|  |
| --- |
| Họ và tên: Nguyễn Văn Hưng  Mã số sinh viên: 23520569  Lớp: ATTN2023 |

HỆ ĐIỀU HÀNH  
BÁO CÁO LAB 3

**CHECKLIST (Đánh dấu x khi hoàn thành)**

**Lưu ý mỗi câu phải làm đủ 3 yêu cầu**

1. **CLASSWORK**

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

1. **HOMEWORK**

|  |  |
| --- | --- |
|  | **3.6** |
| **Trình bày cách làm** |  |
| **Chụp hình minh chứng** |  |
| **Giải thích kết quả** |  |

**Tự chấm điểm:**  9

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

1. **CLASSWORK**

* Ví dụ 3-1:
* Code: [test\_fork.c](https://github.com/miscitaofvh/HDH-Lab3/blob/main/test_fork.c)
* Kết quả:

A screen shot of a computer

Description automatically generated

* Giải thích code:

+ Hàm fork() thực hiện tạo ra tiến trình mới là tiến trình con.

+ pid > 0: Đây là tiến trình cha, pid chứa giá trị Process ID của tiến trình con

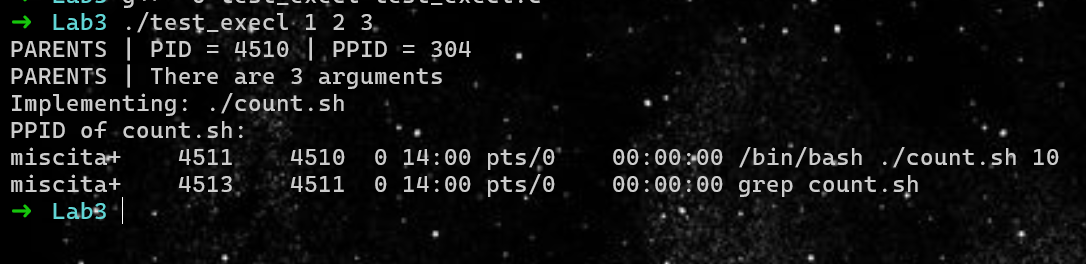
Thực hiện in ra Process ID và Parent Process ID của tiến trình cha

In ra số lượng tham số được truyền vào.

+ pid == 0: Đây là tiến trình con, pid bằng 0 trong tiến trình con.

Thực hiện in ra Process ID và Parent ID của tiến trình con

In ra giá trị của các tham số được truyền vào.

* Ví dụ 3-2:
* Code: [test\_execl.c](https://github.com/miscitaofvh/HDH-Lab3/blob/main/test_execl.c)
* Kết quả: 
* Giải thích code:

+ fork() tạo ra tiến trình con.

+ pid > 0: Tiến trình cha in ra Process ID và Parent Process ID

In ra số lượng tham số được truyền vào  
 wait(NULL) đợi đến khi tiến trình con hoàn thành rồi tiếp tục chạy.

+ pid == 0: Gọi hàm execl thay thế chương trình hiện tại bằng count.sh

count.sh thực hiện ps –ef để in ra các tiến tình đang chạy và lọc bằng grep count.sh

In từ 1 đến ($1 = 10) vào file count.txt

* Ví dụ 3-3:
* Code: [test\_system.c](https://github.com/miscitaofvh/HDH-Lab3/blob/main/test_system.c)
* Kết quả:

A screenshot of a computer

Description automatically generated

* Giải thích:

+ fork() tạo ra các tiến trình như các ví dụ trên.

+ Tiến trình cha in ra PID, PPID và số lượng tham số truyền vào.

+ system(“./count.sh 10”) tạo ra 1 tiến trình mới nên khi tiến trình mời hoàn thành xong thì chương trình con vẫn tiếp tục chạy.

* Ví dụ 3-4:
* Code: [test\_fork\_wait.c](https://github.com/miscitaofvh/HDH-Lab3/blob/main/test_fork_wait.c)
* Kết quả: A computer screen with white text

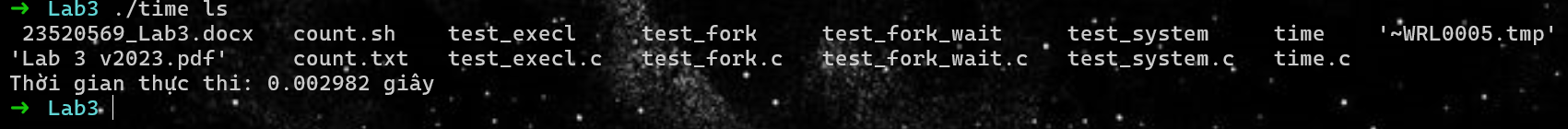
  Description automatically generated
* Giải thích:

+ fork() tạo ra tiến trình con

+ pid = 0, tiến trình con in ra pid của nó ( luôn bằng 0)

+ pid > 0, tiến trình cha thực hiện lệnh wait(NULL) đợi khi tiến trình con hoàn thành mới chạy tiếp, sau đó in ra pid của tiến trình cha.

* Code: [time.c](https://github.com/miscitaofvh/HDH-Lab3/blob/main/time.c)
* Kết quả:



* Giải thích:

+ gettimeofday(&start, NULL) tính thời gian bắt đầu

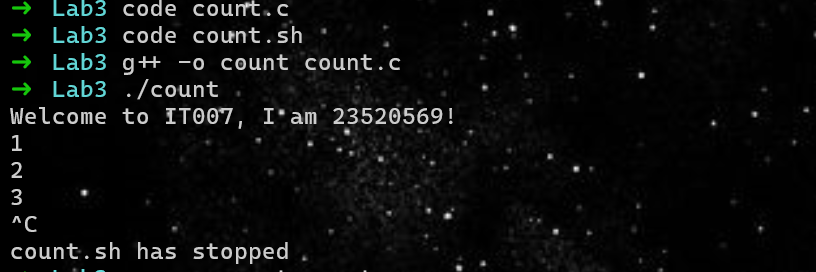
+ fork() tạo ra tiến trình con.

+ Chương trình con thực hiện <command> được nhập bằng execlp

+ Chương trình cha gọi wait(NULL) đợi đến khi tiến trình con hoàn thành.

+ gettimeofday(&end, NULL); tính thời gian kết thúc

+ Sau đó lấy thời gian kết thúc trừ thời gian bắt đầu là ra được kết quả.

* Code: [count.c](https://github.com/miscitaofvh/HDH-Lab3/blob/main/count.c)
* Kết quả: 
* Giải thích:

+ fork() tạo ra chương trình con.

+ Chương trình con thực execl("./count.sh", "./count.sh", "120", NULL);

+ Khi nhấn Ctrl+C SIGINT được gửi đến tiến trình cha và tiến trình con, tiến trình con bị dừng ngay lập tức nhưng tiến trình cha vẫn chạy.

+ signal(SIGINT, handle\_sigint) tiến trình cha khi nhận được SIGINT sẽ gọi hàm handle\_sigint.

+ handle\_sigint thực hiện kill(child\_pid, SIGKILL) sẽ gửi tín hiệu SIGKILL đến count.sh và buộc nó phải dừng

+ Sau đó in ra count.sh has stopped

* Code: [product-consumer](https://github.com/miscitaofvh/HDH-Lab3/blob/main/prod_cons.c)
* Kết quả: A screenshot of a computer screen

  Description automatically generated
* Giải thích:

+ Tạo ra shared\_memory 10 bytes để chia sẻ dữ liệu giữa tiến trình cha và tiến trình con.

+ fork() tạo ra tiến trình con.

+ Chương trình cha thực hiện random một số ngẫu nhiên trong khoảng từ 10 đến 20, đợi đến khi \*shared\_memory bằng 0 hoặc -1 thì tiếp tục chạy.

+ \*shared\_memory bằng 0 thì tiến trình con đã nhận dữ liệu và tính tổng.

+ \*shared\_memory bằng -1 thì tiến trình con đã kết thúc.

+ Tiến trình con thực hiện đợi cho đến khi \*shared\_memory khác 0 có nghĩa là tiến trình cha để cho dữ liệu vào.

+ Sau đó, tiến trình con thực hiện tính tổng

+ Nếu tổng lớn hơn 100 thì gán \*shared\_memory bằng -1 và break để kết thúc chương trình. Ngược lại, gán \*shared\_memory bằng 0 thể hiện đã xử lí dữ liệu.

….

1. **HOMEWORK**

* Code: [collatz.c](https://github.com/miscitaofvh/HDH-Lab3/blob/main/collatz.c)
* Kết quả: A screenshot of a computer

  Description automatically generated
* Giải thích:

+ Kiểm tra nếu argc < 2 thì yêu cầu nhập tham số.  
+ Kiếm tra nếu input < 1 thì yêu cầu tham số số lớn hơn 0

+ Tạo ra shared\_memory và fork() để tạo tiến trình con.

+ Tiến trình cha đợi \*(shared\_memory + 1) bằng 1 mới xử lí, tiến trình con thì bằng 0

+ Tiến trình cha nhận giá trị hiện tại \*(shared\_memory), in ra màn hình sau đó kiểu tra giá trị hiện tại bằng 1 thì không xử lí nữa.

+ Tiến trình cha đợi tiến trình con kết thúc mới dừng chương trình.

+ Tiến tình con đợi \*(shared\_memory + 1) bằng 0 mới xử lí.  
+ Tiến trình con nhận giá trị tại \*(shared\_memory) tính toán giá trị tiếp theo và thay đổi \*(shared\_memory + 1) bằng 1.

+ Nếu trước khi xử lí bằng 1 hoặc bằng 2 thì dừng chương trình.