BỘ GIÁO DỤC VÀ ĐÀO TẠO
ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

HỆ ĐIỀU HÀNH – THỰC HÀNH

BÁO CÁO LAB03

Lớp: IT007.N12.KHCL

Tên: Lê Gia Kiệt

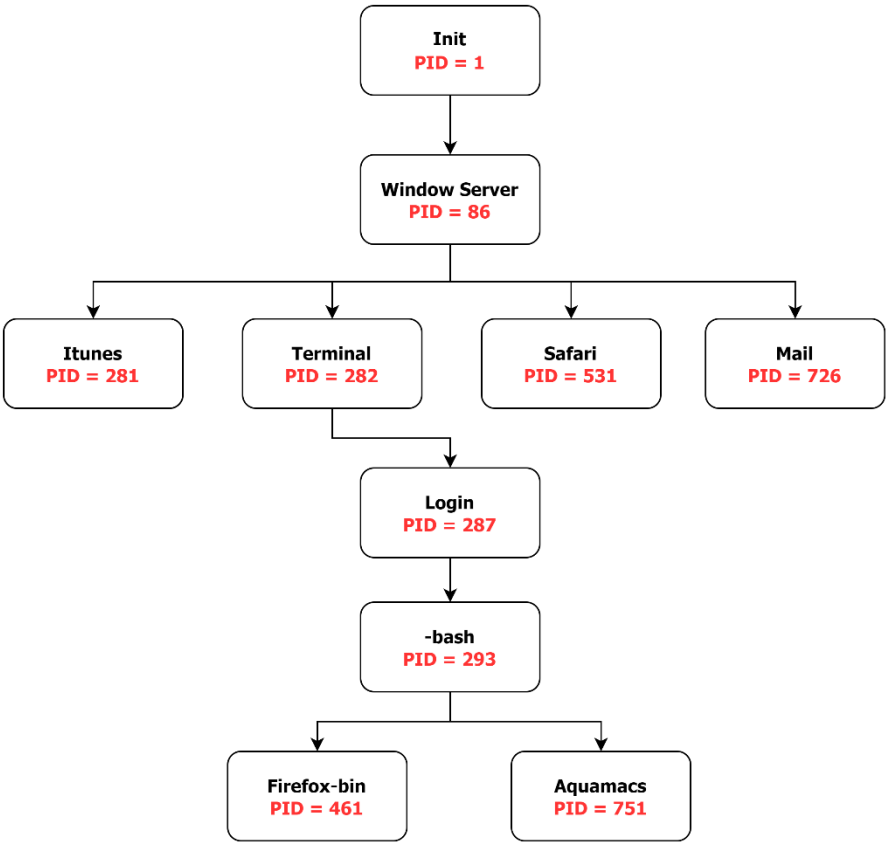
MSSV: 21522255

BÀI LÀM

1. Mối quan hệ cha con giữa các tiến trình

a. Vẽ cây quan hệ parent-child của các tiến trình bên dưới:

UID	PID	PPID	COMMAND
88	86	1	WindowServer
501	281	86	iTunes
501	282	86	Terminal
0	287	282	login
501	461	293	firefox-bin
501	531	86	Safari
501	726	86	Mail
501	751	293	Aquamacs
501	293	287	-bash



b. Trình bày cách sử dụng lệnh ps để tìm tiến trình cha của một một tiến trình dựa vào PID của nó.

- **Lệnh "ps" dùng để liệt kê chi tiết các tiến trình.**

```
giakiet@giakiet-virtual-machine:~$ ps
    PID TTY          TIME CMD
   3445 pts/0    00:00:00 bash
  10418 pts/0    00:00:00 ps
```

- **Lệnh " *ps -f* " hoặc "ps -ef "dùng để liệt kê đầy đủ và chi tiết hơn lệnh "*ps* ".**

```
giakiet@giakiet-virtual-machine:~$ ps -f
UID          PID    PPID  C STIME TTY          TIME CMD
giakiet     3445    3427  0 23:21 pts/0    00:00:00 bash
giakiet    10731    3445  0 23:28 pts/0    00:00:00 ps -f
```

```
giakiet@giakiet-virtual-machine:~$ ps -ef
UID          PID    PPID  C STIME TTY          TIME CMD
root           1       0  0 23:19 ?        00:00:03 /sbin/init auto noprompt sp
root           2       0  0 23:19 ?        00:00:00 [kthreadd]
root           3       2  0 23:19 ?        00:00:00 [rcu_gp]
root           4       2  0 23:19 ?        00:00:00 [rcu_par_gp]
root           5       2  0 23:19 ?        00:00:00 [netns]
root           7       2  0 23:19 ?        00:00:00 [kworker/0:0H-events_highpr
root           9       2  0 23:19 ?        00:00:00 [kworker/0:1H-events_highpr
root          10       2  0 23:19 ?        00:00:00 [mm_percpu_wq]
root          11       2  0 23:19 ?        00:00:00 [rcu_tasks_rude_]
root          12       2  0 23:19 ?        00:00:00 [rcu_tasks_trace]
root          13       2  0 23:19 ?        00:00:00 [ksoftirqd/0]
root          14       2  0 23:19 ?        00:00:00 [rcu_sched]
root          15       2  0 23:19 ?        00:00:00 [migration/0]
root          16       2  0 23:19 ?        00:00:00 [idle_inject/0]
root          18       2  0 23:19 ?        00:00:00 [cpuhp/0]
root          19       2  0 23:19 ?        00:00:00 [cpuhp/1]
root          20       2  0 23:19 ?        00:00:00 [idle_inject/1]
root          21       2  0 23:19 ?        00:00:00 [migration/1]
root          22       2  0 23:19 ?        00:00:00 [ksoftirqd/1]
root          24       2  0 23:19 ?        00:00:00 [kworker/1:0H-events_highpr
root          25       2  0 23:19 ?        00:00:00 [kdevtmpfs]
root          26       2  0 23:19 ?        00:00:00 [inet_frag_wq]
root          27       2  0 23:19 ?        00:00:00 [kauditd]
root          29       2  0 23:19 ?        00:00:00 [khungtaskd]
root          30       2  0 23:19 ?        00:00:00 [oom_reaper]
root          31       2  0 23:19 ?        00:00:00 [writeback]
root          32       2  0 23:19 ?        00:00:00 [kcompactd0]
root          33       2  0 23:19 ?        00:00:00 [ksmd]
root          34       2  0 23:19 ?        00:00:00 [khugepaged]
root          39       2  0 23:19 ?        00:00:00 [kworker/1:1-rcu_gp]
root          81       2  0 23:19 ?        00:00:00 [kintegrityd]
```

- **Các thông số được thể hiện**

Cột	Mô tả
UID	ID người sử dụng, mà tiến trình buộc phải sở hữu.
PID	ID của tiến trình.
PPID	ID của tiến trình cha.
C	CPU sử dụng tiến trình.
STIME	Thời gian bắt đầu tiến trình.
TTY	Kiểu terminal liên kết với tiến trình.
CMD	Lệnh bắt đầu tiến trình này.

- **Các tham số của lệnh "ps".**

Tùy chọn	Mô tả
-a	Chỉ thông tin về tất cả người dùng.
-x	Chỉ thông tin các tiến trình mà không có termial.
-u	Chỉ thông tin thêm vào chức năng -f
-e	Hiển thị thông tin mở rộng

- **Cách tìm tiến trình cha**

  Sử dụng lệnh "*ps -f <PID của tiến trình cần tìm cha>*". Giá trị trả về là thông tin chi tiết của tiến trình cần tìm, với cột PPID chứa PID của tiến trình cha của tiến trình cần tìm.

```
giakiet@giakiet-virtual-machine:~$ ps -f 3445
UID          PID    PPID  C STIME TTY      STAT   TIME CMD
giakiet     3445    3427  0 23:21 pts/0    Ss     0:00 bash
```

c. Tìm hiểu và cài đặt lệnh pstree (nếu chưa được cài đặt), sau đó trình bày cách sử dụng lệnh này để tìm tiến trình cha của một tiến trình dựa vào PID của nó.

- Lệnh "pstree" liệt kê tiến trình theo dạng cây.

  - Tiến trình cha là tiến tình ở bên trái của một tiến trình

  - Ngược lại, tiến trình con là tiến trình nằm bên phải của một tiến trình

Ví dụ:

*systemd là **tiến trình cha** của ModemManager.*

2[ModemManager}] là **tiến trình con** của ModemManager.*

```
giakiet@giakiet-virtual-machine:~$ pstree
systemd─┬─ModemManager───2*[{ModemManager}]
        ├─NetworkManager───2*[{NetworkManager}]
        ├─VGAuthService
        ├─accounts-daemon───2*[{accounts-daemon}]
        ├─acpid
        ├─avahi-daemon───avahi-daemon
        ├─colord───2*[{colord}]
        ├─cron
        ├─cups-browsed───2*[{cups-browsed}]
        ├─cupsd
        ├─dbus-daemon
        ├─fprintd───4*[{fprintd}]
        ├─fwupd───4*[{fwupd}]
        ├─gdm3─┬─gdm-session-wor─┬─gdm-wayland-ses─┬─gnome-session-b───2*[{gnome-session-b}]
        │      │                 │                 └─2*[{gdm-wayland-ses}]
        │      │                 └─3*[{gdm-session-wor}]
        │      └─2*[{gdm3}]
```

- Để tìm tiến trình cha của một tiến trình dựa vào PID của nó, ta sử dụng lệnh "***pstree -s -p <PID của tiến trình cần tìm cha>",*** màn hình sẽ trả về một cây quan hệ và dựa trên quy tắc nêu trên có thể

```
giakiet@giakiet-virtual-machine:~$ ps -f
UID          PID    PPID  C STIME TTY          TIME CMD
giakiet     3445    3427  0 23:21 pts/0    00:00:00 bash
giakiet    11189    3445  0 23:59 pts/0    00:00:00 ps -f
giakiet@giakiet-virtual-machine:~$ pstree -s -p 3445
systemd(1)───systemd(1636)───gnome-terminal-(3427)───bash(3445)───pstree(11190)
giakiet@giakiet-virtual-machine:~$ pstree -s -p 3427
systemd(1)───systemd(1636)───gnome-terminal-(3427)─┬─bash(3445)───pstree(11268)
                                                   ├─{gnome-terminal-}(3428)
                                                   ├─{gnome-terminal-}(3430)
                                                   └─{gnome-terminal-}(3431)

giakiet@giakiet-virtual-machine:~$
```

nhận thấy đâu là tiến trình cha và tiến trình con. *(sau mỗi tiến trình sẽ có thông số PID của tiến trình đó)*

2. Chương trình bên dưới in ra kết quả gì? Giải thích tại sao?

Sau khi biên dịch chương trình, terminal báo lỗi như hình, qua tìm hiểu thấy đoạn code khai báo thiếu thư viện.

```
giakiet@giakiet-virtual-machine:~$ gcc exercise2.c -o exercise2
exercise2.c: In function ‘main’:
exercise2.c:9:2: error: unknown type name ‘pid_t’
    9 |  pid_t pid;
      |  ^~~~~
exercise2.c:11:8: warning: implicit declaration of function ‘fork’ [-Wimplicit-
function-declaration]
   11 |  pid = fork();
      |        ^~~~
exercise2.c:14:2: warning: implicit declaration of function ‘exit’ [-Wimplicit-
function-declaration]
   14 |  exit(0);
      |  ^~~~
exercise2.c:8:1: note: include ‘<stdlib.h>’ or provide a declaration of ‘exit’
    7 | #include<stdio.h>
  +++ |+#include <stdlib.h>
    8 | int main(){
exercise2.c:14:2: warning: incompatible implicit declaration of built-in functi
on ‘exit’ [-Wbuiltin-declaration-mismatch]
   14 |  exit(0);
      |  ^~~~
exercise2.c:14:2: note: include ‘<stdlib.h>’ or provide a declaration of ‘exit’
exercise2.c:16:2: warning: implicit declaration of function ‘wait’ [-Wimplicit-
```

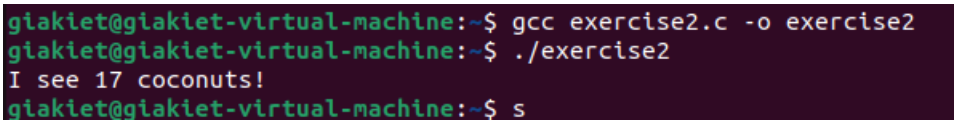
Chủ động khai báo thêm các thư viện vào file .c

```
giakiet@giakiet-virtual-machine:~$ cat exercise2.c
/*#######################################
# University of Information Technology #
# IT007 Operating System #
# <Your name>, <your Student ID> #
# File: exercise_2.c #
#######################################*/
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/wait.h>

int main(){
        pid_t pid;
        int num_coconuts = 17;
        pid = fork();
        if(pid == 0) {
                num_coconuts = 42;
                exit(0);
        } else {
                wait(NULL); /*wait until the child terminates */
        }
        printf("I see %d coconuts!\n", num_coconuts);
        exit(0);
}
```

Chương trình biên dịch thành công và đã chạy thành công.

Và kết quả sau khi chạy file là **"I see 17 coconuts!".**



Giải thích: Đầu tiên, đặt biến pid với kiểu dữ liệu pid_t, sau đó gán biến num_coconuts = 17 kiểu số nguyên và tạo tiến trình với lệnh pid = fork();

Sau khi gặp hàm fork(), chương trình tạo ra tiến trình con với giá trị pid của tiến trình cha là pid của tiến trình con (pid > 0). Với tiến trình con giá trị pid = 0.

Tiến trình sẽ được thực thi trước, trong khi đó tiến trình cha sẽ chờ tiến trình con kết thúc rồi mới thực thi.

- Trong tiến trình con, với pid == 0, thì num_coconuts = 42 và kết thúc tiến trình con và không thực hiện gì.

- Sau khi tiến trình con kết thúc, tiến trình cha với pid > 0, sẽ thực hiện khối lệnh trong else{}, vì tiến trình con đã kết thúc với lệnh wait(NULL); ở đây coi như đã thực hiện, sau đó thực hiện printf("I see %d coconuts!\n", num_coconuts) và exit(0);

Vấn đề có thể làm băn khoăn, num_coconuts trong tiến trình con đã = 42 tại sao nhưng về lại tiến trình cha thì num_coconuts là 17, vì khi tạo tiến trình con sẽ sử dụng vùng nhớ khác với tiến trình cha, vì vậy không thể thay đổi giá trị trong tiến trình cha.

3. Trong phần thực hành, các ví dụ chỉ sử dụng thuộc tính mặc định của pthread, hãy tìm hiểu POSIX thread và trình bày tất cả các hàm được sử dụng để làm thay đổi thuộc tính của pthread, sau đó viết các chương trình minh họa tác động của các thuộc tính này và chú thích đầy đủ cách sử dụng hàm này trong chương trình. (Gợi ý các hàm liên quan đến thuộc tính của pthread đều bắt đầu bởi: pthread_attr_*)

Trả lời:

Các hàm thuộc tính

Thuộc Tính	Giá trị mặc định	Ý nghĩa	Hàm
Guradsize	PAGEIZES	Kích thước đảm bảo cho tiểu trình không dùng quá không gian được cấp phát	int pthread_attr_setguardsize(pthread_attr_t *attr, size_t guardsize)
Scope	PTHREAD_SCOPE_PROCESS	Dùng tài nguyên trong phạm vi cho phép của tiến trình	int pthread_attr_setscope(pthread_attr_t *attr, int scope)
Detachstate	PTHREAD_CREATE_JOINABLE	Tiểu trình được hợp với các tiến trình khác	int pthread_attr_setdetachstate(pthread_attr_t *attr, int detachstate)
Stackaddr	NULL	Tiểu trình mới có địa chỉ trong system-allocated stack	int pthread_attr_setstack(pthread_attr_t *attr, void *stackaddr)
SatckSize	NULL	Tiểu trình tới sẽ có kich thước do system quy định	int pthread_attr_setstacksize(pthread_attr_t *attr, size_t stacksize)
Inheritsched	PTHREAD_INHERIT_SCHED	Tiểu trình con sẽ thừa kế lịch độ ưu tiên của tiểu trình cha	int pthread_attr_setinheritsched (pthread_attr_t *attr, int Inheritsched)
SchedPolicy	SCHED_OTHER	Tiểu trình sẽ chạy tuân theo độ ưu tiên của tiểu trình	int pthread_attr_setschedpolicy(pthread_attr_t *attr, int policy)

Code Minh hoạ:

Hello.sh exercise3.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
int main()
{
    size_t stksize;
    pthread_attr_t atr;
    pthread_attr_getstacksize(&atr, &stksize); // Lấy kích thước stack hiện tại, gán vào biến stksize
    printf("Kich thuoc stack cu: %ld\n", stksize);
    pthread_attr_setstacksize(&atr, 21522255); // Set kích thước stack hiện tại thành 21522255
    pthread_attr_getstacksize(&atr, &stksize); // Lấy kích thước stack hiện tại, gán vào biến stksize
    printf("Kich thuoc stack moi: %ld\n", stksize);
    return 0;
}

```

Kết quả:

```
giakiet@giakiet-virtual-machine:~/Desktop$ gcc exercise3.c -o exercise3
giakiet@giakiet-virtual-machine:~/Desktop$ ./exercise3
Kich thuoc stack cu: 140728384821209
Kich thuoc stack moi: 21522255
giakiet@giakiet-virtual-machine:~/Desktop$
```

4. Viết chương trình làm các công việc sau theo thứ tự:

a. In ra dòng chữ: "Welcome to IT007, I am <your_Student_ID>!"

b. Mở tệp abcd.txt bằng vim editor

c. Tắt vim editor khi người dùng nhấn CRTL+C

d. Khi người dùng nhấn CTRL+C thì in ra dòng chữ: "You are pressed CTRL+C! Goodbye!"

Code:

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <signal.h>
#include <sys/wait.h>

int close_vim = 1;

void on_sigint()
{
    system("sudo kill -9 `pidof vim`");
    printf("\nYou are pressed CTRL+C! Goodbye!");
    close_vim = 0;
}
int main()
{
    printf("Welcome to IT007, I am 21522255!\n");
    system("gnome-terminal -- vim abcd.txt");
    signal(SIGINT, on_sigint);
    while (close_vim){}

    return 0;
}
```

Kết quả chương trình:

```
giakiet@giakiet-virtual-machine:~/Desktop$ ./exercise4
Welcome to IT007, I am 20520605!
```


```
giakiet@giakiet-virtual-machine:~/Desktop$ ./exercise4
Welcome to IT007, I am 20520605!
^C
You are press CTRL+C! Goodbye
giakiet@giakiet-virtual-machine:~/Desktop$
```