|  |
| --- |
| Họ và tên: Đoàn Phương Nam - Trương Thiên Lộc - Phạm Duy Long - Trần Quỳnh Thy  Mã số sinh viên: 22520908 – 21520330 – 20521573 – 22521461  Lớp: IT007.O11.2 |

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:** 10/10

*\*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...

* Ví dụ 3-1

+ Hình minh chứng:

A computer screen shot of text

Description automatically generated

A screenshot of a computer

Description automatically generated

+ Giải thích:

. \_\_pid\_t: Là kiểu dữ liệu của số hiệu tiến trình.

. fork(): Tạo tiến trình mới là tiến trình con của tiến trình hiện tại

. if (pid > 0): Nếu tiến trình là tiến trình cha.

. if (pid == 0): Nếu tiến trình là tiến trình con.

. getpid(): Lấy số hiệu tiến trình hiện tại.

. getppid(): Lấy số hiệu tiến trình cha của tiến trình con.

. argc: Số lượng tham số

. argv[]: Mảng chứa các tham số

. wait(NULL): đợi tiến trình con kết thúc

Giải thích kết quả:

Vì ta nhập vào 3 tham số nên ở tiến trình cha argc = 4 > 2 nên sẽ in ra dòng   
“PARENTS | There are %d arguments\n”, ở tiến trình con thì sẽ in ra giá trị của 3 tham số nhập vào.

* Ví dụ 3-2

+ Hình minh chứng:

A screen shot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a black screen

Description automatically generated

+ Giải thích:

Đoạn code trên tạo một tiến trình con và thực thi shell script trong tiến trình con

Lệnh execl: thay tiến trình hiện tại bằng tiến trình khác (ở đoạn code trên có nghĩa là thay tiến trình con thành tiến trình count.sh). Vì vậy, các dòng printf sau lệnh execl sẽ không được thực thi vì lúc này tiến trình con đã chuyển sang thành count.sh)

* Ví dụ 3-3

+ Hình minh chứng:

A computer screen shot of a program code

Description automatically generated

A screenshot of a computer

Description automatically generated

+ Giải thích:

Lệnh system(“./count.sh 10”) sẽ tạo tiến trình mới là sh với lệnh là “./count.sh 10”

Trong file count.sh, dòng #!/bin/bash được đọc và sẽ tạo tiến trình /bin/bash ./count.sh 10

Vì tạo mới hoàn toàn 1 tiến trình, nên các dòng code sau lệnh system vẫn được thực thi bình thường

* Ví dụ 3-4

+ Hình minh chứng:

A screen shot of a computer program

Description automatically generated

A screen shot of a computer program

Description automatically generated



+ Giải thích:

fd = shm\_open(name, O\_CREAT | O\_RDWR, 0666) : Khởi tạo vùng nhớ được chia sẻ

ftruncate(fd, SIZE): Cài đặt độ lớn của vùng nhớ chia sẻ

mmap(0, SIZE, PROT\_READ | PROT\_WRITE, MAP\_SHARED, fd,

0): Khởi tạo file ánh xạ bộ nhớ có đối tượng vùng nhớ chia sẻ

munmap(ptr, SIZE);

close(fd);

shm\_unlink(name): Thu hồi bộ nhớ

test\_shm\_A.c: strcpy(ptr, “Hello Process B”): Ghi vào vùng nhớ chia sẻ chuỗi “Hello Process B”, while (strncmp(ptr, “Hello Process B”), 15 == 0) kiểm tra vùng nhớ chia sẻ đã được thay đổi hay chưa, sleep(1): đợi 1 giây

test\_shm\_B.c: strcpy(ptr, “Hello Process A”): Ghi vào vùng nhớ chia sẻ chuỗi “Hello Process A”, sleep(5): đợi 5 giây

Vì vậy ở bài này khi thực thi test\_shm\_A, dòng while sẽ được lặp lại liên tục đến khi test\_shm\_B được thực thi, đến khi vùng nhớ chia sẻ được đổi thành “Hello Process A” thì dòng while sẽ ngừng.

# 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...

**Bước 1: Xác thực Tham số Đầu Vào**

Kiểm tra xem có đủ tham số dòng lệnh để chạy chương trình không. Nếu không, chương trình sẽ in hướng dẫn cách sử dụng và kết thúc.

**Bước 2: Lấy Thời Gian Bắt Đầu**

Sử dụng hàm gettimeofday() để lấy thời gian khi chương trình bắt đầu thực hiện lệnh shell.

**Bước 3: Tạo Tiến Trình Con**

Sử dụng fork() để tạo một tiến trình con mới.

**Bước 4: Thực Thi Lệnh Shell**

Trong tiến trình con, chương trình sử dụng execl() để thực thi lệnh shell được truyền qua tham số dòng lệnh.

**Bước 5: Đợi Tiến Trình Con Kết Thúc**

Trong tiến trình cha, chương trình sử dụng wait() để đợi tiến trình con kết thúc thực thi lệnh.

**Bước 6: Lấy Thời Gian Kết Thúc**

Sau khi tiến trình con kết thúc, sử dụng gettimeofday() để lấy thời gian tại thời điểm tiến trình con hoàn thành công việc.

**Bước 7: Tính Thời Gian Thực Thi**

Tính toán thời gian thực thi bằng cách lấy sự chênh lệch giữa thời gian bắt đầu và thời gian kết thúc.

**Bước 8: Xuất Kết Quả**

Kết quả về thời gian thực thi của lệnh shell được in ra màn hình.

Chương trình này sử dụng các hàm hệ thống để tạo và quản lý các tiến trình con, sau đó đo và báo cáo thời gian thực thi của lệnh shell được chạy.

A screen shot of a computer code

Description automatically generated

A screen shot of a computer program

Description automatically generated

Kết quả của chương trình sau khi gõ lệnh ./time ls:

A screen shot of a computer screen

Description automatically generated

**Giải thích kết quả:**

- Khi chạy ./time ls, chương trình của bạn thực hiện theo thứ tự sau:

- Bắt đầu đo thời gian (gettimeofday(&start, NULL)).

- Tạo một tiến trình con để thực hiện lệnh ls.

- Tiến trình con chạy lệnh ls để hiển thị danh sách các file trong thư mục.

- Tiến trình cha đợi tiến trình con hoàn thành (bằng cách sử dụng wait()).

- Sau khi tiến trình con kết thúc, tiến trình cha đo thời gian (gettimeofday(&end,NULL)) và tính toán thời gian thực thi.

- Cuối cùng, chương trình in ra danh sách các file và sau đó in ra thời gian thực thi.

# 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...

**Ý tưởng**

Sử dụng các hàm hệ thống.

**A screen shot of a computer screen

Description automatically generated**

Bước 1: In ra thông điệp chào mừng và lấy thông tin Student ID từ người dùng

1. Sử dụng printf để hiển thị thông điệp chào mừng.
2. Sử dụng scanf để lấy thông tin Student ID từ người dùng lưu vào mảng char đã khai báo từ trước.

A computer screen shot of text

Description automatically generated

Bước 2: Thực thi file script count.sh

1. Sử dụng fork() để tạo một tiến trình con.
2. Trong tiến trình con:
3. Sử dụng execl() để thực thi file script count.sh với tham số là 120.

A black background with white text

Description automatically generated

Bước 3: Xử lý tín hiệu Ctrl+C

Trong tiến trình cha:

1. Đăng ký xử lý tín hiệu Ctrl+C bằng hàm signal(SIGINT, handler\_function).
2. Khi nhận tín hiệu Ctrl+C, chạy hàm xử lý tín hiệu để in ra thông điệp "count.sh has stopped" và kết thúc chương trình.

A screen shot of a computer program

Description automatically generated

A computer screen shot of text

Description automatically generated

Kết quả:

* Nhập thử một mssv.
* Theo dõi file count.txt, dừng tiến trình ở 11 bằng Ctrl+C. Terminal trả về thông báo “count.sh has stopped” và kết thúc chương trình

A screenshot of a computer

Description automatically generated

# 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

1. **Giả định:**

* Vì độ dài của buffer là 10 bytes, nên có thể chia buffer thành 10 phần với mỗi phần 1 byte.
* Thực tế, khi tạo ra một buffer, phải đồng thời cấp phát ô nhớ để lưu các thông tin của buffer như điểm start (điểm bắt đầu), end (điểm kết thúc)… Nên lượng tài nguyên cần để tạo ra một buffer thực tế sẽ lớn hơn độ dài của buffer.

1. **Ý tưởng:**

* Định nghĩa cấu trúc Buffer chứa một buffer và các thông tin liên quan của nó. Trong đó phải có một thông tin cho biết buffer có còn được sử dụng hay không (isFinish).
* Tạo một shared memory và khởi tạo một buffer rỗng.
* Tạo function Producer và Consumer:
  + Producer: Khi buffer vẫn còn được sử dụng, thực hiện công việc tạo ra số ngẫu nhiên và đẩy số ngẫu nhiên đó vào trong buffer.
  + Consumer: Khi có sự thay đổi trong buffer, thực hiện công việc in ra các số có trong buffer và tính tổng. Nếu tổng các số trong buffer lớn hơn 100, sẽ chỉnh thông tin buffer lại thành “không còn được sử dụng”.
* Dùng lệnh fork() để tạo tiến trình con.
* Tiến trình cha sẽ gọi function Producer và tiến trình con sẽ gọi function Consumer.
* Cuối cùng là đóng kết nối và giải phóng bộ nhớ.

1. **Code:**

A screenshot of a computer code

Description automatically generated

*Khai báo thư viện*

A computer screen shot of text

Description automatically generated

*Khai báo cấu trúc của buffer*

* Bao gồm:
  + **buffer**: chứa buffer.
  + **length**: chứa độ dài của buffer.
  + **isFinish**: cho biết đã không còn được sử dụng hay không. (**false** 🡪 buffer đang được sử dụng; **true** 🡪 buffer không còn được sử dụng).

A screen shot of a computer

Description automatically generated

*Khai báo các hàm helper*

* Ở đây sẽ sử dụng hàm **min** để xử lí bounded cho buffer. Được sử dụng trong function xử lí của Consumer.

A computer screen shot of text

Description automatically generated

*Hàm xử lí của Producer*

* Input nhận vào một con trỏ ánh xạ đến vùng shared memory chứa buffer.
* Điều kiện của dòng while có nghĩa là nếu buffer vẫn còn được sử dụng thì vẫn thực hiện các lệnh bên trong.
* Bên trong dòng while bao gồm:
  + Tạo ra số ngẫu nhiên.
  + Thêm số vào buffer. Có hỗ trợ cơ chế bounded.
  + Sau khi thêm số vào buffer thì chờ cho bên **Consumer** xử lí rồi mới thêm số khác vào. Thời gian chờ đợi ở code trên là **1** **giây**.
* Sau khi dòng while đã dừng thì chờ cho bên Consumer off trước rồi Producer mới thông báo off sau.

A screen shot of a computer program

Description automatically generated

*Hàm xử lí của Consumer*

* Input đầu vào là một con trỏ ánh xạ đến shared memory chứa buffer.
* Để xét được buffer có bị thay đổi dữ liệu không thì cần có một dependency để so sánh sự thay đổi. Ở đây chọn thuộc tính **length** làm dependency (do buffer chỉ có thể được thêm, chứ không được chỉnh sửa nên sự thay đổi dễ thấy nhất là độ dài của buffer). Vì thế, cần có một biến để lưu lại giá trị length ở vòng lặp trước đó để so sánh.
* Đây là một listener nên cần phải lặp vô tận để bắt các sự thay đổi. Vì vậy, nên vòng lặp sẽ là **while(1).**
* Bên trong dòng while:
  + Nếu dòng length mới không khác length cũ, thì sẽ không thực hiện việc xử lí.
  + Gán length = length mới.
  + Dòng for có tác dụng in ra các số có trong buffer và tính tổng.
  + Sau khi đã tính tổng, in tổng ra.
  + Xét nếu tổng đã lớn hơn 100 thì cho **isFinish = true** (nghĩa là buffer này không còn được sử dụng nữa). Sau đó thông báo Consumer off và **return**.

A computer screen shot of text

Description automatically generated

*Hàm main: set up buffer và random*

* Tạo ra một shared memory bằng shm\_open với quyền **O\_CREAT**, cấp quyền đọc ghi bằng **O\_RDWR** và truyền mã quyền là **0666**.
* Cấp phát không gian cho shared memory bằng **ftruncate** với kích thước được cấp phát là độ lớn của cấu trúc Buffer.
* Ánh xạ vùng nhớ buffer vào con trỏ ptr bằng **mmap**, với vị trí ánh xạ đầu là **0** và độ lớn ánh xạ bằng với độ lớn của cấu trúc Buffer. Cấp quyền đọc ghi bằng **PROT\_READ** và **PROT\_WRITE**, cấp quyền chia sẽ thay đổi bằng **MAP\_SHARED**.
* Khởi tạo trạng thái ban đầu cho buffer với **isFinish = false** (buffer đang được sử dụng) và **length = 0** (buffer chưa có bất kỳ phần tử nào).
* Khởi tạo seed cho random bằng **srand**, seed sẽ là thời gian hiện tại.

A screen shot of a computer program

Description automatically generated

*Hàm main: Tạo tiến trình con và chia Producer – Consumer*

* Lệnh **fork** để tạo tiến trình còn.
* Với **pid == 0**, đây là ngữ cảnh của tiến trình con. Thực hiện function xử lí của **Consumer**.
* Với **pid > 0**, đây là ngữ cảnh của tiến trình cha. Thực hiện function xử lí của **Producer**.
* Trường hợp **pid < 0**, fork thất bại, thông báo lỗi và **return 1** (lỗi).

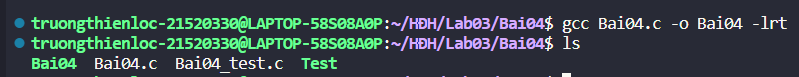
A computer code with text

Description automatically generated with medium confidence

*Hàm main: Đóng kết nối với shared memory và thu hồi bộ nhớ*

* Đóng kết nối với shared memory bằng **munmap**.
* Thu hồi bộ nhớ bằng **close**.

1. **Kết quả:**



*Build file thực thi*

* Build file **Bai04.c** (file chứa code) thành file **Bai04** (file thực thi).
* Khi build file, phải gán thêm cờ **-lrt**, nếu không gán thì gcc sẽ không cho phép build.
* Như trên hình, ta thấy file **Bai04** đã được build và khi dùng lệnh **ls**, nó xuất hiện có chứa **màu xanh** (nghĩa là có quyền thực thi).

**A screenshot of a computer program

Description automatically generated**

*Chạy file thực thi*

* Khi thực thi file Bai04, cứ sau **1 giây**, màn hình sẽ in ra giá trị của buffer và tổng của các số trong buffer.
* Lần đầu tiên, buffer có một số duy nhất là **18** và có tổng là **18**. Sau **1 giây**, giá trị bên trong buffer có thêm số **11** và tổng là **29**… Đến khi tổng các số bên trong buffer **113** (>100) thì **Consumer thông báo đóng**, kế tiếp là **Producer thông báo đóng**.

**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...