNH

# Ví Dụ 1.1:

* Tiến trình con đọc dữ liệu từ đối số truyền (argv), ghi vào pipe.
* Tiến trình cha đọc từ pipe và xuất ra màn hình.
* Unnamed Pipe

## 1.1.1: Code Chương Trình

#include <stdio.h>

#include <unistd.h>

#include <string.h>

int main(int argc, char \*argv[])

{

char result[100]; // biến lưu dữ liệu tiến trình cha ghi dữ liệu vào pipe

int fp[2];

int pid;

if (argc < 2) // đối số bị sai sẽ dừng chương trình

{

printf("Doi so thieu.\n");

return -1;

}

// điều kiện đúng thì vào

if (pipe(fp) == 0)

{

pid = fork();

if (pid < 0)

{

printf("Fork failed\n");

return -1;

}

else if (pid == 0)

{

printf("Data from child : %s\n", argv[1]);

close(fp[0]);

// tiến trình con đọc, hiển thị dữ liệu, ghi dữ liệu

write(fp[1], argv[1], strlen(argv[1]));

}

else

{

close(fp[1]);

read(fp[0], result, strlen(argv[1])); // tiến trình cha đọc dữ liệu, hiển thị ra màn hình

printf("Read from child : %s\n", result);

}

}

else

{

// kết quả trả về -2

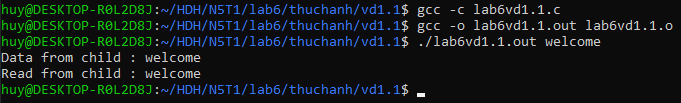
printf("Pipe failed\n");

return -2;

}

}

## 1.1.2: Kết Quả Demo



# Ví Dụ 1.2:

Tương tự ví dụ 1 nhưng sử dụng Named Pipe

## 1.2.1: Code chương trình

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <string.h>

#include <sys/types.h>

#include <sys/stat.h>

#include <sys/errno.h>

// my include

#include <sys/wait.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(); // tạo tiến trình

if (childpid == 0)

{ // child

if ((readfd = open(FIFO1, 0)) < 0)

perror("Child cannot open readFIFO.\n");

if ((writefd = open(FIFO2, 1)) < 0)

perror("Child cannot open writeFIFO.\n");

// đọc dữ liệu từ s2

// in ra màng hình

// nhập dữ liệu từ bàn phím

// ghi vào s1

read(readfd, s2, PIPE\_BUF);

printf("Child read from parent: %s\n", s2);

printf("Enter response: ");

gets(s1);

write(writefd, s1, strlen(s1));

close(readfd);

close(writefd);

return 1;

}

else if (childpid > 0)

{ // parent

if ((writefd = open(FIFO1, 1)) < 0)

perror("Parent cannot open writeFIFO.\n");

if ((readfd = open(FIFO2, 0)) < 0)

perror("Child cannot open readFIFO.\n");

// nhận dữ liệu từ bàn phím

// ghi vào s1 của mình

// in ra màng hình

// đợi tiến trình con hoàn thành

printf("Enter data to FIFO1: ");

gets(s1);

write(writefd, s1, strlen(s1));

read(readfd, , PIPE\_BUF);

printf("Parent read from child: %s\n", s2);

while (wait((int \*)0) != childpid)

;

close(readfd);

close(writefd);

if (unlink(FIFO1) < 0)

perror("Cannot remove FIFO1.\n");

if (unlink(FIFO2) < 0)

perror("Cannot remove FIFO2.\n");

return 1;

}

else

{

printf("Fork failed\n");

return -1;

}

}

## 1.2.2 Kết quả Demo

Text

Description automatically generated

# Ví Dụ 2.1:

* Tạo vùng nhớ chia sẻ
* Các tiến trình phải gắn vùng nhớ chia sẻ vào không gian địa chỉ của mình trước khi sử dụng
* Sau khi sử dụng xong, có thể gỡ vùng nhớ ra khỏi không gian địa chỉ của tiến trình

## 2.1.1 Code chương trình

#include <stdio.h>

#include <unistd.h>

#include <limits.h>

#include <string.h>

#include <stdlib.h>

#include <sys/types.h>

#include <sys/ipc.h>

#include <sys/shm.h>

#define SIZE 256

int main(int argc, char \*argv[])

{

// tạo biến lưu trữ

int \*shm, shmid, k, pid;

key\_t key;

if ((key = ftok(".", 'a')) == -1)

{

perror("Key created.\n");

return 1;

}

// shmid lấy id từ shmget dùng để tạo SM

if (shmid = shmget(key, SIZE, IPC\_CREAT | 0666) == -1)

{

perror("Shared memory created.\n");

return 2;

}

shm = (int \*)shmat(shmid, 0, 0);

pid = fork();

if (pid == 0)

{ // child

shm[0] = atoi(argv[1]);

shm[1] = atoi(argv[2]);

sleep(3);

printf("%d + %d = %d\n", shm[0], shm[1], shm[2]);

// sử dụng để gỡ SM khỏi tiến trình

shmdt((void \*)shm);

shmctl(shmid, IPC\_RMID, (struct shmid\_ds \*)0);

return 0;

}

else if (pid > 0)

{ // parent

sleep(1);

shm[2] = shm[1] + shm[0];

// sử dụng để gỡ SM khỏi tiến trình

shmdt((void \*)shm);

sleep(5);

return 0;

}

else

{

perror("Fork failed.");

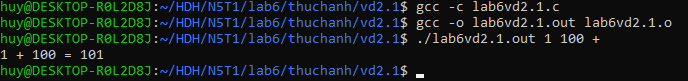
return 4;

}

return 0;

}

## 3.2 Kết quả demo



# Ví Dụ 3.1:

Message queues

## 3.1.1 Code chương trình

#include <stdio.h>

#include <sys/types.h>

#include <sys/ipc.h>

int main(int argc, char \*argv[])

{

key\_t key;

char i;

// truyền vào đối số

for (i = 'a'; i < 'f'; i++)

{

key = ftok(".", i);

printf("Proj = %c key = %d.\n", i, key);

// proj là một ký tự định danh

}

return 0;

}

## 3.1.2 Kết quả demo

Text

Description automatically generated

# Ví Dụ 3.2:

## 3.2.1 Code chương trình

#include <stdio.h>

#include <unistd.h>

#include <limits.h>

#include <sys/types.h>

#include <sys/ipc.h>

#include <sys/msg.h>

#define MAX 5

int main(int argc, char \*argv[])

{

FILE \*fin;

char buffer[PIPE\_BUF], proj = 'A';

int i, n, mid[MAX];

key\_t key;

for (i = 0; i < MAX; i++, proj++)

{

key = ftok(".", proj);

if (mid[i] = msgget(key, IPC\_CREAT | 0666) == -1)

{

perror("Queue created.\n");

return 1;

}

}

// mở file

fin = popen("ipcs", "r");

while ((n = read(fileno(fin), buffer, PIPE\_BUF)) > 0)

{

write(fileno(stdout), buffer, n);

}

// đống file

pclose(fin);

for (i = 0; i < MAX; i++)

{

// xóa MQ có id tương ứng

msgctl(mid[i], IPC\_RMID, (struct msqid\_ds \*)0);

}

return 0;

}

## 3.2.2 Kết quả demo

Text

Description automatically generated

Table

Description automatically generated

# Ví Dụ 3.3:

## 3.3.1 Code chương trình

#include <stdio.h>

#include <unistd.h>

#include <limits.h>

#include <sys/types.h>

#include <sys/ipc.h>

#include <sys/msg.h>

#define MAX 5

int main(int argc, char \*argv[])

{

FILE \*fin;

char buffer[PIPE\_BUF], proj = 'A';

int i, n, mid[MAX];

key\_t key;

for (i = 0; i < MAX; i++, proj++)

{

key = ftok(".", proj);

// dùng để tạo MQ

if (mid[i] = msgget(key, IPC\_CREAT | 0666) == -1)

{

perror("Queue created.\n");

return 1;

}

}

// mở file

fin = popen("ipcs", "r");

while ((n = read(fileno(fin), buffer, PIPE\_BUF)) > 0)

write(fileno(stdout), buffer, n);

// đống file

pclose(fin);

for (i = 0; i < MAX; i++)

msgctl(mid[i], IPC\_RMID, (struct msqid\_ds \*)0);

return 0;

}

## 3.3.2 Kết quả demo

Text

Description automatically generated

# Ví Dụ 3.4:

Thao tác trên MQ

## 3.4.1 Code chương trình

#include <stdio.h>

#include <unistd.h>

#include <limits.h>

#include <sys/types.h>

#include <sys/ipc.h>

#include <sys/msg.h>

#include <string.h>

#include <stdlib.h>

struct Message

{

int mtype;

char \*content;

};

int main(int argc, char \*argv[])

{

int mid, id, n, pid;

key\_t key;

struct Message msg;

msg.content = (char \*)malloc(BUFSIZ);

if ((key = ftok(".", 'a')) == -1)

{

perror("Key created.\n");

return 1;

}

if (mid = msgget(key, IPC\_CREAT | 0666) == -1)

{

perror("Queue created.\n");

return 2;

}

pid = fork();

if (pid == 0)

{ // child

msg.mtype = 10;

msg.content = argv[1];

n = strlen(msg.content);

printf("%s\n", msg.content);

n += sizeof(msg.mtype);

if (msgsnd(mid, &msg, n, 0) == -1)

{

perror("Message sent.\n");

return 4;

}

sleep(5);

printf("Child receive from parent: \n");

if (n = msgrcv(mid, &msg, BUFSIZ, 11, 0) == -1)

{

perror("Message received.\n");

return 5;

}

msg.content[strlen(msg.content)] = 0;

printf("%s\n", msg.content);

return 0;

}

else if (pid > 0)

{ // parent

sleep(1);

printf("Child receive from parent: \n");

if (n = msgrcv(mid, &msg, BUFSIZ, 10, 0) == -1)

{

perror("Message received.\n");

return 5;

}

printf("Parent received from child: \n");

msg.content[strlen(msg.content)] = 0;

printf("%s\n", msg.content);

printf("-- -- -- -- -- -- -- -- -- - \n");

printf("Message from parent:\n");

msg.content = argv[2];

n = strlen(msg.content);

msg.mtype = 11;

n += sizeof(msg.mtype);

if (msgsnd(mid, &msg, n, 0) == -1)

{

perror("Message sent.\n");

return 4;

}

sleep(10);

return 0;

}

else

{

perror("Fork failed.");

return 3;

}

return 0;

}

## 3.4.2 Kết quả demo

Text

Description automatically generated

# B. PHẦN BÀI TẬP

# KẾT LUẬN

Sau khi học và hoàn thành phần **LAB 6** nhóm thu được kết sau:

* Sử dụng kỹ thuật pipes
* Sử dụng kỹ thuật messages

# KẾT QUẢ LÀM VIỆC CÁC THÀNH VIÊN TRONG NHÓM

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| HỌ TÊN SV | MÃ SV | VAI TRÒ | CÔNG VIỆC ĐƯỢC PHÂN CÔNG | HT |
| Nguyễn Trần Quang Huy | 52000668 | TN | Làm cả phần thực hành và bài tập | 100% |

CHÚ THÍCH:

TN: TRƯỞNG NHÓM

TV: THÀNH VIÊN

HT: MỨC ĐỘ HOÀN THÀNH CÔNG VIỆC ĐƯỢC GIAO