

Name: Vương Đình Thanh Ngân

ID: 20521649

Class: ATCL2022

OPERATING SYSTEM LAB 1'S REPORT

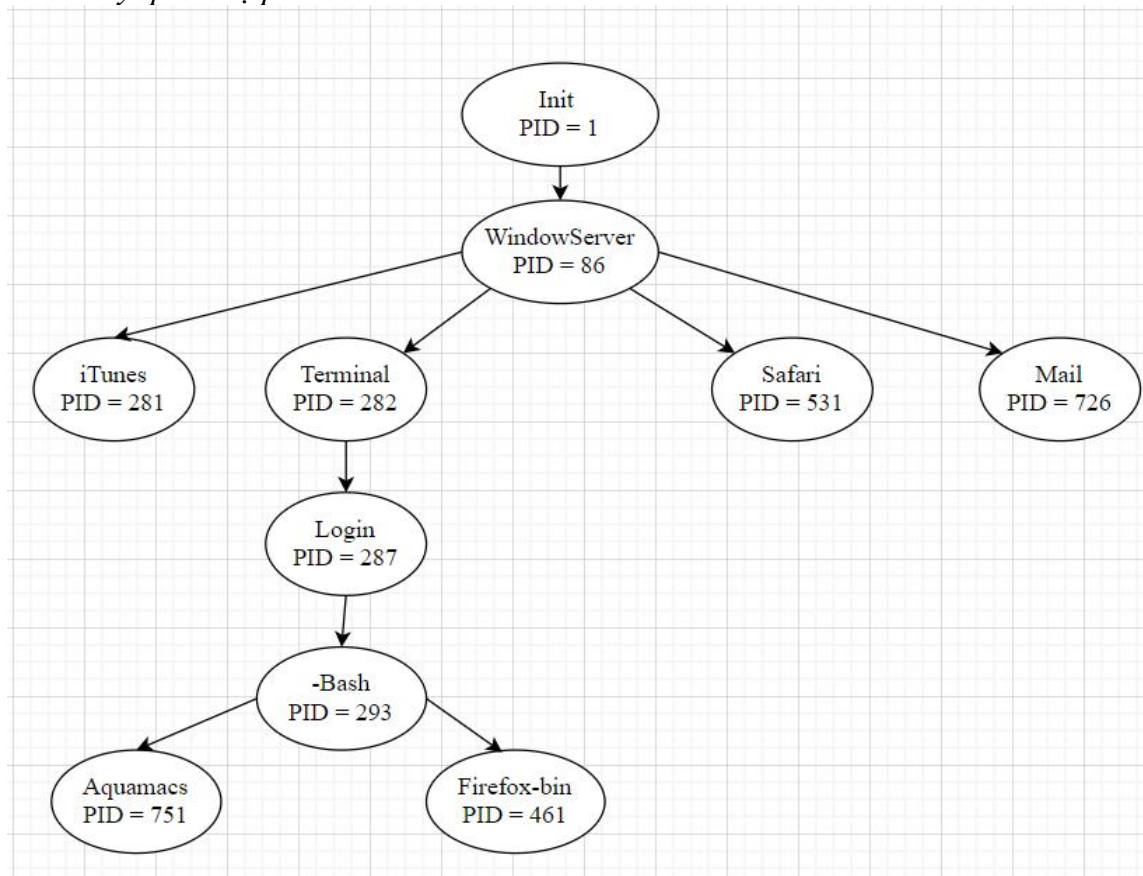
SUMMARY

Task	Status	Page
Section 3.5	Ex 1	Done
	Ex 2	Done
	Ex 3	Done
	Ex 4	Done

Self-scores:

Câu 1: **Mối quan hệ cha-con giữa các tiến trình**

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



Hình 1.1: Cây quan hệ parent-child của các tiến trình

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

- Lệnh `ps` dùng để liệt kê chi tiết các process tại thời điểm khởi chạy lệnh.
- Lệnh `ps -f` liệt kê chi tiết hơn lệnh `ps`

```
VuongDinhThanhNgan_20521649@ubuntu:~$ cd Desktop
VuongDinhThanhNgan_20521649@ubuntu:~/Desktop$ ps
  PID TTY          TIME CMD
 3972 pts/0        00:00:00 bash
 3981 pts/0        00:00:00 ps
VuongDinhThanhNgan_20521649@ubuntu:~/Desktop$ ps -f
UID          PID    PPID  C STIME TTY          TIME CMD
tina         3972    3961  0 18:31 pts/0        00:00:00 bash
tina         3993    3972  0 18:32 pts/0        00:00:00 ps -f
```

Hình 1.2: Kết quả khi thực thi lệnh `ps` và lệnh `ps -f`

- Mô tả các thông tin hiển thị bởi lệnh `ps -f`:
- + UID: ID người sử dụng mà tiến trình này thuộc sở hữu (người chạy nó).
- + PID: ID của tiến trình.
- + PPID: ID của tiến trình cha.
- + C: CPU sử dụng của tiến trình.
- + STIME: Thời gian bắt đầu tiến trình.
- + TTY: Kiểu terminal liên kết với tiến trình.
- + TIME: Thời gian CPU bị sử dụng bởi tiến trình.
- + CMD: Lệnh bắt đầu tiến trình này.
- Có thể tìm kiếm tiến trình cha bằng cách nhìn vào cột PID để xác định ID của tiến trình con, nhìn qua cột PPID để xác định ID của tiến trình cha của các tiến trình con tương ứng.

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.

```

VuongDinhThanhNgan_20521649@ubuntu:~/Desktop$ pstree
systemd--ModemManager--2*[{ModemManager}]
--NetworkManager--2*[{NetworkManager}]
--VGAuthService
--accounts-daemon--2*[{accounts-daemon}]
--acpid
--anacron
--avahi-daemon--avahi-daemon
--bluetoothd
--colord--2*[{colord}]
--containerd--12*[{containerd}]
--cron
--cups-browsed--2*[{cups-browsed}]
--cupsd
--dbus-daemon
--dockerd--9*[{dockerd}]
--fwupd--4*[{fwupd}]
--gdm3--gdm-session-wor--gdm-x-session--Xorg--{Xorg}
--gnome-session-b--ssh-agent
--2*[{gnome-+
--2*[{gdm-x-session}]
--2*[{gdm-session-wor}]
--2*[{gdm3}]
--gnome-keyring-d--3*[{gnome-keyring-d}]
--irqbalance--{irqbalance}
--2*[{kerneloops}]
--networkd-dispat
--packagekitd--2*[{packagekitd}]
--polkitd--2*[{polkitd}]
--rsyslogd--3*[{rsyslogd}]
--rtkit-daemon--2*[{rtkit-daemon}]
--snapd--10*[{snapd}]
--switcheroo-cont--2*[{switcheroo-cont}]
--systemd--(sd-pam)
--at-spi-bus-laun--dbus-daemon
--3*[{at-spi-bus-laun}]
--at-spi2-registr--2*[{at-spi2-registr}]
--dbus-daemon
--dconf-service--2*[{dconf-service}]
--evolution-addre--5*[{evolution-addre}]
--evolution-calen--11*[{evolution-calen}]
--evolution-sourc--3*[{evolution-sourc}]
--gjs--4*[{gjs}]

```

Hình 1.3: Kết quả khi sử dụng lệnh `pstree`

Lệnh `pstree` liệt kê tiến trình theo dạng cây thay vì theo dạng danh sách như lệnh `ps`

- Tiến trình cha là tiến trình nằm bên trái cùng nhánh tiến trình đó
- Tiến trình con là tiến trình nằm bên phải cùng nhánh tiến trình đó.

+ Ví dụ:

- `modemManager` là tiến trình cha của tiến trình con `2*[{modemManager}]` 4
- `gdm3` là tiến trình cha của các tiến trình con `gdm-session-wor` và `2*[{gdm3}]`
- Để hiển thị PID của mỗi tiến trình trong cây ta dùng lệnh `pstree -p`
- Sau mỗi tiến trình có một thông số đặt trong dấu () là PID

```
VuongDinhThanhNgan_20521649@ubuntu:~/Desktop$ pstree -p
systemd(1)─ModemManager(866)─{ModemManager}(907)
          │                 └─{ModemManager}(913)
          └─NetworkManager(770)─{NetworkManager}(865)
                                └─{NetworkManager}(868)
          └─VGAuthService(739)
          └─accounts-daemon(761)─{accounts-daemon}(799)
                                └─{accounts-daemon}(854)
          └─acpid(762)
          └─avahi-daemon(765)─avahi-daemon(800)
          └─bluetoothd(766)
          └─colord(1909)─{colord}(1911)
                       └─{colord}(1913)
          └─containerd(881)─{containerd}(1012)
                           └─{containerd}(1013)
                               └─{containerd}(1014)
                                   └─{containerd}(1016)
                                       └─{containerd}(1093)
                                           └─{containerd}(1149)
                                               └─{containerd}(1157)
                                                   └─{containerd}(1216)
                                                       └─{containerd}(1218)
                                                           └─{containerd}(1236)
                                                               └─{containerd}(1250)
                                                                   └─{containerd}(1251)
          └─cron(767)
          └─cups-browsed(860)─{cups-browsed}(898)
                              └─{cups-browsed}(900)
          └─cupsd(852)
          └─dbus-daemon(769)
          └─dockerd(1241)─{dockerd}(1260)
                           └─{dockerd}(1261)
                               └─{dockerd}(1262)
                                   └─{dockerd}(1264)
                                       └─{dockerd}(1280)
                                           └─{dockerd}(1281)
                                               └─{dockerd}(1318)
                                                   └─{dockerd}(1324)
                                                       └─{dockerd}(1326)
          └─fwupd(2690)─{fwupd}(2691)
                       └─{fwupd}(2692)
                           └─{fwupd}(2693)
                               └─{fwupd}(2694)
```

Hình 1.4: Kết quả khi sử dụng lệnh `pstree -p`

o Ví dụ: colord có PID là 1909, là tiến trình cha của các tiến trình {colord} có PID là 1911 và tiến trình {colord} có PID là 1913.

- Để tìm kiếm tiến trình cha, ta sử dụng lệnh `pstree -ps <PID>`

o Ví dụ: Tìm kiếm tiến trình cha của một tiến trình có PID = 1957.

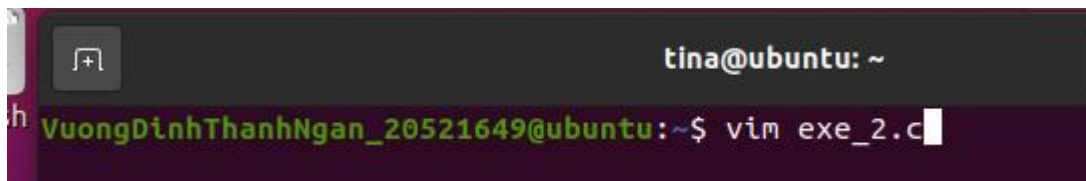

```
└─wpa_supplicant(788)
VuongDinhThanhNgan_20521649@ubuntu:~/Desktop$ pstree -ps 1957
systemd(1)──gdm3(914)──gdm-session-wor(1956)──{gdm-session-wor}(1957)
VuongDinhThanhNgan_20521649@ubuntu:~/Desktop$
```

Hình 1.5: Sử dụng lệnh `pstree -ps 1957` để tìm tiến trình cha

Sử dụng lệnh `pstree -ps 1957` => Tiến trình `gdm - session - wor (1957)` là trình cha cần tìm.

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

- Tạo `exe_2.c` bằng lệnh `vim exe_2.c`



```
tina@ubuntu: ~
VuongDinhThanhNgan_20521649@ubuntu:~$ vim exe_2.c
```

Hình 2.1: Tạo `exe_2.c` bằng lệnh `vim exercise_2.c`

- Sau khi tạo được `exe_2.c`, nhập vào chương trình như dưới đây

```
/*#####
# University of Information Technology #
# IT007 Operating System #
# <Vuong Dinh Thanh Ngan>, <20521649> #
#####*/
#include<stdio.h>
int main(){
    pid_t pid;
    int num_coconuts = 17;
    pid = fork();
    if(pid == 0) {
        num_coconuts = 42;
        exit(0);
    } else {
53    wait(NULL); /*wait until the child terminates */
    }
    printf("I see %d coconuts!\n", num_coconuts);
    exit(0);
}
~
```

Hình 2.2: Nhập đoạn chương trình của đề bài

- Sau khi nhập xong chương trình, tiến hành biên dịch chương trình bằng câu lệnh `gcc exe_2.c -o exe_2`. Thấy thông báo các lỗi như hình dưới:

```

VuongDinhThanhNgan_20521649@ubuntu:~$ gcc exe_2.c -o exe_2
exe_2.c: In function 'main':
exe_2.c:9:2: error: unknown type name 'pid_t'
   9 |     pid_t pid;
     |     ^~~~~~
exe_2.c:11:8: warning: implicit declaration of function 'fork' [-Wimplicit-function-declaration]
   11 |     pid = fork();
     |           ^~~~~
exe_2.c:14:2: warning: implicit declaration of function 'exit' [-Wimplicit-function-declaration]
   14 |     exit(0);
     |     ^~~~~
exe_2.c:14:2: warning: incompatible implicit declaration of built-in function 'exit'
exe_2.c:8:1: note: include '<stdlib.h>' or provide a declaration of 'exit'
   7 | #include<stdio.h>
+++ |+#include <stdlib.h>
   8 | int main(){
exe_2.c:16:3: error: expected ';' before 'wait'
   16 |     53
     |     ^
   17 |     ;
     |     ~~~~~
exe_2.c:20:2: warning: incompatible implicit declaration of built-in function 'exit'
   20 |     exit(0);
     |     ^~~~~
exe_2.c:20:2: note: include '<stdlib.h>' or provide a declaration of 'exit'
VuongDinhThanhNgan_20521649@ubuntu:~$

```

Hình 2.3: Các lỗi được hiển thị khi biên dịch chương trình

- Từ các lỗi trên, nhận thấy chương trình thiếu các thư viện chứa các hàm fork(), exit(), wait() và không định nghĩa được pid_t là gì.
- Cần thêm các thư viện: <stdlib.h>, <unistd.h>, <sys/wait.h> vào chương trình như hình dưới đây

```
1 /*#####
2 # University of Information Technology #
3 # IT007 Operating System #
4 # <Vuong Dinh Thanh Ngan>, <20521649> #
5 # File: exercise_2.c #
6 #####*/
7 #include<stdio.h>
8 #include<stdlib.h>
9 #include<unistd.h>
10 #include<sys/wait.h>
11 int main(){
12     pid_t pid;
13     int num_coconuts = 17;
14     pid = fork();
15     if(pid == 0) {
16         num_coconuts = 42;
17         exit(0);
18     }
19     else {
20         wait(NULL); /*wait until the child terminates */
21     }
22     printf("I see %d coconuts!\n", num_coconuts);
23     exit(0);
24 }
```

Hình 2.4: Thêm thư viện vào chương trình

- Lưu và thoát chương trình, biên dịch lại chương trình, nhận thấy chương trình được biên dịch thành công.

```
tina@ubuntu: ~
VuongDinhThanhNgan_20521649@ubuntu:~$ gcc exe_2.c -o exe_2
VuongDinhThanhNgan_20521649@ubuntu:~$ ./exe_2
I see 17 coconuts!
VuongDinhThanhNgan_20521649@ubuntu:~$
```

Hình 2.5: Biên dịch và thực thi chương trình

Thực thi chương trình bằng lệnh ./exe_2. Kết quả in ra màn hình “I see 17 coconuts!”

+ Giải thích đoạn chương trình:

- Bước 1: Đặt biến pid với kiểu dữ liệu pid_t
- Bước 2: Gán biến num_coconuts = 17 kiểu số nguyên
- Bước 3: Thực thi lệnh pid = fork(), tiến trình cha sẽ tạo ra một tiến trình con riêng biệt.

- Bước 4: Điều kiện `pid == 0` là của tiến trình con, nên ở tiến trình con biến `num_coconuts = 42`. Vì thực hiện lệnh `exit(0)` nên tiến trình con đã kết thúc và không in ra màn hình gì cả.
- Bước 5: Ở điều kiện còn lại (`pid > 0`) là của tiến trình cha với biến `num_coconuts = 17`, sau khi chờ tiến trình con thực hiện xong (xong hàm `wait(NULL)`) tiến trình cha thoát điều kiện và thực hiện lệnh `printf("I see %d coconuts! \n", num_coconuts)`, in ra màn hình `I see 17 coconuts!`

Câu 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_*`)

Hàm mô tả POSIX threa	Chức năng
<code>pthread_attr_destroy ()</code>	Giải phóng bộ nhớ cấp phát cho biến thuộc tính khi đối tượng đó không còn được yêu cầu nữa.
<code>pthread_attr_init ()</code>	Khởi tạo chuỗi đối tượng thuộc tính được trỏ đến bởi <code>attr</code> với các giá trị thuộc tính mặc định.
<code>pthread_attr_setstacksize()</code>	Xác định kích thước stack của các thuộc tính được gọi bởi <code>attr</code> để các giá trị quy định tại <code>STACKSIZE</code>
<code>pthread_attr_setstackaddr()</code>	Xác định địa chỉ stack của các đối tượng được gọi bởi <code>attr</code> để giá trị được chỉ định trong <code>stackaddr</code> .
<code>pthread_attr_getschedparam()</code>	Trả về tham số scheduling của tiến trình trong đối số <code>attr</code> . Nội dung của tham số được xác định trong <code><Schedule.h></code>
<code>pthread_attr_getdetachstate()</code>	Trả về thuộc tính trạng thái detach của đối tượng thuộc tính thread trong bộ đệm được trỏ tới bởi <code>detachstate</code>
<code>pthread_attr_getschedpolicy()</code>	Trả về thuộc scheduling policy của đối tượng <code>attr</code> trong bộ đệm được trỏ bởi <code>policy</code>
<code>pthread_attr_setscope()</code>	Đặt phạm vi tranh chấp thuộc tính của các thuộc tính chủ đề đối tượng được gọi bằng <code>attr</code> để giá trị được chỉ định trong <code>scope</code> .
<code>pthread_attr_getscope()</code>	Trả về các thuộc tính phạm vi tranh chấp của đối tượng thuộc tính thread được tham chiếu bởi <code>attr</code> trong bộ đệm được trỏ tới <code>scope</code>
<code>pthread_attr_getguardsize()</code>	Trả về thuộc tính Guardsize trong đối tượng <code>attr</code> . Thuộc tính

này sẽ được trả về trong tham số guardsize.

- Chương trình minh họa:



```
1 /* CELEBP10 */
2 #define _OPEN_THREADS
3 #include <stdio.h>
4 #include <pthread.h>
5
6 void *thread1(void *arg)
7 {
8     printf("hello from the thread\n");
9     pthread_exit(NULL);
10 }
11
12 int main()
13 {
14     int rc, stat;
15     pthread_attr_t attr;
16     pthread_t thid;
17
18     rc = pthread_attr_init(&attr);
19     if (rc == -1) {
20         perror("error in pthread_attr_init");
21         exit(1);
22     }
23
24     rc = pthread_create(&thid, &attr, thread1, NULL);
25     if (rc == -1) {
26         perror("error in pthread_create");
27         exit(2);
28     }
29
30     rc = pthread_join(thid, (void *)&stat);
31     exit(0);
32 }
```

Hình 3.1: Đoạn code minh họa `pthread_attr_init()`

```

VuongDinhThanhNgan_20521649@ubuntu:~$ gcc thread.c -o thread -pthread
thread.c: In function 'main':
thread.c:21:7: warning: implicit declaration of function 'exit' [-Wimplicit-func
tion-declaration]
    21 |         exit(1);
        |         ^~~~~
thread.c:21:7: warning: incompatible implicit declaration of built-in function '
exit'
thread.c:5:1: note: include '<stdlib.h>' or provide a declaration of 'exit'
    4 | #include <pthread.h>
    +++ |+#include <stdlib.h>
    5 |
thread.c:27:7: warning: incompatible implicit declaration of built-in function '
exit'
    27 |         exit(2);
        |         ^~~~~
thread.c:27:7: note: include '<stdlib.h>' or provide a declaration of 'exit'
thread.c:31:4: warning: incompatible implicit declaration of built-in function '
exit'
    31 |         exit(0);
        |         ^~~~~
thread.c:31:4: note: include '<stdlib.h>' or provide a declaration of 'exit'
VuongDinhThanhNgan_20521649@ubuntu:~$ ./thread
hello from the thread

```

Hình 3.2: Thực thi đoạn `pthread_attr_init()`

```

#define _OPEN_THREADS
#include <stdio.h>
#include <pthread.h>

void *thread1(void *arg) {
    pthread_exit(NULL);
}

int main()
{
    pthread_t      thid;
    pthread_attr_t attr;

    if (pthread_attr_init(&attr) == -1) {
        perror("error in pthread_attr_init");
        exit(1);
    }

    if (pthread_create(&thid, &attr, thread1, NULL) == -1) {
        perror("error in pthread_create");
        exit(2);
    }

    if (pthread_detach(&thid) == -1) {
        perror("error in pthread_detach");
        exit(4);
    }

    if (pthread_attr_destroy(&attr) == -1) {
        perror("error in pthread_attr_destroy");
        exit(5);
    }
    exit(0);
}

```

Hình 3.3: Đoạn code minh họa pthread_attr_destroy

```

VuongDinhThanhNgan_20521649@ubuntu:~$ ./thread_destroy
VuongDinhThanhNgan_20521649@ubuntu:~$

```

Hình 3.4: thực thi minh họa pthread_attr_destroy

```

1 #define _OPEN_THREADS
2 #include <stdio.h>
3 #include <pthread.h>
4
5 void *thread1(void *arg)
6 {
7     printf("hello from the thread\n");
8     pthread_exit(NULL);
9 }
10
11 int main()
12 {
13     int rc, stat;
14     size_t s1;
15     pthread_attr_t attr;
16     pthread_t thid;
17
18     rc = pthread_attr_init(&attr);
19     if (rc == -1) {
20         perror("error in pthread_attr_init");
21         exit(1);
22     }
23
24     s1 = 4096;
25     rc = pthread_attr_setstacksize(&attr, s1);
26     if (rc == -1) {
27         perror("error in pthread_attr_setstacksize");
28         exit(2);
29     }
30
31     rc = pthread_create(&thid, &attr, thread1, NULL);
32     if (rc == -1) {
33         perror("error in pthread_create");
34         exit(3);
35     }
36
37     rc = pthread_join(thid, (void *)&stat);
38     exit(0);
39 }

```

Hình 3.5: Đoạn chương trình minh họa `pthread_attr_setstacksize()`


```

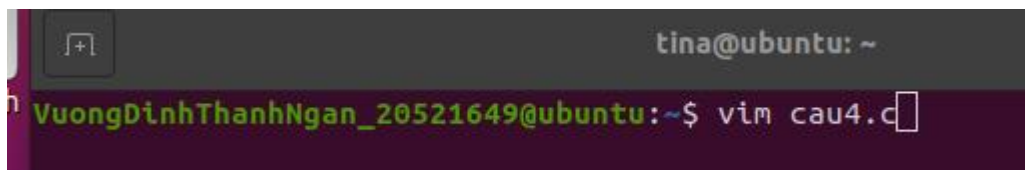
VuongDinhThanhNgan_20521649@ubuntu:~$ gcc setstacksize.c -o setstacksize -pthread
setstacksize.c: In function 'main':
setstacksize.c:21:7: warning: implicit declaration of function 'exit' [-Wimplicit-function-declaration]
   21 |         exit(1);
       |         ^~~~~
setstacksize.c:21:7: warning: incompatible implicit declaration of built-in function 'exit'
setstacksize.c:4:1: note: include '<stdlib.h>' or provide a declaration of 'exit'
   3 | #include <pthread.h>
+++ |+#include <stdlib.h>
   4 |
setstacksize.c:28:7: warning: incompatible implicit declaration of built-in function 'exit'
   28 |         exit(2);
       |         ^~~~~
setstacksize.c:28:7: note: include '<stdlib.h>' or provide a declaration of 'exit'
setstacksize.c:34:7: warning: incompatible implicit declaration of built-in function 'exit'
   34 |         exit(3);
       |         ^~~~~
setstacksize.c:34:7: note: include '<stdlib.h>' or provide a declaration of 'exit'
setstacksize.c:38:4: warning: incompatible implicit declaration of built-in function 'exit'
   38 |         exit(0);
       |         ^~~~~
setstacksize.c:38:4: note: include '<stdlib.h>' or provide a declaration of 'exit'
VuongDinhThanhNgan_20521649@ubuntu:~$ ./setstacksize
hello from the thread
VuongDinhThanhNgan_20521649@ubuntu:~$

```

Hình 3.6: thực thi chương trình minh họa pthread_attr_setstacksize()

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

- In ra dòng chữ: "Welcome to IT007, I am <your_Student_ID>!"
- Sử dụng lệnh vim cau4.c để tạo file



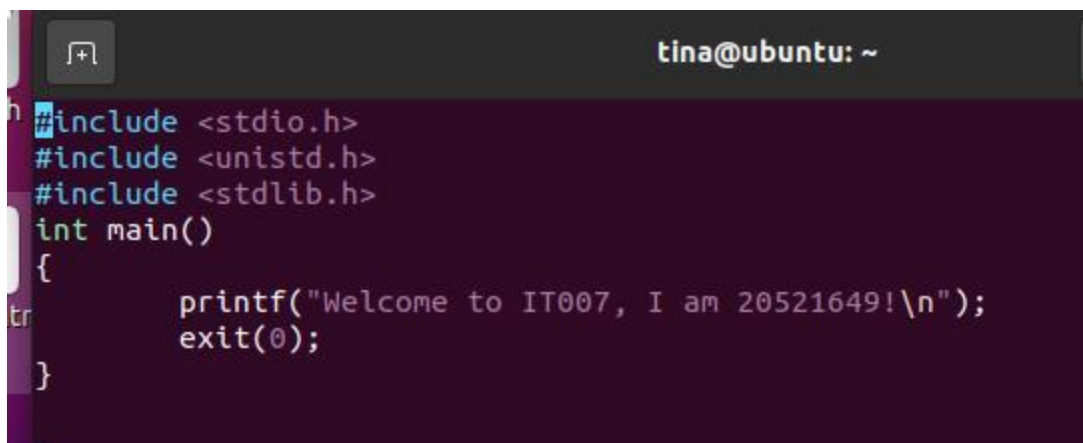
```

tina@ubuntu: ~
VuongDinhThanhNgan_20521649@ubuntu:~$ vim cau4.c

```

Hình 4.1: Tạo file bằng lệnh vim cau4.c

- Sau khi tạo được file, code đoạn chương trình như sau



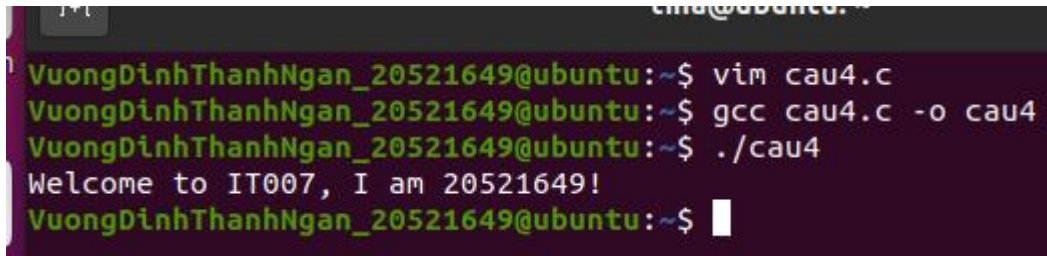
```

#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
int main()
{
    printf("Welcome to IT007, I am 20521649!\n");
    exit(0);
}

```

Hình 4.2: Sử dụng lệnh printf để in chuỗi ra màn hình

- Đoạn code sử dụng lệnh `printf("Welcome to IT007, I am 20521649!\n")` để in dòng dữ liệu ra màn hình.
- Sử dụng lệnh `gcc cau4.c -o cau4` để biên dịch chương trình, khi biên dịch thành công, dùng tiếp lệnh `./cau4` để thực thi chương trình, kết quả như hình dưới




```
VuongDinhThanhNgan_20521649@ubuntu:~$ vim cau4.c
VuongDinhThanhNgan_20521649@ubuntu:~$ gcc cau4.c -o cau4
VuongDinhThanhNgan_20521649@ubuntu:~$ ./cau4
Welcome to IT007, I am 20521649!
VuongDinhThanhNgan_20521649@ubuntu:~$
```

Hình 4.3: Biên dịch và thực thi chương trình

b. Mở tệp `abcd.txt` bằng vim editor

- Để mở tệp `abcd.txt` bằng vim editor, ta sử dụng lệnh `system("vim ~/abcd.txt")` trong đoạn chương trình



```
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
int main()
{
    printf("Welcome to IT007, I am 20521649!\n");
    system("vim ~/abcd.txt");
    exit(0);
}
```

Hình 4.4: Sử dụng lệnh `system("vim ~/abcd.txt")`

- Biên dịch và chạy chương trình, ta có tệp `abcd.txt` như hình dưới.



Hình 4.5: File `abcd.txt` được mở

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

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

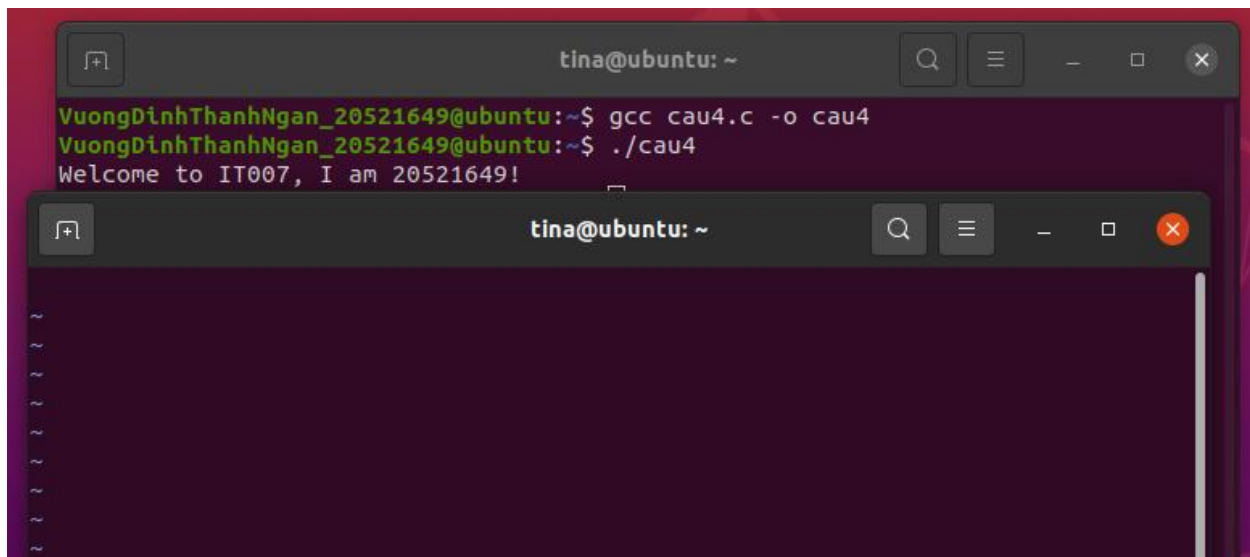
- Tắt vim sau khi người dùng nhấn CTRL+C.
- + Bước 1: Tạo vim ở một terminal mới bằng lệnh `system("gnome-terminal -- bash -c 'vim ~/abcd.txt'")`
- + Bước 2: Tạo một biến cờ hiệu `close_vim`, gán giá trị bằng 1, sử dụng vòng lặp `while` để lặp vô hạn.
- + Bước 3: Vì không thể tắt bằng cách thông thường, nên ta sử dụng một hàm `signal(SIGINT, on_sigint)`, trong đó hàm `on_sigint()` sử dụng lệnh `system("kill -9 `pidof vim`")` để gửi tín hiệu đến vim cần tắt. Và cũng trong hàm `on_sigint()`, ta gán biến `close_vim = 0`, để khi thực hiện thành công việc tắt vim, biến `close_vim` làm vòng lặp `while` dừng lại và đi tới dòng lệnh tiếp theo.
- Từ các bước trên, khi người dùng nhấn CTRL+C, dòng lệnh `printf("You press CTRL+C! Goodbye!")` sẽ được thực thi và in dòng chữ ra màn hình.
- Cụ thể đoạn chương trình và kết quả được thể hiện ở các hình dưới đây:

```

1 #include <stdio.h>
2 #include <unistd.h>
3 #include <stdlib.h>
4 #include <signal.h>
5 #include <sys/wait.h>
6
7 int close_vim = 1;
8
9 void on_sigint()
10 {
11     system("kill -9 `pidof vim`");
12     close_vim = 0 ;
13 }
14 int main()
15 {
16     printf("Welcome to IT007, I am 20521649!\n");
17     //system("vim ~/abcd.txt");
18     system("gnome-terminal -- bash -c 'vim ~/abcd.txt'");
19     signal(SIGINT, on_sigint);
20     while(close_vim){}
21     printf("You press Ctrl + C! Goodbye!\n");
22     exit(0);

```

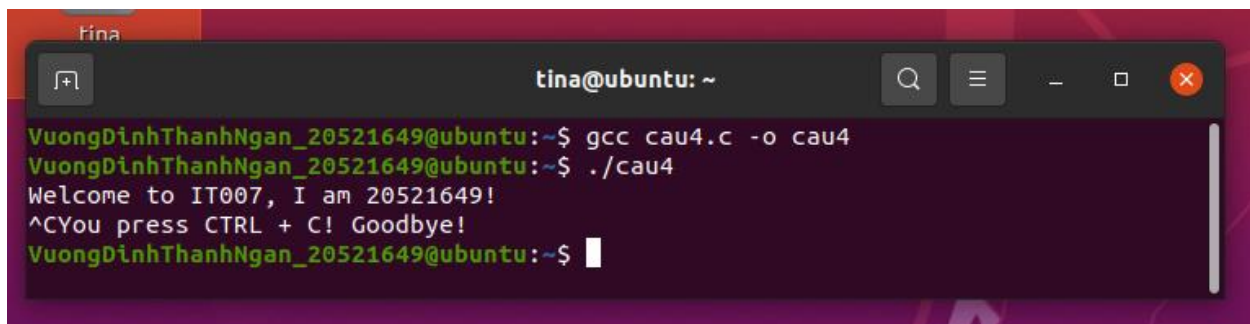
Hình 4.6: Đoạn chương trình của câu hỏi c,d



A terminal window titled 'tina@ubuntu: ~' with search, menu, and window control icons. The prompt is 'VuongDinhThanhNgan_20521649@ubuntu:~\$'. The user enters 'gcc cau4.c -o cau4' and then './cau4'. The program outputs 'Welcome to IT007, I am 20521649!'. Below the terminal, there are several tilde (~) characters.

```
tina@ubuntu: ~
VuongDinhThanhNgan_20521649@ubuntu:~$ gcc cau4.c -o cau4
VuongDinhThanhNgan_20521649@ubuntu:~$ ./cau4
Welcome to IT007, I am 20521649!
```

Hình 4.7: Kết quả sau khi tạo vòm ở terminal mới



A terminal window titled 'tina@ubuntu: ~' with search, menu, and window control icons. The prompt is 'VuongDinhThanhNgan_20521649@ubuntu:~\$'. The user enters 'gcc cau4.c -o cau4' and then './cau4'. The program outputs 'Welcome to IT007, I am 20521649!' and '^CYou press CTRL + C! Goodbye!'. The prompt returns to 'VuongDinhThanhNgan_20521649@ubuntu:~\$'.

```
tina@ubuntu: ~
VuongDinhThanhNgan_20521649@ubuntu:~$ gcc cau4.c -o cau4
VuongDinhThanhNgan_20521649@ubuntu:~$ ./cau4
Welcome to IT007, I am 20521649!
^CYou press CTRL + C! Goodbye!
VuongDinhThanhNgan_20521649@ubuntu:~$
```

Hình 4.8: Kết quả khi người dùng nhấn CTRL+C