

Name: Lê Minh Nguyệt

ID: 21521211

Class: IT007.N11

OPERATING SYSTEM

LAB 03'S REPORT

SUMMARY

Task		Status	Page
Section 3.5	Ex 1. Mối quan hệ cha – con giữa các tiến trình	Done	2
	Ex 2. Kiểm tra kết quả của chương trình cho sẵn	Done	5
	Ex 3. Các hàm được sử dụng để thay đổi thuộc tính của pthread	Done	8
	Ex 4. Viết chương trình mở - tắt vim editor khi nhận signal	Done	9

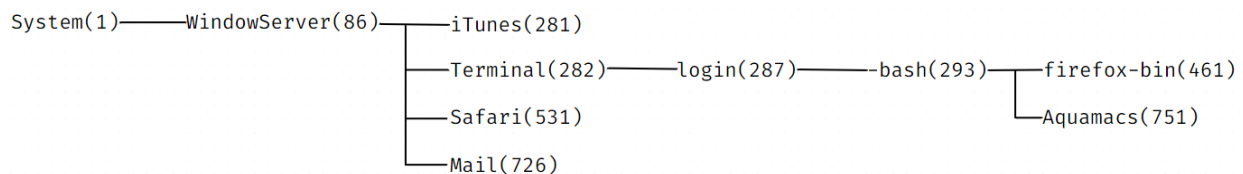
Self-scores: 9,0

Section 3.5

Ex 1. Môi quan hệ cha – con giữa các tiến trình:

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

- Quan sát trong bảng dữ liệu cho sẵn, ta biết được PID và PPID (PID của tiến trình cha) của mỗi tiến trình.
- Ví dụ tiến trình WindowServer có PID = 86, và là con của tiến trình có PID = 1. Tương tự như vậy, ta thấy iTunes (PID = 281), Terminal (PID = 282), Safari (PID = 531) và Mail (PID = 726) có cùng tiến trình cha là WindowServer (PID = 86). login (PID = 287) là con của Terminal, -bash (PID = 293) là con của login, firefox_bin (PID = 461) và Aquamacs (PID = 751) là con của -bash.



Hình 1. Cây quan hệ parent – child của các tiến trình

b. Sử dụng lệnh ps tìm tiến trình cha của một tiến trình dựa vào PID của nó:

- Ta sử dụng lệnh `ps -f`, với option `-f` để hiển thị thông tin đầy đủ (bao gồm PID của tiến trình cha) của các tiến trình tại thời điểm hiện tại.

```
nguyet-21521211@nguyet21521211-VirtualBox:~$ ps -f
UID          PID     PPID  C  STIME TTY          TIME CMD
nguyet-+    3061     3028  0   15:22 pts/0        00:00:00 bash
nguyet-+    3328     3061  0   15:51 pts/0        00:00:00 ps -f
nguyet-21521211@nguyet21521211-VirtualBox:~$
```

Hình 2. Sử dụng lệnh `ps -f` để hiển thị thông tin đầy đủ của các tiến trình tại thời điểm hiện tại

- Ta thấy thông tin đầy đủ của một tiến trình hiển thị ra có cột PPID, là PID tiến trình cha của tiến trình đó.
- Ví dụ tiến trình có PID = 3328 có cha là tiến trình mang PID = 3061.
- Để tìm tiến trình cha của một tiến trình cụ thể, ta dùng lệnh `ps -f <PID>`, trong đó, <PID> là PID của tiến trình muốn tìm cha.

```

nguyet-21521211@nguyet21521211-VirtualBox:~$ ps -f
UID          PID     PPID  C  STIME TTY          TIME CMD
nguyet-+    3061     3028  0  15:22 pts/0        00:00:00 bash
nguyet-+    3328     3061  0  15:51 pts/0        00:00:00 ps -f
nguyet-21521211@nguyet21521211-VirtualBox:~$ ps -f 3061
UID          PID     PPID  C  STIME TTY          STAT      TIME CMD
nguyet-+    3061     3028  0  15:22 pts/0        Ss         0:00 bash
nguyet-21521211@nguyet21521211-VirtualBox:~$

```

Hình 3. Sử dụng lệnh *ps -f 3061* để hiển thị thông tin đầy đủ của tiến trình có *PID = 3061*

- Ta thấy được các thông tin đầy đủ của tiến trình đó (gồm user ID, PID của nó, và PPID là PID của tiến trình cha).
 - Sử dụng lệnh *ps -f 3061*, ta tìm được tiến trình cha của nó là tiến trình có *PID = 3028*.
- c. Sử dụng lệnh *pstree* tìm tiến trình cha của một tiến trình dựa vào PID của nó:**
- Ta sử dụng lệnh *pstree -p* để hiển thị toàn bộ cây tiến trình trong hệ thống. Option *-p* để hiển thị PID kèm theo mỗi tiến trình tương ứng.

```

nguyet-21521211@nguyet21521211-VirtualBox: ~$ pstree -p
systemd(1)─ModemManager(1257)─{ModemManager}(1358)
          │                  {ModemManager}(1369)
          │
          └─NetworkManager(1184)─{NetworkManager}(1240)
                                   {NetworkManager}(1244)
          └─accounts-daemon(1175)─{accounts-daemon}(1258)
                                   {accounts-daemon}(1264)
          └─acpid(1176)
          └─avahi-daemon(1179)─avahi-daemon(1231)
          └─colord(1826)─{colord}(1827)
                        {colord}(1829)
          └─cron(1181)
          └─cups-browsed(1463)─{cups-browsed}(1481)
                              {cups-browsed}(1483)
          └─cupsd(1268)─dbus(1359)
                        │
                        └─dbus(1360)
          └─dbus-daemon(1183)
          └─gdm3(1354)─gdm-session-wor(2298)─gdm-wayland-ses(2348)─gnom+
                                                              {gdm+
                                                              {gdm+
                                                              {gdm-session-wor}(2299)
                                                              {gdm-session-wor}(2301)
                                                              {gdm3}(1361)
                                                              {gdm3}(1362)
          └─gnome-keyring-d(2329)─{gnome-keyring-d}(2330)
                                   {gnome-keyring-d}(2331)
                                   {gnome-keyring-d}(2449)
          └─kerneloops(1472)
          └─kerneloops(1482)
          └─networkd-dispat(1193)
          └─packagekitd(1659)─{packagekitd}(1660)
                              {packagekitd}(1661)
          └─polkitd(1195)─{polkitd}(1206)
                          {polkitd}(1215)
          └─power-profiles-(1197)─{power-profiles-}(1254)
                                   {power-profiles-}(1256)
          └─rsyslogd(1200)─{rsyslogd}(1232)
                           {rsyslogd}(1233)
                           {rsyslogd}(1234)
          └─rtkit-daemon(1494)─{rtkit-daemon}(1497)
                              {rtkit-daemon}(1498)
          └─snapd(1865)─{snapd}(1884)
                        {snapd}(1885)
                        {snapd}(1886)
                        {snapd}(1887)
                        {snapd}(1890)
                        {snapd}(1922)
                        {snapd}(1927)
                        {snapd}(1981)
                        {snapd}(1982)
          └─switcheroo-cont(1207)─{switcheroo-cont}(1276)
                                   {switcheroo-cont}(1280)
          └─systemd(2311)─(sd-pam)(2312)
                          └─at-spi2-registr(2706)─{at-spi2-registr}(2709)
                                                              {at-spi2-registr}(2711)
                          └─dbus-daemon(2333)
                          └─dconf-service(2668)─{dconf-service}(2669)
                                                              {dconf-service}(2671)
                          └─evolution-adre(2675)─{evolution-adre}(2676)
                                                              {evolution-adre}(2677)
                                                              {evolution-adre}(2679)
                                                              {evolution-adre}(2680)
                                                              {evolution-adre}(2682)
                          └─evolution-calen(2659)─{evolution-calen}(2660)
                                                              {evolution-calen}(2661)
                                                              {evolution-calen}(2663)
                                                              {evolution-calen}(2664)
                                                              {evolution-calen}(2667)
                                                              {evolution-calen}(2673)
                                                              {evolution-calen}(2674)
                                                              {evolution-calen}(2941)
                          └─evolution-sourc(2650)─{evolution-sourc}(2651)
                                                              {evolution-sourc}(2652)
                                                              {evolution-sourc}(2653)
                          └─gjs(2704)─{gjs}(2710)
                                          {gjs}(2712)
                                          {gjs}(2713)
                                          {gjs}(2714)
                          └─gjs(2936)─{gjs}(2947)
                                          {gjs}(2948)
                                          {gjs}(2949)
                                          {gjs}(2950)

```

Hình 4. Một phần của cây tiến trình trong hệ thống kèm theo PID, sử dụng lệnh `ps tree -p`

- Để tìm tiến trình cha của 1 tiến trình cụ thể, ta sử dụng lệnh *ps* *tree* -*ps* <PID>, với <PID> là PID của tiến trình cần tìm cha. Option -s để tìm tiến trình cha của 1 tiến trình cụ thể.

```
nguyet-21521211@nguyet21521211-VirtualBox:~$ pstree -ps 2299
systemd(1)---gdm3(1354)---gdm-session-wor(2298)---{gdm-session-wor}(2299)
nguyet-21521211@nguyet21521211-VirtualBox:~$
```

Hình 5. Sử dụng lệnh *ps* *tree* -*ps* 2299 để hiển thị cây tiến trình ứng với tiến trình con có PID = 2299

- Khi dùng lệnh *ps* *tree* -*ps* 2299, ta thấy cây tiến trình ứng với tiến trình PID = 2299, và biết được tiến trình cha của tiến trình này có PID = 2298.

Ex 2. Kiểm tra kết quả của chương trình cho sẵn:

- Sử dụng lệnh *vim* *ex2.c* để tạo/mở file *ex2.c* bằng *vim* editor, với nội dung như mẫu đề bài (dùng phím i để chuyển sang chế độ insert).

```
/*#####
#University of Information Technology#
#IT007 Operating System             #
#Le Minh Nguyet, 21521211           #
#File: ex2.c                        #
#####*/

#include <stdio.h>

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

~
~
~
~
:wq
```

Hình 6. Nội dung file *ex2.c* theo mẫu đề bài

- Dùng phím *esc* để thoát khỏi chế độ insert, gõ *:wq* để lưu lại nội dung của file và thoát ra khỏi *vim* editor.
- Quay lại màn hình shell, sử dụng lệnh *gcc* *ex2.c* -o *ex2* để tạo một file khả thực thi tên *ex2* từ file text *ex2.c*

```

nguyet-21521211@nguyet21521211-VirtualBox:~/LAB3$ vim ex2.c
nguyet-21521211@nguyet21521211-VirtualBox:~/LAB3$ gcc ex2.c -o ex2
ex2.c: In function 'main':
ex2.c:11:9: error: unknown type name 'pid_t'
    11 |         pid_t pid;
        |         ^~~~~~
ex2.c:13:13: warning: implicit declaration of function 'fork' [-Wimplicit-funct
ion-declaration]
    13 |         pid=fork();
        |         ^~~~~~
ex2.c:16:17: warning: implicit declaration of function 'exit' [-Wimplicit-funct
ion-declaration]
    16 |         exit(0);
        |         ^~~~~~
ex2.c:9:1: note: include '<stdlib.h>' or provide a declaration of 'exit'
     8 | #include <stdio.h>
    ++ | +#include <stdlib.h>
     9 |
ex2.c:16:17: warning: incompatible implicit declaration of built-in function 'e
xit' [-Wbuiltin-declaration-mismatch]
    16 |         exit(0);
        |         ^~~~~~
ex2.c:16:17: note: include '<stdlib.h>' or provide a declaration of 'exit'
ex2.c:19:17: warning: implicit declaration of function 'wait' [-Wimplicit-funct
ion-declaration]
    19 |         wait(NULL);
        |         ^~~~~~
ex2.c:22:9: warning: incompatible implicit declaration of built-in function 'ex
it' [-Wbuiltin-declaration-mismatch]
    22 |         exit(0);
        |         ^~~~~~
ex2.c:22:9: note: include '<stdlib.h>' or provide a declaration of 'exit'
nguyet-21521211@nguyet21521211-VirtualBox:~/LAB3$

```

Hình 7. Sử dụng lệnh `gcc ex2 -o ex2` để tạo file `ex2` khả thực thi từ file `ex2.c`

- Khi đó, ta thấy xuất hiện một số lỗi trong chương trình: không tìm thấy kiểu dữ liệu `pid_t`, không tìm được định nghĩa hàm `fork()`, hàm `exit()`, hàm `wait()`.
- Ta cũng thấy tại phần chú thích: include '`<stdlib.h>`' or provide a declaration of '`exit`'.
- Để sửa những lỗi này, ta thêm các thư viện chứa những hàm/kiểu dữ liệu trên vào chương trình (file `ex2.c`), bằng các lệnh:
`#include <stdlib.h>`
`#include <unistd.h>`
`#include <sys/wait.h>`
- Trong đó, hàm `exit()` được định nghĩa trong thư viện `stdlib.h`, hàm `fork()` và kiểu dữ liệu `pid_t` được cung cấp trong thư viện `unistd.h`, hàm `wait()` được định nghĩa trong thư viện `sys/wait.h`

```

/*#####
#University of Information Technology#
#IT007 Operating System             #
#Le Minh Nguyet, 21521211          #
#File: ex2.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);
    }
    printf("I see %d coconuts!\n", num_coconuts);
    exit(0);
}
~
~
:wq

```

Hình 8. Nội dung file *ex2.c* sau khi thêm các thư viện cần thiết

- Sau khi sửa lại nội dung file source code, ta cần thực hiện tạo lại file khả thực thi *ex2* từ nội dung của source code mới trong file *ex2.c* đã chỉnh sửa.
- Ta dùng lại lệnh `gcc ex2.c -o ex2`, và lệnh `./ex2` để thực thi file này.

```

nguyet-21521211@nguyet21521211-VirtualBox:~/LAB3$ vim ex2.c
nguyet-21521211@nguyet21521211-VirtualBox:~/LAB3$ gcc ex2.c -o ex2
nguyet-21521211@nguyet21521211-VirtualBox:~/LAB3$ ./ex2
I see 17 coconuts!
nguyet-21521211@nguyet21521211-VirtualBox:~/LAB3$

```

Hình 9. Sử dụng lệnh `gcc ex2 -o ex2` để tạo file *ex2* khả thực thi từ file *ex2.c*, và lệnh `./ex2` để thực thi file *ex2*

- Chương trình thực thi thành công, kết quả in ra màn hình “I see 17 coconuts!”, với `num_coconuts=17`.
- Quan sát source code, ta thấy: tại dòng lệnh `pid=fork()`; sinh ra 2 tiến trình con và tiến trình cha. Tiến trình cha khi đó sẽ có `pid>0` nên chạy đến dòng lệnh `wait(NULL)`; thực hiện đợi tiến trình con chạy xong rồi mới chạy tiếp. Tiến trình con có `pid=0` nên thực hiện gán `num_coconuts` của nó bằng 42 và kết thúc tiến trình với `exit(0)`; . Vì vậy, tiến trình con không in

gì ra màn hình. Sau khi tiến trình con kết thúc, tiến trình cha tiếp tục chạy và in ra màn hình chuỗi chứa num_coconuts của nó (num_coconuts vẫn bằng 17 trong bộ nhớ của tiến trình cha). Tiến trình cha kết thúc với *exit(0)*;

Ex 3. Các hàm được sử dụng để thay đổi thuộc tính của pthread:

STT	Hàm	Chức năng	Các tham số	Giá trị trả về	Chú thích
1	pthread_attr_init (pthread_attr_t *attr);	Khởi tạo thuộc tính với giá trị mặc định.	- attr trỏ đến thuộc tính cần khởi tạo.	Trả về 0 nếu thành công. Trả về một số ≠ 0 nếu thất bại.	
2	pthread_attr_destroy (pthread_attr_t *attr);	Thu hồi tài nguyên cấp cho thuộc tính (Xóa thuộc tính).	- attr trỏ đến thuộc tính cần thu hồi.		
3	pthread_attr_setguardsize (pthread_attr_t *attr, size_t guardsize);	Đặt guardsize của thuộc tính bằng giá trị biến guardsize.	- attr trỏ đến thuộc tính muốn thao tác, được khởi tạo bằng hàm pthread_attr_init() .		Guardsize cung cấp khả năng bảo vệ để tránh trường hợp bị tràn của stack pointer.
4	pthread_attr_getguardsize (const pthread_attr_t *restrict attr, size_t *restrict guardsize);	Lấy guardsize của thuộc tính và lưu vào guardsize.	- guardsize chứa giá trị guardsize.		
5	pthread_attr_setinheritsched (pthread_attr_t *attr, int inheritsched);	Đặt inherit scheduler của thuộc tính bằng giá trị biến inheritsched.	- attr trỏ đến thuộc tính muốn thao tác.		Inherit scheduler quyết định một tiến trình tạo ra với thuộc tính attr sẽ kế thừa scheduling từ đâu.
6	pthread_attr_getinheritsched (const pthread_attr_t *restrict attr, int *restrict inheritsched);	Lấy inherit scheduler của thuộc tính và lưu vào inheritsched.	- inheritsched chứa giá trị inherit scheduler.		
7	pthread_attr_setschedparam (pthread_attr_t *attr, const struct sched_param *param);	Tạo các thông số lập lịch trong attr, sử dụng giá trị từ param.	- attr trỏ đến thuộc tính muốn thao tác.		
8	pthread_attr_getschedparam (const pthread_attr_t *restrict attr, struct sched_param *restrict param);	Lấy thuộc tính ưu tiên lập lịch từ attr và lưu trữ nó vào param.	- param trỏ đến các giá trị muốn sử dụng để thao tác.		

9	pthread_attr_setstacksize (pthread_attr_t *attr, size_t stacksize);	Đặt lại kích thước stack của thuộc tính bằng giá trị trong <i>stacksize</i> .	- <i>attr</i> trỏ đến thuộc tính muốn thao tác. - <i>stacksize</i> chứa kích thước stack.		
10	pthread_attr_getstacksize (const pthread_attr_t *restrict attr, size_t *restrict stacksize);	Lấy kích thước stack của thuộc tính lưu vào <i>stacksize</i> .			

Ex 4. Viết chương trình mở - tắt vim editor khi nhận signal:

- Sử dụng lệnh *vim ex4.c* để tạo/mở file *ex4.c* bằng vim editor.
- Chuyển sang chế độ insert và tạo nội dung file như hình 11.

```

/*****
# University of Information Technology #
# IT007 Operating System              #
# Le Minh Nguyet, 21521211            #
# File: ex4.c                         #
*****/

#include <stdio.h>
#include <stdlib.h>
#include <signal.h>

int loop=2;
void handler(){
    if(loop==2){
        system("kill -9 `pidof vim`");
    }
    if(loop==1){
        printf("You pressed CTRL+C! Goodbye!\n");
    }
    loop--;
}

int main(){
    printf("Welcome to IT007, I am 21521211!\n");
    signal(SIGINT, handler);
    system("gnome-terminal -- vim abcd.txt");
    while(loop!=0);
    exit(0);
}
:wq

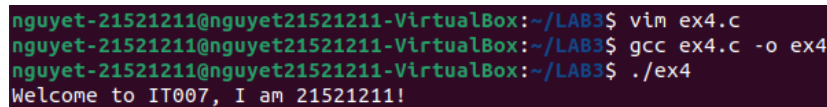
```

Hình 10. Nội dung file *ex4.c* chứa source code cho chương trình

- Trong hàm *main()*, thực hiện in ra màn hình dòng chữ theo yêu cầu câu a bằng lệnh *printf()*;
- Thực hiện bắt tín hiệu bằng lệnh *signal(SIGINT, handler)*; . Trong đó, *SIGINT* là tín hiệu bắt tổ hợp phím CTRL+C từ bàn phím, *handler* là thao tác sẽ được thực hiện khi bắt được tín hiệu. Như vậy, khi người dùng nhấn tổ hợp phím CTRL+C, hàm *handler()* sẽ được thực thi.
- Câu lệnh *system("gnome-terminal -- vim abcd.txt")*; để thực hiện mở một cửa sổ Terminal mới và chạy câu lệnh *vim abcd.txt* trên đó. Khi đó file *abcd.txt* trong vim editor sẽ được bật trên cửa sổ Terminal này. Ta thực hiện mở một cửa sổ khác chạy tiến trình mới, để tránh tiến trình mới chạy

- trên foreground của tiến trình hiện tại. Khi đó sẽ không thể bắt được tín hiệu (vì tín hiệu được cài trên tiến trình hiện tại).
- Ta đặt câu lệnh `gnome-terminal -- vim abcd.txt` trong lệnh `system()`, để tạo một tiến trình mới làm công việc thực thi lệnh này mà không thay thế tiến trình cha (vì ta đang ở trong một file text/file c, nên không thể trực tiếp chạy các lệnh shell).
 - Đặt một biến toàn cục `loop=2`. Khi ta nhấn CTRL+C, hàm `handler()` được thực thi:
 - + Nếu là lần thực thi đầu tiên (`loop` vẫn bằng 2), ta thực hiện tắt tiến trình đang chạy vim editor bằng câu lệnh `system("kill -9 `pidof vim`");`. Trong đó, lệnh `kill -9 `pidof vim`` thực hiện gửi tín hiệu SIGKILL(9) cho tiến trình được gọi tới (ở đây là vim). Khi nhận tín hiệu này, vim sẽ phải thoát ra ngay lập tức. Ta dùng lệnh `pidof vim` đặt trong cặp dấu `` để lấy PID của tiến trình vim, trả về cho lệnh `kill -9`.
 - + Nếu là lần thứ hai (`loop` đã giảm xuống bằng 1), ta thực hiện in ra màn hình dòng chữ theo yêu cầu câu d.
 - Chương trình vẫn chạy cho đến khi CTRL+C được nhấn 2 lần (`loop` giảm xuống bằng 0) thì kết thúc.

a. In ra màn hình dòng chữ “Welcome to IT007, I am <MSSV>!”



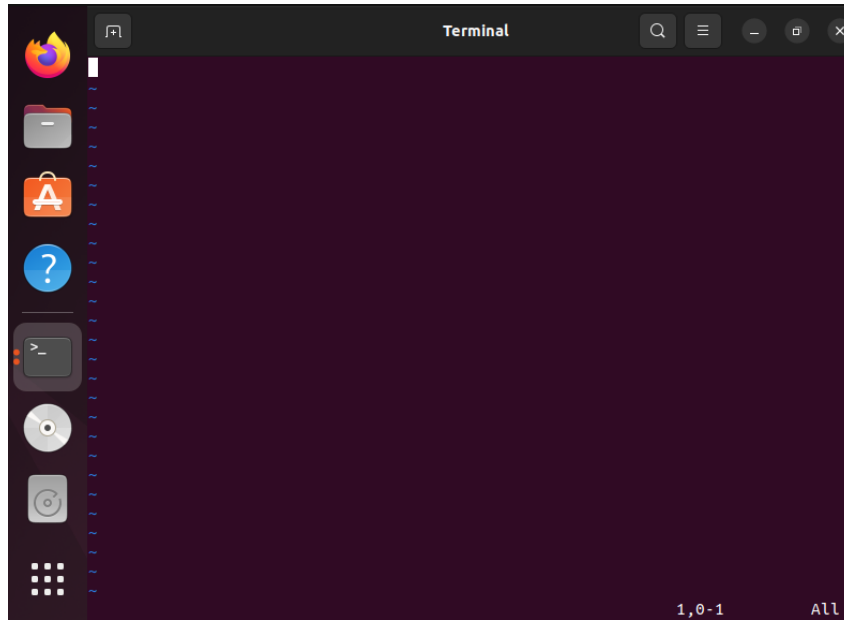
```

nguyet-21521211@nguyet21521211-VirtualBox:~/LAB3$ vim ex4.c
nguyet-21521211@nguyet21521211-VirtualBox:~/LAB3$ gcc ex4.c -o ex4
nguyet-21521211@nguyet21521211-VirtualBox:~/LAB3$ ./ex4
Welcome to IT007, I am 21521211!

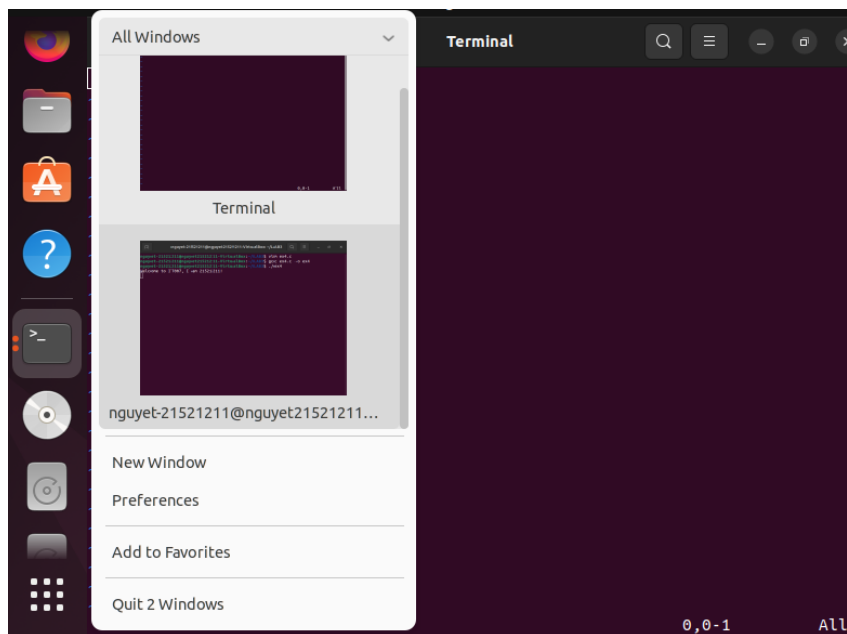
```

Hình 11. Sử dụng lệnh `gcc ex4 -o ex4` để tạo file `ex4` khả thực thi từ file `ex4.c`, và lệnh `./ex4` để thực thi file `ex4`

- Thực thi file `ex4`, ta thấy trên màn hình in ra dòng chữ theo yêu cầu của câu a.
- b. Mở file `abcd.txt` trong vim editor**
- Sau khi in ra màn hình, một cửa sổ Terminal mới tự động được bật lên, chạy vim editor và mở file `abcd.txt`



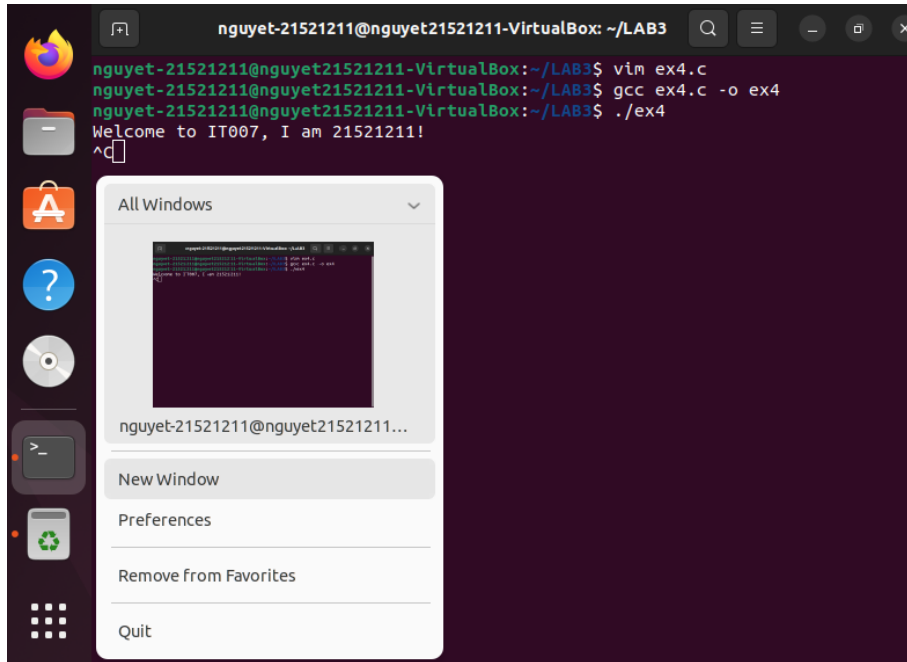
Hình 12. Màn hình Terminal mới được mở và chạy vim editor (file abcd.txt)



Hình 13. Hai màn hình Terminal chạy đồng thời

c. Tắt vim editor khi nhấn phím CTRL+C

- Ta thực hiện nhấn tổ hợp phím CTRL+C tại cửa sổ Terminal đang chạy file ex4. Khi đó, cửa sổ Terminal chạy vim bị tắt đi, chỉ còn một cửa sổ hiện tại.



Hình 14. Sau khi bấm phím CTRL+C, cửa sổ Terminal chạy vim editor đã bị tắt, chỉ còn màn hình hiện tại

d. In ra màn hình “You pressed CTRL+C! Goodbye!” khi nhấn CTRL+C

- Ta tiếp tục nhấn CTRL+C một lần nữa, trên màn hình in ra dòng chữ theo yêu cầu câu d và kết thúc.

```
nguyet-21521211@nguyet21521211-VirtualBox:~/LAB3$ vim ex4.c
nguyet-21521211@nguyet21521211-VirtualBox:~/LAB3$ gcc ex4.c -o ex4
nguyet-21521211@nguyet21521211-VirtualBox:~/LAB3$ ./ex4
Welcome to IT007, I am 21521211!
^C^CYou pressed CTRL+C! Goodbye!
nguyet-21521211@nguyet21521211-VirtualBox:~/LAB3$
```

Hình 15. Bấm phím CTRL+C một lần nữa, in ra màn hình dòng chữ và kết thúc chương trình