|  |
| --- |
| Họ và tên:  Mã số sinh viên:  Lớp: |

HỆ ĐIỀU HÀNH  
BÁO CÁO LAB 3

**CHECKLIST**

**3.5. BÀI TẬP THỰC HÀNH**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **BT 1** | **BT 2** | **BT 3** | **BT 4** |
| **Trình bày cách làm** |  |  |  |  |
| **Chụp hình minh chứng** |  |  |  |  |
| **Giải thích kết quả** |  |  |  |  |

**3.6. BÀI TẬP ÔN TẬP**

|  |  |
| --- | --- |
|  | **BT 1** |
| **Trình bày cách làm** |  |
| **Chụp hình minh chứng** |  |
| **Giải thích kết quả** |  |

**Tự chấm điểm:**

*\*Lưu ý: Xuất báo cáo theo định dạng PDF, đặt tên theo cú pháp:* ***<MSSV>\_LAB3.pdf***

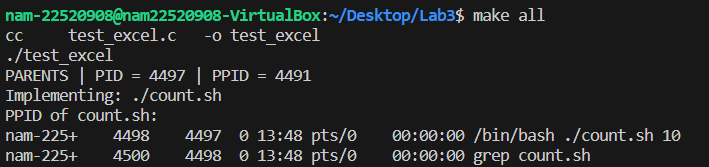
**2.5. BÀI TẬP THỰC HÀNH**

# Thực hiện Ví dụ 3-1, Ví dụ 3-2, Ví dụ 3-3, Ví dụ 3-4 giải thích code và kết quả nhận được?

Trả lời...

A screen shot of a computer

Description automatically generated



A screen shot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated  
A black screen with green and white text

Description automatically generated

# Viết chương trình time.c thực hiện đo thời gian thực thi của một lệnh shell. Chương trình sẽ được chạy với cú pháp "./time <command>" với <command> là lệnh shell muốn đo thời gian thực thi.

Ví dụ:

$ ./time ls

time.c

time

Thời gian thực thi: 0.25422  
**Gợi ý: Tiến trình cha gọi hàm fork() tạo ra tiến trình con rồi wait(). Tiến trình con gọi hàm gettimeofday() để lấy mốc thời gian trước khi thực thi lệnh shell, sau đó sử dụng hàm execl() để thực thi lệnh. Sau khi tiến trình con kết thúc, tiến trình cha tiếp tục gọi hàm gettimeifday() một lần nữa để lấy mốc thời gian sau khi thực thi lệnh shell và tính toán.**

Trả lời...

# Viết một chương trình làm bốn công việc sau theo thứ tự:

# In ra dòng chữ: “Welcome to IT007, I am <your\_Student\_ID>!”

# Thực thi file script count.sh với số lần đếm là 120

# Trước khi count.sh đếm đến 120, bấm CTRL+C để dừng tiến trình này

# Khi người dùng nhấn CTRL+C thì in ra dòng chữ: “count.sh has stoppped”

Trả lời...

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <signal.h>

// Hàm xử lý khi người dùng nhấn Ctrl+C

void handle\_ctrlc(int sig) {

printf("\ncount.sh has stopped\n");

exit(0);

}

int main() {

// Lấy Student ID từ người dùng

char studentID[50];

printf("Nhập Student ID của bạn: ");

scanf("%s", studentID);

printf("Welcome to IT007, I am %s!\n", studentID);

// Chạy file script count.sh với số lần đếm là 120

pid\_t pid = fork();

if (pid == 0) {

execl("./count.sh", "count.sh", "120", NULL);

} else {

// Đăng ký xử lý tín hiệu khi người dùng nhấn Ctrl+C

signal(SIGINT, handle\_ctrlc);

// Chờ tiến trình con kết thúc

wait(NULL);

}

return 0;

}

# Viết chương trình mô phỏng bài toán Producer - Consumer như sau:

# Sử dụng kỹ thuật shared-memory để tạo một bounded-buffer có độ lớn là 10 bytes.

# Tiến trình cha đóng vai trò là Producer, tạo một số ngẫu nhiên trong khoảng [10, 20] và ghi dữ liệu vào buffer

# Tiến trình con đóng vai trò là Consumer đọc dữ liệu từ buffer, in ra màn hình và tính tổng

# Khi tổng lớn hơn 100 thì cả 2 dừng lại

Trả lời...

**2.6. BÀI TẬP ÔN TẬP**

# Phỏng đoán Collatz xem xét chuyện gì sẽ xảy ra nếu ta lấy một số nguyên dương bất kỳ và áp dụng theo thuật toán sau đây: Phỏng đoán phát biểu rằng khi thuật toán này được áp dụng liên tục, tất cả số nguyên dương đều sẽ tiến đến 1. Ví dụ, với n = 35, ta sẽ có chuỗi kết quả như sau:

35, 106, 53, 160, 80, 40, 20, 10, 5, 16, 8, 4, 2, 1  
Viết chương trình C sử dụng hàm fork() để tạo ra chuỗi này trong tiến trình con. Số bắt đầu sẽ được truyền từ dòng lệnh. Ví dụ lệnh thực thi ./collatz 8 sẽ chạy thuật toán trên n = 8 và chuỗi kết quả sẽ ra là 8, 4, 2, 1. Khi thực hiện, tiến trình cha và tiến trình con chia sẻ một buffer, sử dụng phương pháp bộ nhớ chia sẻ, hãy tính toán chuỗi trên tiến trình con, ghi kết quả vào buffer và dùng tiến trình cha để in kết quả ra màn hình. Lưu ý, hãy nhớ thực hiện các thao tác để kiểm tra input là số nguyên dương.

Trả lời...