

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA KHOA HỌC VÀ KỸ THUẬT THÔNG TIN



UIT
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

HỆ ĐIỀU HÀNH
IT007.O21.LT

BÁO CÁO
BÁO CÁO THỰC HÀNH LAB 3

Sinh viên thực hiện:
Trần Duy Tân - 22550020

Giảng viên:
ThS. Đỗ Trí Nhựt

Thành phố Hồ Chí Minh, năm 2024

3.3.2 Trình biên dịch gcc

```
tant9@tant9-VMware-Virtual-Platform:~/Desktop/UIT/lab3$ gcc hello.c -o hello
tant9@tant9-VMware-Virtual-Platform:~/Desktop/UIT/lab3$ ./hello
Hello, I am <TranDuyTan>,
Welcome to IT007!
tant9@tant9-VMware-Virtual-Platform:~/Desktop/UIT/lab3$
```

3.3.3 Makefile

```
all: hello run

hello:
    gcc main.c hello.c -o hello

run:
    ./hello

clean:
    rm -f hello

~
~
~
```

```
all: hello run

hello:
    gcc main.c hello.c -o hello

run:
    ./hello

clean:
    rm -f hello

.PHONY: hello

~
```

3.3.4 Trình gỡ lỗi

```
tantd9@tantd9-VMware-Virtual-Platform: ~/Desktop/UIT/lab3
For bug reporting instructions, please see:
<https://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from factorial...
(gdb) break 10
Breakpoint 1 at 0x1195: file factorial.c, line 10.
(gdb) run
Starting program: /home/tantd9/Desktop/UIT/lab3/factorial

This GDB supports auto-downloading debuginfo from the following URLs:
<https://debuginfod.ubuntu.com>
Enable debuginfod for this session? (y or [n]) y
Debuginfod has been enabled.
To make this setting permanent, add 'set debuginfod enabled on' to .gdbinit.
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".

Breakpoint 1, main () at factorial.c:10
10      {
(gdb)
```

Thử đặt breakpoint ở dòng khác

```
tantd9@tantd9-VMware-Virtual-Platform: ~/Desktop/UIT/lab3
<https://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from factorial...
(gdb) break 15
Breakpoint 1 at 0x11d3: file factorial.c, line 15.
(gdb) run
Starting program: /home/tantd9/Desktop/UIT/lab3/factorial

This GDB supports auto-downloading debuginfo from the following URLs:
<https://debuginfod.ubuntu.com>
Enable debuginfod for this session? (y or [n]) y
Debuginfod has been enabled.
To make this setting permanent, add 'set debuginfod enabled on' to .gdbinit.
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
Enter the number: 1

Breakpoint 1, main () at factorial.c:15
15      for (i=1; i<num; i++)
(gdb)
```

3.4.1.2 Tiến trình trong môi trường Linux

```

tantd9@tantd9-VMware-Virtual-Platform: ~/Desktop/UIT/lab3
tantd9@tantd9-VMware-Virtual-Platform:~/Desktop/UIT/lab3$ ps -f
ID          PID      PPID    C  STIME TTY          TIME CMD
tantd9      25550    25543   1  15:40 pts/0      00:00:00 bash
tantd9      25556    25550  99  15:40 pts/0      00:00:00 ps -f
tantd9@tantd9-VMware-Virtual-Platform:~/Desktop/UIT/lab3$ ps -a
    PID TTY          TIME CMD
    2610 tty2      00:00:00 gnome-session-b
    25560 pts/0      00:00:00 ps
tantd9@tantd9-VMware-Virtual-Platform:~/Desktop/UIT/lab3$ ps -x
    PID TTY      STAT   TIME COMMAND
    2548 ?        Ss      0:03 /usr/lib/systemd/systemd --user
    2549 ?        S        0:00 (sd-pam)
    2560 ?        S<sl    2:51 /usr/bin/pipewire
    2564 ?        Ssl     0:00 /usr/bin/pipewire -c filter-chain.conf
    2567 ?        S<sl    0:08 /usr/bin/wireplumber
    2572 ?        S<sl    0:04 /usr/bin/pipewire-pulse
    2574 ?        Sslsl   0:00 /usr/bin/gnome-keyring-daemon --foreground --compon
    2575 ?        Ss      0:06 /usr/bin/dbus-daemon --session --address=systemd: -
    2605 tty2     Ssl+    0:00 /usr/libexec/gdm-wayland-session env GNOME_SHELL_SE
    2610 tty2     Sl+     0:00 /usr/libexec/gnome-session-binary --session=ubuntu
    2650 ?        Ssl     0:00 /usr/libexec/gcr-ssh-agent --base-dir /run/user/100
    2652 ?        Ssl     0:00 /usr/libexec/gnome-session-ctl --monitor
    2670 ?        Ssl     0:00 /usr/libexec/gvfsd
    2687 ?        Sl      0:00 /usr/libexec/gvfsd-fuse /run/user/1000/gvfs -f

```

```

tantd9@tantd9-VMware-Virtual-Platform:~/Desktop/UIT/lab3$ ps -u
USER      PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
tantd9    2605   0.0   0.1 235668  5888 tty2     Ssl+  09:43   0:00 /usr/libexec/
tantd9    2610   0.0   0.4 298200 16384 tty2     Sl+   09:43   0:00 /usr/libexec/
tantd9    25550   0.1   0.1 11024   5248 pts/0    Ss    15:40   0:00 bash
tantd9    25562  1100   0.1 13748   4736 pts/0    R+    15:41   0:00 ps -u
tantd9@tantd9-VMware-Virtual-Platform:~/Desktop/UIT/lab3$ ps -a
    PID TTY          TIME CMD
    2610 tty2      00:00:00 gnome-session-b
    25563 pts/0      00:00:00 ps
tantd9@tantd9-VMware-Virtual-Platform:~/Desktop/UIT/lab3$

```

3.4.1.3 Tạo tiến trình

Example_fork

```

tantd9@tantd9-VMware-Virtual-Platform:~/Desktop/UIT/lab3$ gcc example_fork.c -o example_fork
tantd9@tantd9-VMware-Virtual-Platform:~/Desktop/UIT/lab3$ ./example_fork
Parent Process, pid=25729
tantd9@tantd9-VMware-Virtual-Platform:~/Desktop/UIT/lab3$ Child Process, pid=0
tantd9@tantd9-VMware-Virtual-Platform:~/Desktop/UIT/lab3$

```

Example_execl


```
tantd9@tantd9-VMware-Virtual-Platform:~/Desktop/UIT/lab3$ gcc example_thread_selfexit.c -o example_thread_selfexit
tantd9@tantd9-VMware-Virtual-Platform:~/Desktop/UIT/lab3$ ./example_thread_selfexit
I'm Main Thread: create Thread: #0
I'm Main Thread: create Thread: #1
Hello IT007! I'm Thread #0 ^_^!!!
Hello IT007! I'm Thread #1 ^_^!!!
```

3.4.2.3 Tạo tiểu trình

```
tantd9@tantd9-VMware-Virtual-Platform: ~/Desktop/UIT/lab3
```

```
Hello, How are you?  
Hello, How are you?  
Hello, How are you?  
Hello, How are you?  
Hello, How are you?  
Hello, How are you?  
Hello, How are you?  
Hello, How are you?  
Hello, How are you?  
Hello, How are you?  
Hello, How are you?  
Hello, How are you?  
Hello, How are you?  
Hello, How are you?  
Hello, How are you?  
Hello, How are you?  
Hello, How are you?  
I'm fine, and you?  
I'm fine, and you?  
I'm fine, and you?  
I'm fine, and you?  
I'm fine, and you?
```

3.4.2.4 Dừng tiểu trình

```

tantd9@tantd9-VMware-Virtual-Platform:~/Desktop/UIT/lab3$ gcc example_thread_selfexit.c -o example_thread_selfexit
tantd9@tantd9-VMware-Virtual-Platform:~/Desktop/UIT/lab3$ ./example_thread_selfexit
I'm Main Thread: create Thread: #0
I'm Main Thread: create Thread: #1
Hello IT007! I'm Thread #0 ^_ ^!!!
Hello IT007! I'm Thread #1 ^_ ^!!!
tantd9@tantd9-VMware-Virtual-Platform:~/Desktop/UIT/lab3$

```

3.4.2.5 Hợp và gỡ tiểu trình

```

tantd9@tantd9-VMware-Virtual-Platform:~/Desktop/UIT/lab3$ gcc example_thread_join.c -o example_thread_join
tantd9@tantd9-VMware-Virtual-Platform:~/Desktop/UIT/lab3$ ./example_thread_join
I'm Main Thread: create Thread: #0
I'm Main Thread: create Thread: #1
Hello IT007! I'm Thread #1 ^_ ^!!!
Hello IT007! I'm Thread #0 ^_ ^!!!

```

3.4.2.6 Truyền dữ liệu cho tiểu trình

```

tantd9@tantd9-VMware-Virtual-Platform:~/Desktop/UIT/lab3$ ./example_thread_structure
X
X
X
X
X
X

```

3.4.3.7 Signal (Truyền thông giữa các tiến trình)

```

tantd9@tantd9-VMware-Virtual-Platform:~/Desktop/UIT/lab3$ gcc example_signal.c -o example_signal
tantd9@tantd9-VMware-Virtual-Platform:~/Desktop/UIT/lab3$ kill -l
 1) SIGHUP      2) SIGINT      3) SIGQUIT     4) SIGILL      5) SIGTRAP
 6) SIGABRT     7) SIGBUS     8) SIGFPE     9) SIGKILL    10) SIGUSR1
11) SIGSEGV    12) SIGUSR2    13) SIGPIPE    14) SIGALRM    15) SIGTERM
16) SIGSTKFLT  17) SIGCHLD   18) SIGCONT    19) SIGSTOP    20) SIGTSTP
21) SIGTTIN    22) SIGTTOU   23) SIGURG     24) SIGXCPU    25) SIGXFSZ
26) SIGVTALRM  27) SIGPROF   28) SIGWINCH   29) SIGIO       30) SIGPWR
31) SIGSYS     34) SIGRTMIN  35) SIGRTMIN+1 36) SIGRTMIN+2 37) SIGRTMIN+3
38) SIGRTMIN+4 39) SIGRTMIN+5 40) SIGRTMIN+6 41) SIGRTMIN+7 42) SIGRTMIN+8
43) SIGRTMIN+9 44) SIGRTMIN+10 45) SIGRTMIN+11 46) SIGRTMIN+12 47) SIGRTMIN+13
48) SIGRTMIN+14 49) SIGRTMIN+15 50) SIGRTMAX-14 51) SIGRTMAX-13 52) SIGRTMAX-12
53) SIGRTMAX-11 54) SIGRTMAX-10 55) SIGRTMAX-9 56) SIGRTMAX-8 57) SIGRTMAX-7
58) SIGRTMAX-6 59) SIGRTMAX-5 60) SIGRTMAX-4 61) SIGRTMAX-3 62) SIGRTMAX-2
63) SIGRTMAX-1 64) SIGRTMAX
tantd9@tantd9-VMware-Virtual-Platform:~/Desktop/UIT/lab3$ gcc example_signal.c -o example_signal
tantd9@tantd9-VMware-Virtual-Platform:~/Desktop/UIT/lab3$ ./example_signal
^C
CRT+C is pressed!
tantd9@tantd9-VMware-Virtual-Platform:~/Desktop/UIT/lab3$

```

3.5 Bài tập ôn tập

1. Mối quan hệ cha-con giữa các tiến trình

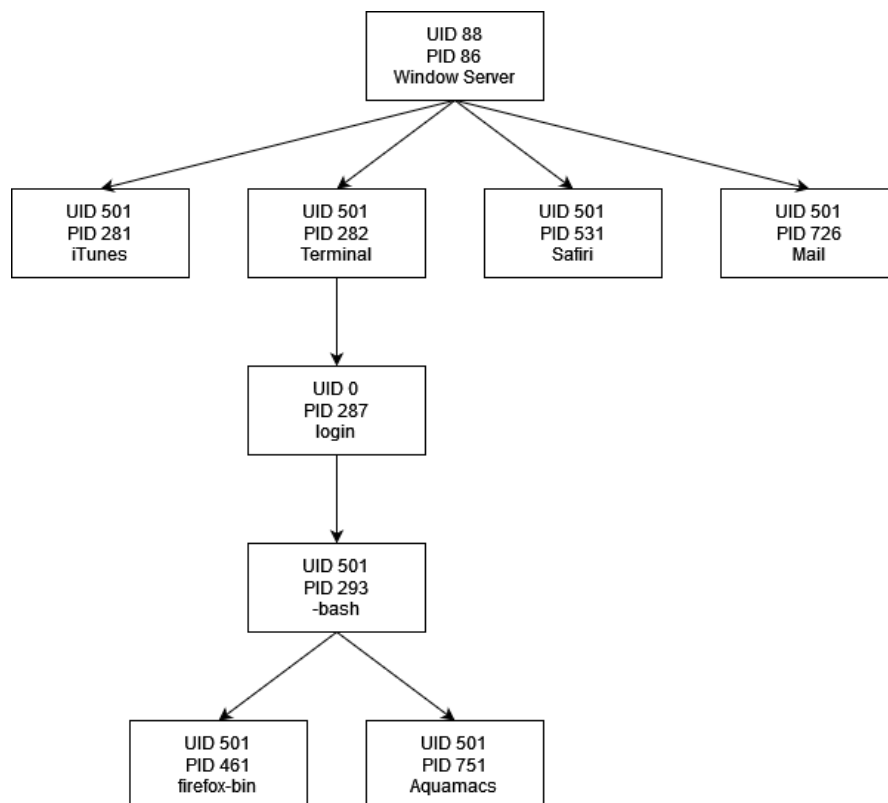
a. Vẽ cây quan hệ parent-child của các tiến trình bên dưới:

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 tiến trình dựa vào PID của nó.

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.

Bài làm:



b. Trình bày cách sử dụng lệnh ps để tìm kiếm trình cha của 1 tiến trình dựa trên PID của nó.

- Ta sẽ dùng lệnh ps -f [pid của tiến trình con] để hiển thị pid của tiến trình cha (chính là PPID).

- Ta sử dụng lệnh ps -f pid lần nữa với pid là pid của tiến trình cha vừa tìm thấy để xem thông tin của tiến trình cha.

```

tantd9@tantd9-VMware-Virtual-Platform:~/Desktop/UIT/lab3$ ps -f
UID          PID    PPID  C STIME TTY          TIME CMD
tantd9       30428   30421  0  18:12 pts/0        00:00:00 bash
tantd9       31442   30428  99  18:53 pts/0        00:00:00 ps -f
tantd9@tantd9-VMware-Virtual-Platform:~/Desktop/UIT/lab3$
  
```

```

tantd9@tantd9-VMware-Virtual-Platform:~/Desktop/UIT/lab3$ pstree -sg 30428
systemd(1)---systemd(2548)---gnome-terminal-(30421)---bash(30428)---pstree(31460)
tantd9@tantd9-VMware-Virtual-Platform:~/Desktop/UIT/lab3$
  
```

c. Tìm hiểu và cài đặt lệnh pstree, sau đó trình bày cách sử dụng lệnh này để tìm kiếm tiến trình cha dự báo pid của nó.

- Sử dụng lệnh pstree -p -s để hiển thị cả cây tiến trình tương ứng với tiến trình con được nhập vào.

```
tant9@tant9-VMware-Virtual-Platform:~/Desktop/UIT/lab3$ pstree -s
systemd├─ModemManager─3*[{ModemManager}]
      ├─NetworkManager─3*[{NetworkManager}]
      ├─VGAuthService
      ├─accounts-daemon─3*[{accounts-daemon}]
      ├─avahi-daemon─avahi-daemon
      ├─bluetoothd
      ├─colord─3*[{colord}]
      ├─cron
      ├─cups-browsed─3*[{cups-browsed}]
      ├─cupsd
      ├─dbus-daemon
      ├─fwupd─5*[{fwupd}]
      └─gdm3├─gdm-session-wor├─gdm-wayland-ses└─gnome-session-b─3*[{gnome-session-b}]
          │               │               │   3*[{gdm-wayland-ses}]
          │               └─3*[{gdm-session-wor}]
          └─3*[{gdm3}]
      ├─gnome-remote-de─3*[{gnome-remote-de}]
      ├─gpg-agent
      ├─2*[{kerneloops}]
      ├─polkitd─3*[{polkitd}]
      ├─power-profiles─3*[{power-profiles-}]
      ├─rsyslogd─3*[{rsyslogd}]
      ├─rtkit-daemon─2*[{rtkit-daemon}]
      └─snapd─9*[{snapd}]
```

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

```
/*#####
# University of Information Technology #
# IT007 Operating System #
# <Your name>, <your Student ID> #
# File: exercise_2.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); /*wait until the child terminates
    */
    }
    printf("I see %d coconuts!\n", num_coconuts);
    exit(0);
}
```

Out Put

- Chương trình chạy sẽ xuất ra lỗi vì thiếu thư viện .

```

exercise_2.c:2:1: note: include '<stdlib.h>' or provide a declaration of 'exit'
  1 | #include<stdio.h>
+++ |+#include <stdlib.h>
  2 | int main(){
exercise_2.c:10:2: warning: implicit declaration of function 'wait' [-Wimplicit-
function-declaration]
 10 | wait(NULL); /*wait until the child terminates */
    |
exercise_2.c:13:1: warning: incompatible implicit declaration of built-in functi
on 'exit'
 13 | exit(0);
    |
exercise_2.c:13:1: note: include '<stdlib.h>' or provide a declaration of 'exit'

```

- Chương trình chạy sẽ xuất ra lỗi vì thiếu thư viện .
- Thêm 3 thư viện `stdlib.h` , `unistd.h`, `sys/wait.h`

```

tantd9@tantd9-VMware-Virtual-Platform:~/Desktop/UIT/lab3$ gcc exercise_2.c -o ex
ercise
tantd9@tantd9-VMware-Virtual-Platform:~/Desktop/UIT/lab3$ ./exercise
I see 17 coconuts!
tantd9@tantd9-VMware-Virtual-Platform:~/Desktop/UIT/lab3$

```

- Kết quả chương trình in ra là “I see 17 coconuts” . Vì vào lệnh `fork()` thì tiến trình con và cha sẽ chạy. Với `pid = 0` thì tiến trình con sẽ thực hiện chương trình, còn tiến trình cha (`pid > 0`) sẽ vào lệnh `wait (NULL)` vậy nên tiến trình cha sẽ đợi đến khi tiến trình con chạy xong.
- Tiến trình con thực hiện lệnh `exit(0)` (lệnh kết thúc) trước khi thực hiện vậy nên lệnh `printf` của tiến trình cha sẽ được thực thi , 2 tiến trình con và cha có bộ nhớ riêng vậy nên biết `num_coconuts = 17`. Thực hiện in ra ‘ I see 17 coconuts!’”

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	Chức năng
<code>Pthread_attr_init</code>	Khởi tạo giá trị mặc định cho đối tượng thuộc tính
<code>Pthread_attr_destroy</code>	Xóa bộ nhớ được cấp phát trong quá trình khởi tạo
<code>Pthread_attr_getschedparam</code>	Trả về các tham số lịch trình (Scheduling Parameter) được xác định bởi <code>pthread_attr_setschedparam ()</code>
<code>Pthread_attr_getschedpolicy</code>	Đề xuất scheduling policy của thread

Pthread_attr_getdetachstate	Lấy truy xuất trạng thái khởi tạo của thread, có thể là riêng lẻ hoặc kết hợp
Pthread_attr_getinheritsched	Trả về chính sách lịch trình (scheduling policy) được set bởi pthread_attr_setinheritsched ().
Pthread_attr_getscope	Truy xuất phạm vi của thread
Pthread_attr_setdetachstate	Sử dụng lại ID và tài nguyên của thread khi nó bị ngắt mà không phải chờ nếu thread có thuộc tính riêng lẻ.
Pthread_attr_setguardsize	Set kích thước của khu vực an toàn của thread
Pthread_attr_setstackaddr	Set địa chỉ stack của thread
Pthread_attr_setstacksize	Set kích thước stack của thread
Pthread_attr_getguardsize	Lấy kích thước của khu vực an toàn của thread
Pthread_attr_getstackaddr	Trả về địa chỉ stack của thread được set bởi pthread_attr_setstackaddr ()
Pthread_attr_getstacksize	Trả về kích thước stack của thread được set bởi pthread_attr_setstacksize ().

Pthread_attr_init

```
int pthread_attr_init(pthread_attr_t *tattr);
#include <pthread.h>

pthread_attr_t tattr;
int ret;

/* khởi tạo giá trị mặc định cho thuộc tính */
ret = pthread_attr_init(&tattr);
```

Pthread_attr_destroy

```
int pthread_attr_destroy(pthread_attr_t *tattr);
#include <pthread.h>
```

```
pthread_attr_t tattr;
int ret;

/* Xóa thuộc tính */
ret = pthread_attr_destroy(&tattr);
```

Pthread_attr_getschedparam

```
int pthread_attr_getschedparam(pthread_attr_t *tattr,
    const struct sched_param *param);
#include <pthread.h>

pthread_attr_t attr;
struct sched_param param;
int ret;

/* Lấy tham số lịch trình đang tồn tại trong thread */
ret = pthread_attr_getschedparam (&tattr, &param);
```

Pthread_attr_getschedpolicy

```
int pthread_attr_getschedpolicy(pthread_attr_t *tattr, int *policy);
#include <pthread.h>

pthread_attr_t tattr;
int policy;
int ret;

/* Lấy chính sách lịch trình của thread */
ret = pthread_attr_getschedpolicy (&tattr, &policy);
```

Pthread_attr_getdetachstate

```
int pthread_attr_getdetachstate(const pthread_attr_t *tattr,
```

```

    int *detachstate;
#include <pthread.h>

pthread_attr_t tattr;
int detachstate;
int ret;

/* Lấy trạng thái khởi tạo của thread */
ret = pthread_attr_getdetachstate (&tattr, &detachstate);

```

Pthread_attr_getinheritsched

```

int pthread_attr_getinheritsched(pthread_attr_t *tattr, int *inherit);
#include <pthread.h>

pthread_attr_t tattr;
int inherit;
int ret;

/* Lấy chính sách lịch trình và độ ưu tiên của thread được khởi tạo */
ret = pthread_attr_getinheritsched (&tattr, &inherit);

```

Pthread_attr_getscope

```

int pthread_attr_getscope(pthread_attr_t *tattr, int *scope);
#include <pthread.h>

pthread_attr_t tattr;
int scope;
int ret;

/* Lấy phạm vi của thread */
ret = pthread_attr_getscope(&tattr, &scope);

```

Pthread_attr_setdetachstate

```
int pthread_attr_setdetachstate(pthread_attr_t *tattr,int detachstate);  
#include <pthread.h>  
  
pthread_attr_t tattr;  
int ret;  
  
/* Set trạng thái riêng lẻ cho thread */  
ret =  
pthread_attr_setdetachstate(&tattr,PTHREAD_CREATE_DETACHED)  
;
```

Pthread_attr_setguardsize

```
#include <pthread.h>  
  
int pthread_attr_setguardsize(pthread_attr_t *attr, size_t guardsize);
```

Pthread_attr_setstackaddr

```
int pthread_attr_setstackaddr(pthread_attr_t *tattr,void *stackaddr);  
#include <pthread.h>  
  
pthread_attr_t tattr;  
void *base;  
int ret;  
  
base = (void *) malloc(PTHREAD_STACK_MIN + 0x4000);  
  
/* Đặt địa chỉ mới cho thread */  
ret = pthread_attr_setstackaddr(&tattr, base);
```

Pthread_attr_setstacksize

```
int pthread_attr_setstacksize(pthread_attr_t *tattr, int size);  
#include <pthread.h>
```



```

pthread_attr_t tattr;
int size;
int ret;

size = (PTHREAD_STACK_MIN + 0x4000);

/* Đặt kích thước mới */
ret = pthread_attr_setstacksize(&tattr, size);

```

Pthread_attr_getguardsize

```

#include <pthread.h>

int pthread_attr_getguardsize(const pthread_attr_t *attr, size_t
*guardsize);

```

Pthread_attr_getstackaddr

```

int pthread_attr_getstackaddr(pthread_attr_t *tattr, void **stackaddr);
#include <pthread.h>

pthread_attr_t tattr;
void *base;
int ret;

/* Lấy một địa chỉ mới */
ret = pthread_attr_getstackaddr (&tattr, *base);

```

Pthread_attr_getstacksize

```

int pthread_attr_getstacksize(pthread_attr_t *tattr, size_t *size);
#include <pthread.h>

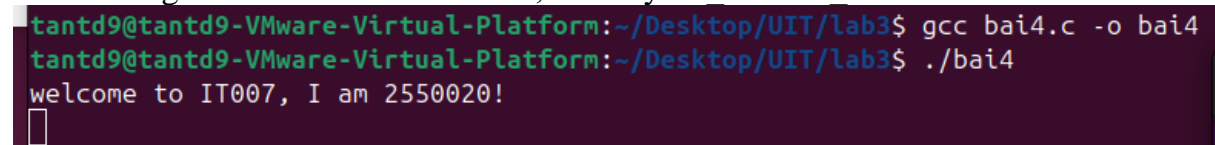
```

```
pthread_attr_t tattr;
int size;
int ret;

/* Lấy kích thước stack */
ret = pthread_attr_getstacksize(&tattr, &size);
```

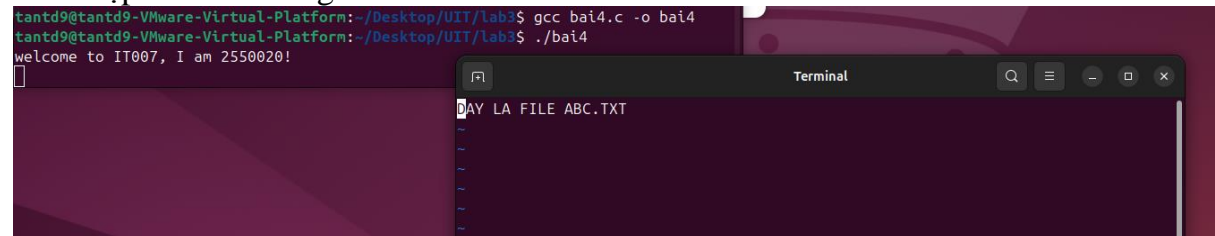
4. Viết chương trình làm các công việc sau theo thứ tự:

a. In ra dòng chữ: “Welcome to IT007, I am <your Student ID>!”



```
tantd9@tantd9-VMware-Virtual-Platform:~/Desktop/UIT/Lab3$ gcc bai4.c -o bai4
tantd9@tantd9-VMware-Virtual-Platform:~/Desktop/UIT/Lab3$ ./bai4
welcome to IT007, I am 2550020!
```

b. Mở tệp abcd.txt bằng vim editor

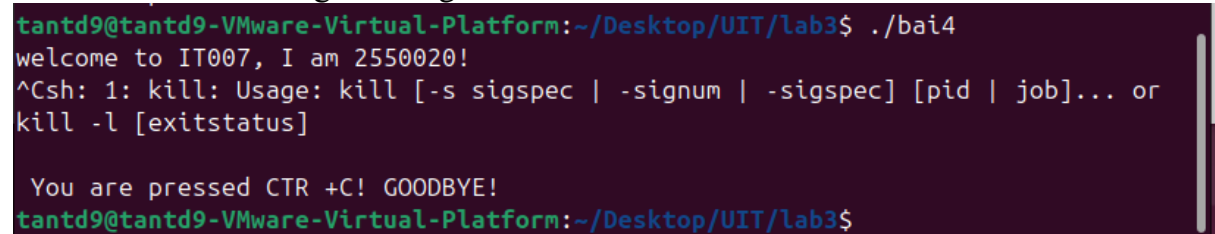


```
tantd9@tantd9-VMware-Virtual-Platform:~/Desktop/UIT/Lab3$ gcc bai4.c -o bai4
tantd9@tantd9-VMware-Virtual-Platform:~/Desktop/UIT/Lab3$ ./bai4
welcome to IT007, I am 2550020!
```

Terminal window content:

```
HAY LA FILE ABC.TXT
```

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



```
tantd9@tantd9-VMware-Virtual-Platform:~/Desktop/UIT/Lab3$ ./bai4
welcome to IT007, I am 2550020!
^Csh: 1: kill: Usage: kill [-s sigspec | -signum | -sigspec] [pid | job]... or
kill -l [exitstatus]

You are pressed CTR +C! GOODBYE!
tantd9@tantd9-VMware-Virtual-Platform:~/Desktop/UIT/Lab3$
```

Đoạn mã bài 4

```

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

#define _XOPNE_SOURCE 600
int loop = 1 ;

void sig_job(){
    system("kill -9 `pidof vim`");
    printf("\nYou are pressed CTRL+C ! GoodBye !\n");
    loop=0;
}
int main (){
    printf ("\n Welcome to IT007, I am 22550020!\n");
    system("gnome-terminal -- vim abc.txt");
    signal(SIGINT , sig_job);
    while(loop){};
    exit(0);
}
~
~

```