**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**BÁO CÁO BÀI TẬP NMHĐH**

**HK2, 2021-2022**

**Lab 3**

**Nhóm:** 05 **Tổ:** 01

Thành Viên Nhóm:

1: Nguyễn Trần Quang Huy (MSSV: 52000668)

Muc lục

[A. PHẦN THỰC HÀNH 5](#_Toc97714268)

[Ví Dụ 1: 5](#_Toc97714269)

[1.1: Code Chương Trình 5](#_Toc97714270)

[1.2: Kết Quả Demo 5](#_Toc97714271)

[Ví Dụ 2: 5](#_Toc97714272)

[2.1: Code chương trình 5](#_Toc97714273)

[2.2 Kết quả Demo 6](#_Toc97714274)

[Ví Dụ 3: 6](#_Toc97714275)

[3.1 Code chương trình 6](#_Toc97714276)

[3.2 Kết quả demo 7](#_Toc97714277)

[Ví Dụ 4: 8](#_Toc97714278)

[4.1 Code chương trình 8](#_Toc97714279)

[4.2 Kết quả demo 9](#_Toc97714280)

[Ví Dụ 5: 10](#_Toc97714281)

[5.1 Code chương trình 10](#_Toc97714282)

[5.2 Kết quả demo 12](#_Toc97714283)

[B. PHẦN BÀI TẬP 13](#_Toc97714284)

[BÀI 1: 13](#_Toc97714285)

[A: Code Chương Trình: 13](#_Toc97714286)

[B: Kết Quả Demo: 14](#_Toc97714287)

[BÀI 2: 14](#_Toc97714288)

[A: Code Chương Trình: 14](#_Toc97714289)

[B: Kết Quả Demo: 16](#_Toc97714290)

[BÀI 3: 18](#_Toc97714291)

[A: Code Chương Trình: 18](#_Toc97714292)

[B: Kết Quả Demo: 20](#_Toc97714293)

[BÀI 4: 21](#_Toc97714294)

[A: Code Chương Trình: 21](#_Toc97714295)

[B: Kết Quả Demo: 21](#_Toc97714296)

[BÀI 5: 22](#_Toc97714297)

[A: Code Chương Trình: 22](#_Toc97714298)

[Kết Quả Demo: 23](#_Toc97714299)

[BÀI 6: 24](#_Toc97714300)

[A: Code Chương Trình: 25](#_Toc97714301)

[B: Kết Quả Demo: 26](#_Toc97714302)

[BÀI 7: 26](#_Toc97714303)

[A: Code Chương Trình: 27](#_Toc97714304)

[B: Kết Quả Demo: 28](#_Toc97714305)

[KẾT LUẬN 29](#_Toc97714306)

[KẾT QUẢ LÀM VIỆC CÁC THÀNH VIÊN TRONG NHÓM 30](#_Toc97714307)

# A. PHẦN THỰC HÀNH

# Ví Dụ 1:

Xem thông tin số hiệu tiến trình

## 1.1: Code Chương Trình

// P1.c

#include <stdio.h>

#include <unistd.h>

int main() {

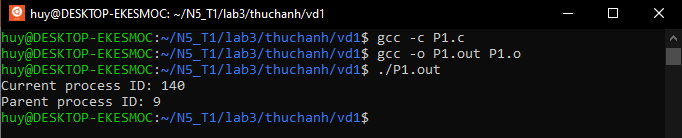
printf("Current process ID: %d\n", getpid());

printf("Parent process ID: %d\n", getppid());

return 0;

}

## 1.2: Kết Quả Demo



# Ví Dụ 2:

Sử dụng hàm fork

## 2.1: Code chương trình

// P2.c

#include <stdio.h>

#include <sys/types.h>

#include <unistd.h>

int main() {

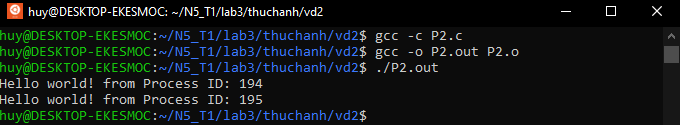
fork();

printf("Hello world! from Process ID: %d\n", getpid());

return 0;

}

## 2.2 Kết quả Demo



# Ví Dụ 3:

Template tiến trình cha và tiến trình con

## 3.1 Code chương trình

// P3.c

#include <stdio.h>

#include <unistd.h>

int main() {

pid\_t pid;

pid = fork();

if (pid < 0)

printf("fork error !No child created\n");

else if (pid == 0) {

printf("Hello from Child !\n");

printf("Child ID is % d !\n", getpid());

} else {

printf("Hello from Parent !\n");

printf("Parent ID is % d !\n", getpid());

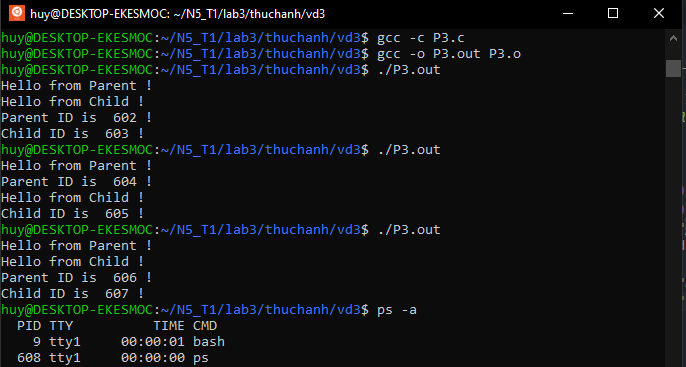
}

return 0;

}

**Lưu ý thứ tự các câu in ra màn hình có thể thay đổi khác nhau ở các lần chạy, hãy giải thích tại sao?** Là tại vì tiến trình con và tiến trình cha chạy song song với nhau nên tùy vào tốc độ sử lý mà kết quả in ra sẽ có thể khác nhau.

## 3.2 Kết quả demo



# Ví Dụ 4:

Tạo nhiều tiến trình con

## 4.1 Code chương trình

#include <sys/wait.h>

#include <stdlib.h>

#include <unistd.h>

#include <stdio.h>

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

int pnum, count, retval, child\_no;

pnum = atoi(argv[1]);

if (pnum <= 0) {

printf("So tien trinh con phai lon hon 0");

return -1;

} else {

retval = 1;

for (count = 0; count < pnum; count++) {

if (retval != 0)

retval = fork();

else

break;

}

if (retval == 0) {

child\_no = count;

printf("Tien trinh % d, PID % d\n ", child\_no,

getpid());

} else {

for (count = 0; count < pnum; count++)

wait(NULL);

printf("Tien trinh cha PID % d", getpid());

}

}

wait(2); // chờ một trong những đứa con.

printf("\n");

return 0;

}

## 4.2 Kết quả demo

Text

Description automatically generated

# Ví Dụ 5:

Kill tiến trình

## 5.1 Code chương trình

#include <stdio.h>

#include <unistd.h>

#include <sys/wait.h>

#include <sys/types.h>

int main() {

int i;

int pid = fork();

if (pid == 0)

for (i = 0; i < 20; i++)

printf("I am Child\n");

else {

signal(SIGCHLD, SIG\_IGN);

printf("I am Parent\n");

while (1) {}

}

}

# 5.2 Kết quả demo

Text

Description automatically generated

# B. PHẦN BÀI TẬP

# BÀI 1:

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

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

## A: Code Chương Trình:

#include <stdio.h>

#include <unistd.h>

#include <stdlib.h>

int main(int argc, char \*\* argv) {

pid\_t pid;

pid = fork();

int n = atoi(argv[1]);

if (pid < 0) {

printf("fork error !No child created\n");

} else if (pid == 0) { // tiến trình con tính tổng ước

int tongUoc = 0;

for (int i = 1; i <= n; i++) {

if (n % i == 0) {

tongUoc += i;

}

}

printf("Tong uoc cua %d la %d\n", n, tongUoc);

} else { // tiến trình cha tính tổng

int tong = 0;

for (int i = 1; i <= n; i++) {

tong += i;

}

printf("Tong tu 1 -> %d la %d\n", n, tong);

}

return 0;

}

## B: Kết Quả Demo:

Text

Description automatically generated

# BÀI 2:

Sử dụng lời gọi system() để viết chương trình thực thi các lệnh Linux đã học tại LAB01.

## A: Code Chương Trình:

#include <stdio.h>

#include <stdlib.h>

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

int re;

printf("Call system to execute [ls -a]\n");

re = system("ls -a"); // gọi hệ thống thực thi ls -a

if (re != -1)

printf("System call ls -a is done!\n\n");

printf("Call system to execute [touch file1.txt]\n");

re = system("touch file1.txt"); // gọi hệ thống thực thi tạo file

re = system("ls -l");

if (re != -1)

printf("System call touch file1.txt is done!\n\n");

printf("Call system to execute [mkdir file1]\n");

re = system("mkdir file1"); // gọi hệ thống thực thi tạo thư mục

re = system("ls -l");

if (re != -1)

printf("System call mkdir file1 is done!\n\n");

printf("Call system to execute [rm file1.txt]\n");

re = system("rm file1.txt"); // gọi hệ thống thực thi xóa file

re = system("ls -l");

if (re != -1)

printf("System call rm file1.txt is done!\n\n");

printf("Call system to execute [rm -r file1]\n");

re = system("rm -r file1"); // gọi hệ thống thực thi xóa thư mục

re = system("ls -l");

if (re != -1)

printf("System call rm -r file1 is done!\n\n");

printf("Call system to execute [ps -a]\n");

re = system("ps -a"); // hiện thị tất cả tiến trình

if (re != -1)

printf("System call ps -a is done!\n\n");

return 0;

}

## B: Kết Quả Demo:

Text

Description automatically generated

Text

Description automatically generated with medium confidence

# BÀI 3:

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

## A: Code Chương Trình:

#include <stdio.h>

#include <unistd.h>

#include <stdlib.h>

#include <sys/wait.h>

#include <sys/types.h>

int main() { // tạo tiến trình B là tiến trình con của A sau đó trong B tạo ra E D con của B tưởng tự cho C

pid\_t pid;

pid = fork();

if (pid < 0) {

printf("fork error !No child created\n");

} else if (pid == 0) {

printf("My ID B: %d and my Parent: %d\n", getpid(), getppid());

pid = fork();

if (pid < 0) {

printf("fork error !No child created\n");

} else if (pid == 0) {

printf("My ID D: %d and my Parent: %d\n", getpid(), getppid());

sleep(1);

exit(0);

}

pid = fork();

if (pid < 0) {

printf("fork error !No child created\n");

} else if (pid == 0) {

printf("My ID E: %d and my Parent: %d\n", getpid(), getppid());

sleep(1);

exit(0);

}

sleep(1);

exit(0);

}

pid = fork();

if (pid < 0) {

printf("fork error !No child created\n");

} else if (pid == 0) {

printf("My ID C: %d and my Parent: %d\n", getpid(), getppid());

pid = fork();

if (pid < 0) {

printf("fork error !No child created\n");

} else if (pid == 0) {

printf("My ID H: %d and my Parent: %d\n", getpid(), getppid());

sleep(1);

exit(0);

}

sleep(1);

exit(0);

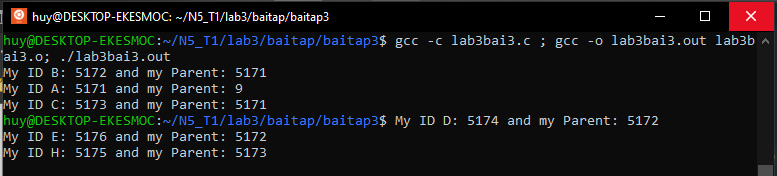
}

printf("My ID A: %d and my Parent: %d\n", getpid(), getppid());

return 0;

}

## B: Kết Quả Demo:



# BÀI 4:

Viết một chương trình lặp vô tận với lời gọi while(1); và thực thi nó. Đưa tiến trình này vào “background” thông qua việc gửi tín hiệu SIGTSTP bằng cách nhấn Ctrl + Z. Sử dụng lệnh ps để xác định PID của nó và sử dụng kill để kết thúc nó.

# A: Code Chương Trình:

#include <stdio.h>

#include <unistd.h>

#include <stdlib.h>

#include <sys/wait.h>

#include <sys/types.h>

int main() {

while (1) {

}

return 0;

}

## B: Kết Quả Demo:

A screenshot of a computer

Description automatically generated

# BÀI 5:

Programming Problems 3.18 in book [1]

Viết chương trình mà khi chạy nó sinh ra tiến trình con, để tiến trình con trở thành zombie. Tiến trình zombie này cần tồn tại trong hệ thống tối thiểu 10 giây (bằng cách dùng lời gọi sleep(10)). Sau đó dùng lệnh ps –l để xem trạng thái các tiến trình. Kết liễu zombie này bằng cách xác định PID của tiến trình cha nó và sử dụng lệnh kill.

## A: Code Chương Trình:

#include <stdio.h>

#include <unistd.h>

#include <sys/wait.h>

#include <sys/types.h>

int main() {

int i;

int pid = fork();

if (pid == 0) {

for (i = 0; i < 20; i++) {

printf("I am Child\n");

}

sleep(10);

} else {

// signal(SIGCHLD, SIG\_IGN);

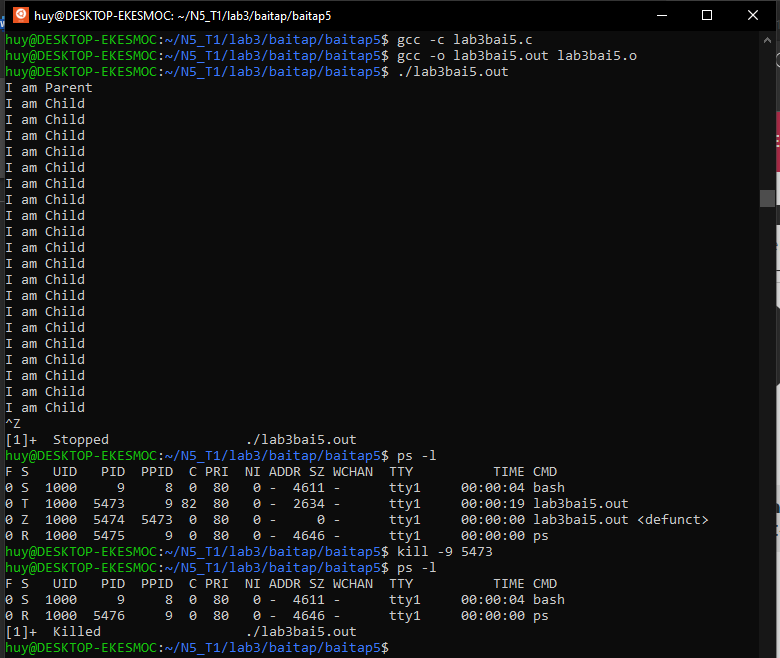
printf("I am Parent\n");

while (1) {}

}

}

## Kết Quả Demo:



# BÀI 6:

Programming Problems 3.19 in book [1]

Viết chương trình xác định thời gian cần thiết để thực thi một lệnh, lệnh này truyền vào qua phần đối số như minh hoạ 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)

## A: Code Chương Trình:

#include <stdio.h>

#include <unistd.h>

#include <sys/wait.h>

#include <sys/types.h>

#include <stdlib.h>

int main(int argc, char \*\* argv) {

int i;

int pid = fork();

if (pid == 0) {

int re;

printf("Call system to execute %s\n", argv[1]);

re = system(argv[1]);

if (re != -1)

printf("System call done!\n\n");

} else {

wait(NULL);

}

return 0;

}

## B: Kết Quả Demo:

Text

Description automatically generated

# BÀI 7:

Programming Problems 3.21 in book [1]

Phỏng đoán Collatz1 tin rằng dãy số sinh ra sẽ luôn tiến về 1 với bất kỳ số nguyên dương nào được tạo ra ở bước đầu. Dãy số được tạo ra theo giải thuật sau:

𝑛 = 𝑛 / 2 𝑛ế𝑢 𝑛 𝑙à 𝑠ố 𝑐ℎẳ𝑛

n = 3 ∗ 𝑛 + 1 𝑛ế𝑢 𝑙à 𝑠ố 𝑙ẻ

Ví dụ với n=35, dãy số sinh ra là 35, 106, 53, 160, 80, 40, 20, 10, 5, 16, 8, 4, 2, 1

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, tronng lúc tiến trình cha cần chờ tiến trình con hoàn thành thông qua lời gọi wait().

## A: Code Chương Trình:

#include <stdio.h>

#include <unistd.h>

#include <sys/wait.h>

#include <sys/types.h>

#include <stdlib.h>

int main(int argc, char \*\* argv) {

int i;

int pid = fork();

int n = atoi(argv[1]);

if (pid == 0) {

printf("%d", n);

while (1) { // trong khi n != 1 thì cứ thực hiện theo công thức khi bằng 1 thì dừng lại

if (n != 1) {

printf(", ");

}

if (n == 1) {

break;

}

if (n % 2) {

n = n \* 3 + 1;

printf("%d", n);

} else {

n = n / 2;

printf("%d", n);

}

}

printf("\nKet thuc tien trinh con \n");

} else {

wait(NULL);

}

return 0;

}

## B: Kết Quả Demo:

Text

Description automatically generated

# KẾT LUẬN

Sau khi học và hoàn thành phần lab 2 nhóm thu được kết sau:

* Cách tạo một tiến trình con.
* Cách quản lý các tiến trình được tạo ra và kill những tiến trình là zombie.
* Học được về phỏng đoán Collatz khá hay.

# 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