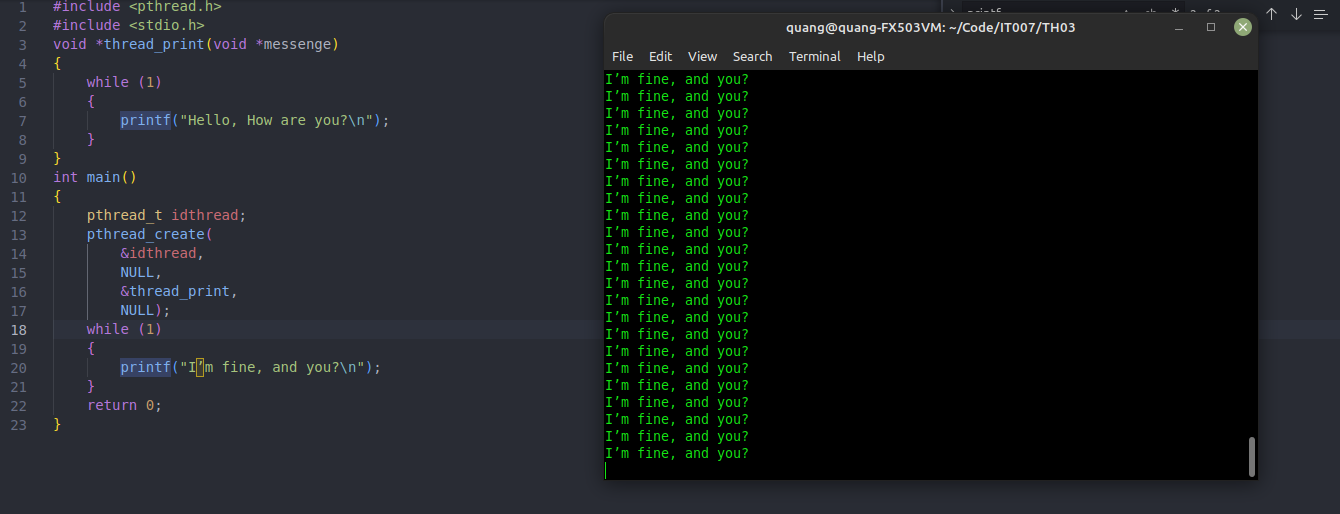
**BÀI THỰC HÀNH LAB 03**

Họ và tên: ***Nguyễn Đỗ Quang***

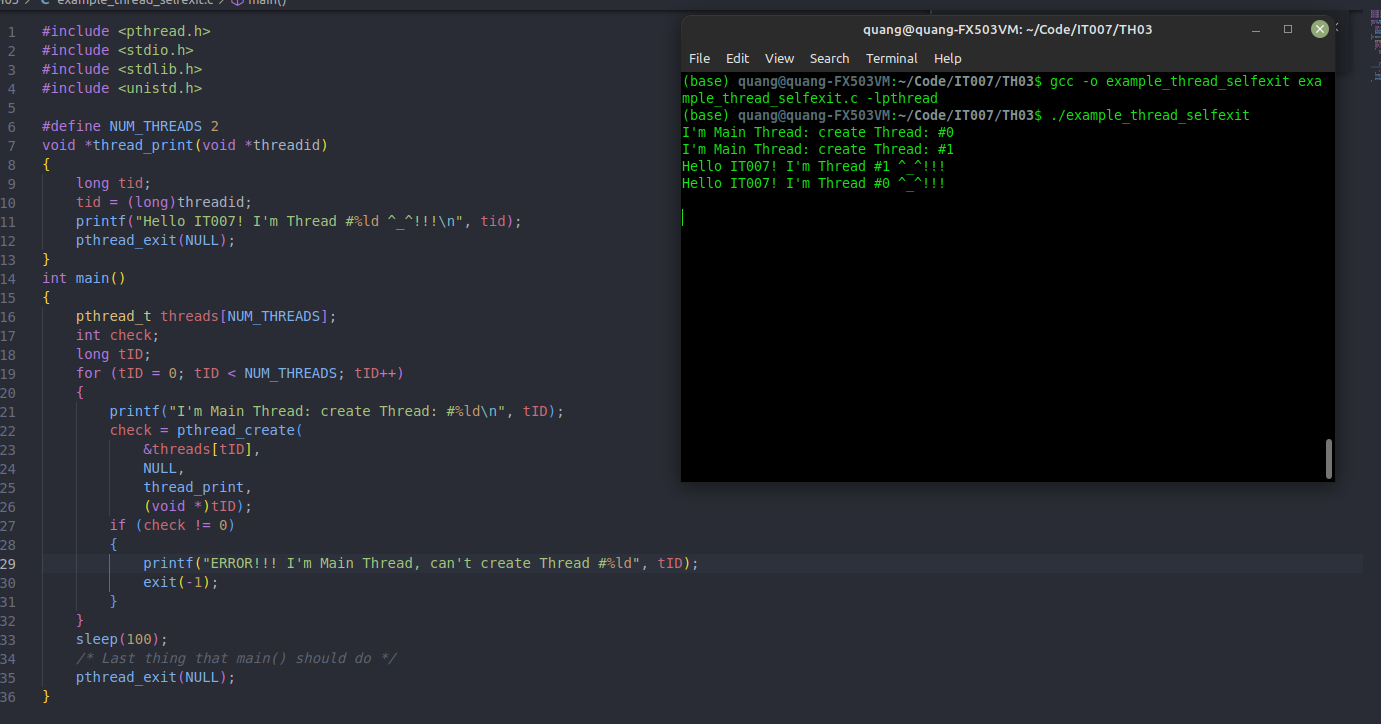
MSSV: ***20520720***

**Phần 3.4**

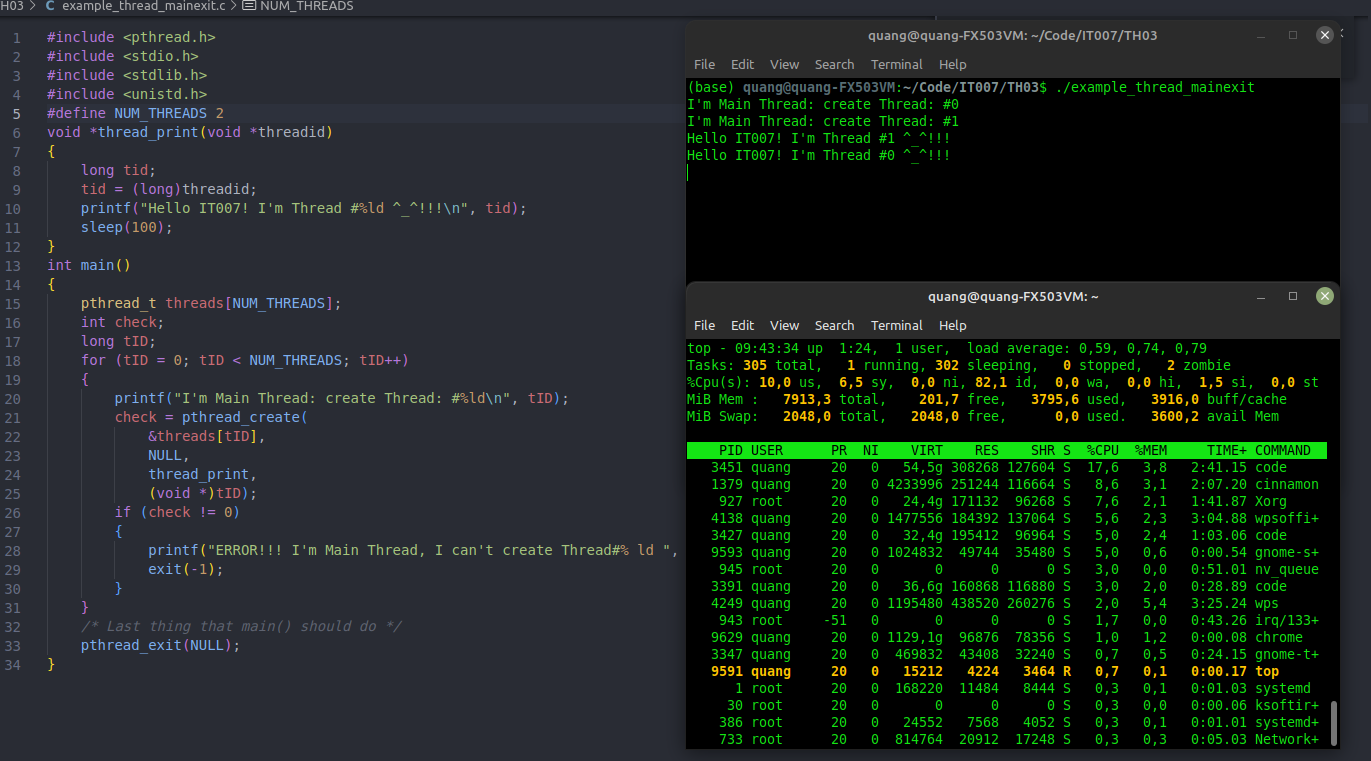
1. **example\_thread\_creation.c**



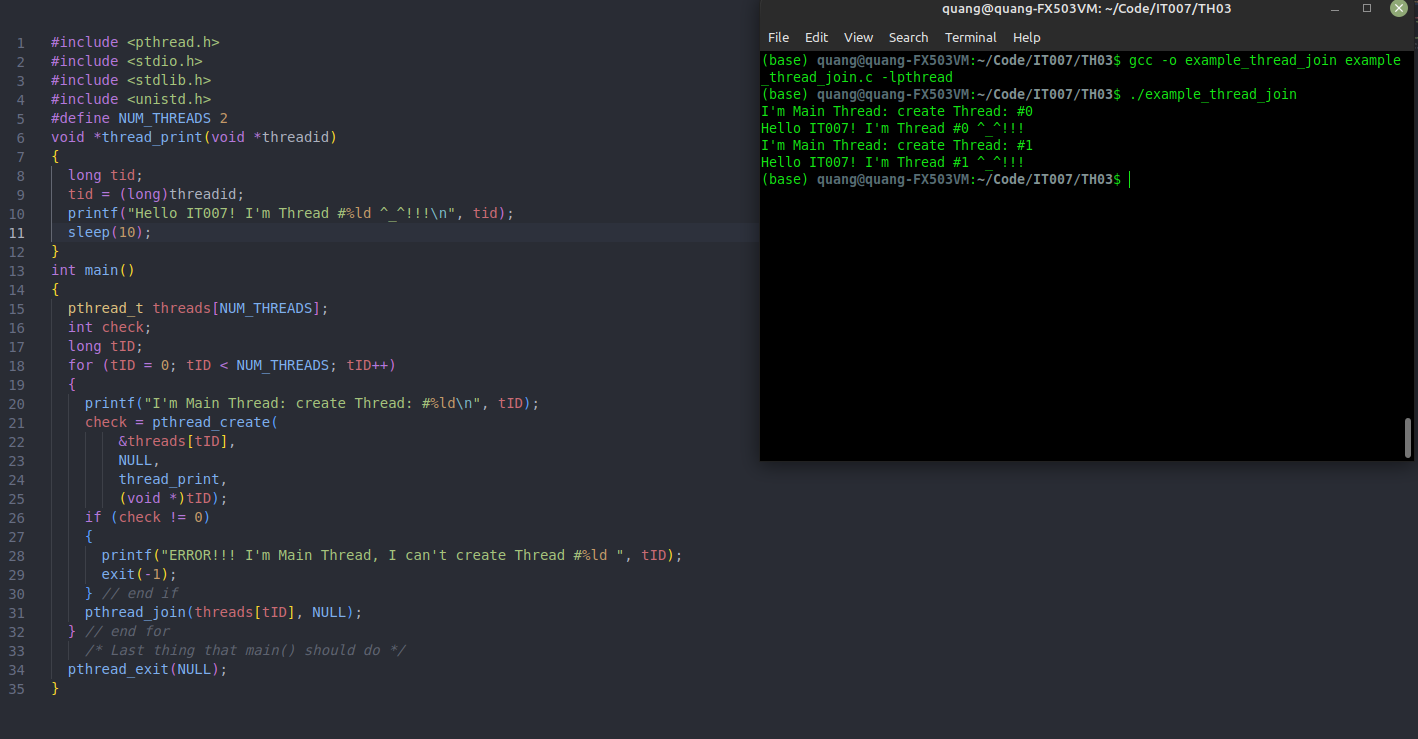
1. **example\_thread\_selfexit.c**



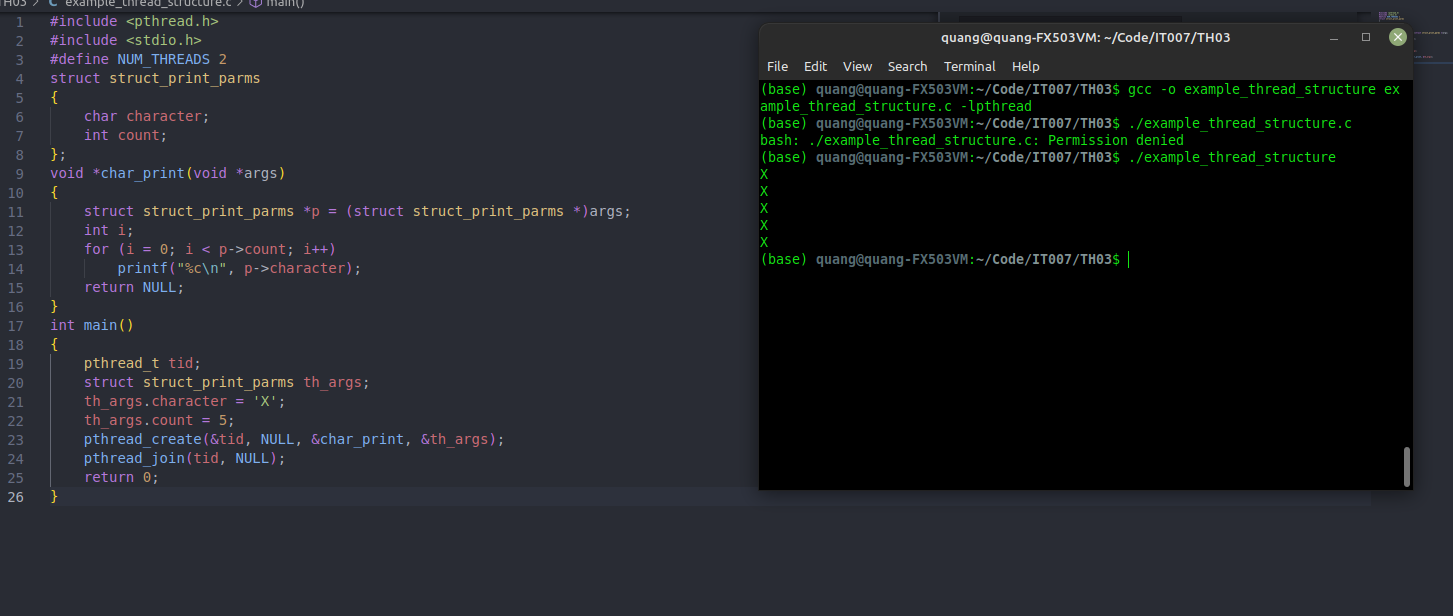
1. **example\_thread\_mainexit.c**



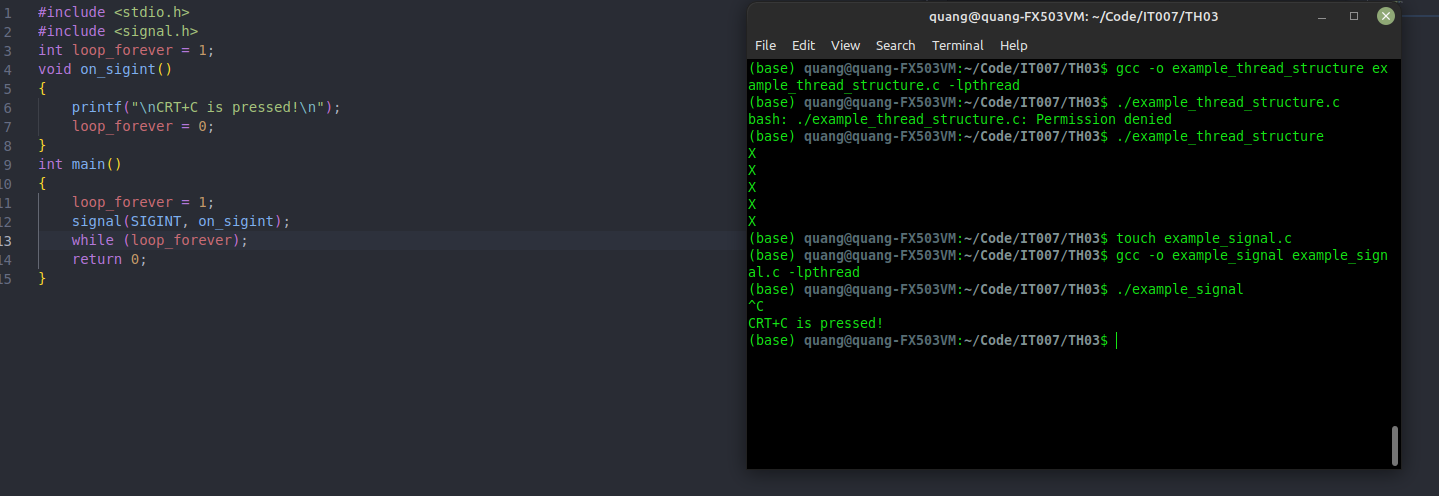
1. **example\_thread\_join.c**



1. **example\_thread\_structure**



1. **example\_signal**



**Phần 3.5**

**Câu 1:**

*a. Vẽ cây quan hệ parent-child trong đó*

**PPID:** ID của tiếng trình cha

**PID:** ID của tiến trình con

(PPID = 1) (PPID = 86) (PPID = 282) (PPID = 287) (PPID = 293)

WindowServer(86) -----> iTunes(281)

-----> Terminal(282) ----->login(287) ------>bash(293)------>firefix-bin(461)

--------->Aquamacs(751)

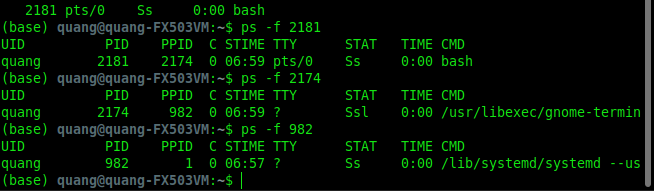
----> Safari(531)

----> Mail(726)

*b. Cách dùng lệnh ps để tìm tiến trình cha:*

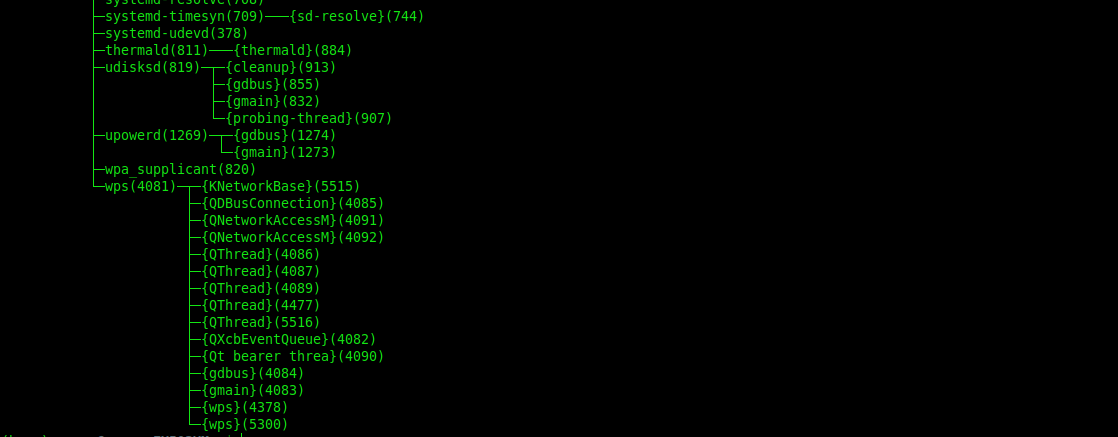
Chúng ta sử dụng lệnh ps -f [PID của tiến trình con] để hiển thị PPID của tiến trình cha.

Sử dụng tiếp lệnh ps -f [PPID của tiến trình cha] để hiển thị tiến trình PPID của tiến trình cha tiếp theo...



*c. lệnh pstree:*

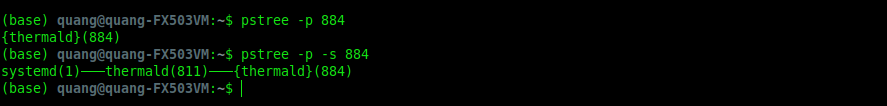
Cây tiến trình như sau:



Giả sử như tìm tiến trình thermald (PID = 884)

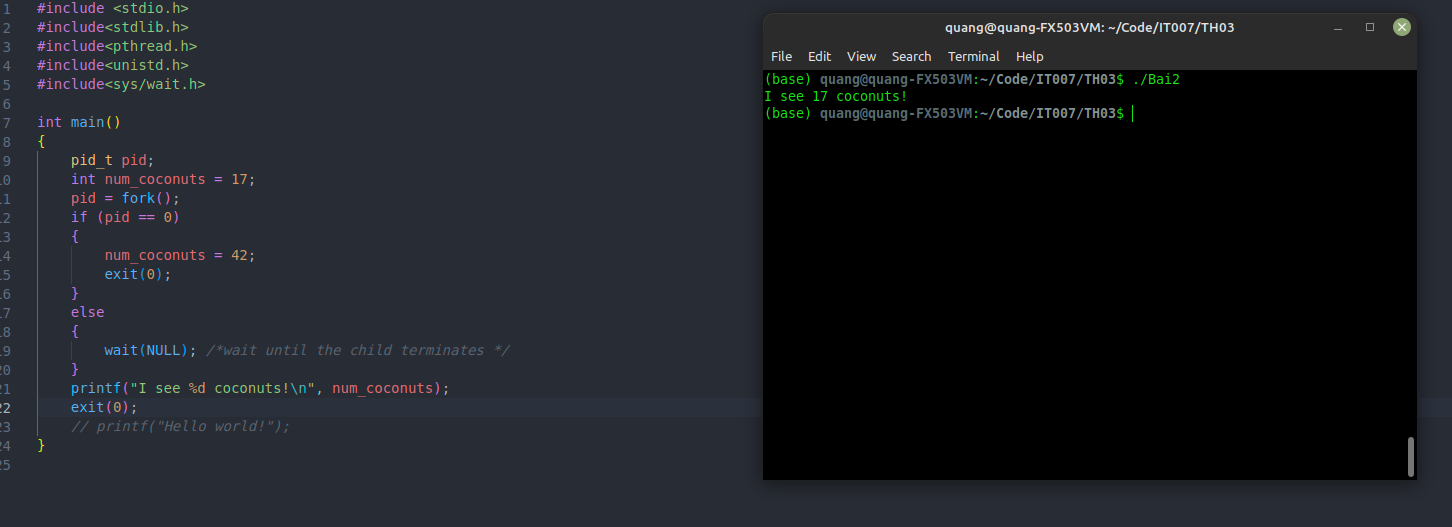
pstree -p -s [PID của tiến trình con]

pstree -p -s 884



**Câu 2:**

Chương trình sẽ in ra “I see 17 coconuts” vì tiến trình con được lệnh kết thúc (exit(0)) trước khi thực hiện câu lệnh printf, nên chỉ có câu lệnh printf của tiến trình cha được thực thi. Mà tiến trình cha và tiến trình con có tài nguyên bộ nhớ riêng nên biến num\_coconuts ở tiến trình cha vẫn là 17.



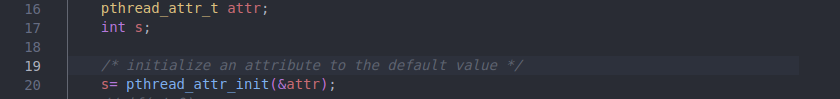
**Câu 3:**

Trình bày về POSIX thread:

Trong khi tạo ra một thread thì ta đều có thể truyền vào các thuộc tính (pthread\_attr\_t)

***Để khởi tạo một thuộc tính với giá trị mặc định***

**int pthread\_attr\_init(pthread\_attr\_t \*tattr);**



***Hủy thuộc tính***

**int pthread\_attr\_destroy(pthread\_attr\_t \*tattr);**

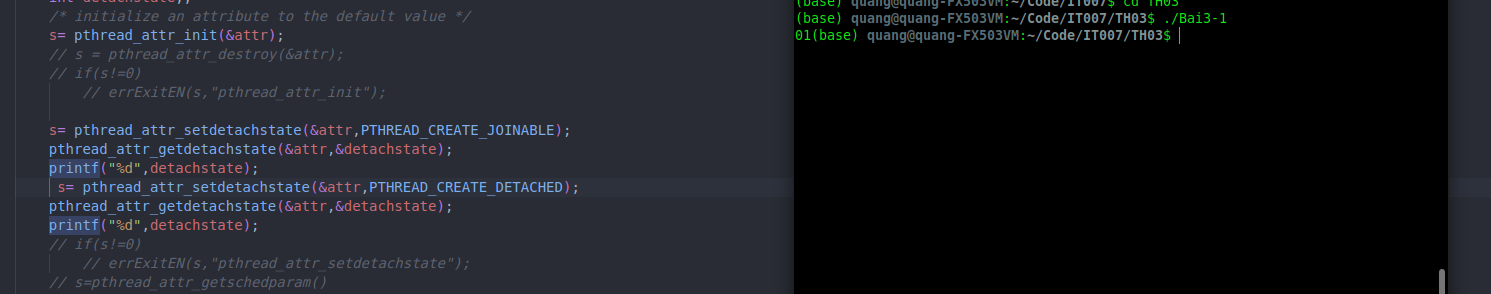
***cài đặt thuộc tính chờ:***

**int pthread\_attr\_setdetachstate(pthread\_attr\_t \*tattr,int detachstate);**

tham số detachstate ta có thể đặt:

PTHREAD\_CREATE\_DETACHED: thread ID cũng như các tài nguyên khác đều được tái sử dụng khi mà thread bị hủy

PTHREAD\_CREATE\_JOINABLE: thread sẽ đợi, dùng tham số này khi ta thực thi pthread\_join() trên thread

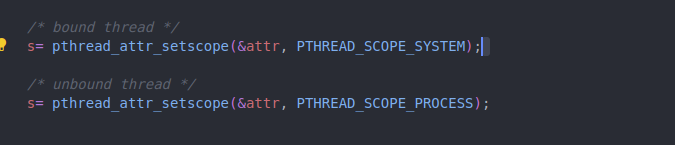


0: đặt thành công PTHREAD\_CREATE\_DETACHED

1: đặt thành công PTHREAD\_CREATE\_JOINABLE

***thuộc tính phạm vi:***

**int pthread\_attr\_setscope(pthread\_attr\_t \*tattr,int scope);**



thread có giới hạn PTHREAD\_SCOPE\_SYSTEM

thread không giới hạn PTHREAD\_SCOPE\_PROCESS

***Thiết lập thuộc tính guardsize:***

**int pthread\_attr\_setguardsize(pthread\_attr\_t \*attr, size\_t guardsize);**

tham số guardsize cấp phát thêm bộ nhớ trước khi ngăn xếp đầy, giúp bảo vệ stack pointer khỏi việc bị overflow

***Thiết lập phương thức lập lịch cho tiểu trình:***

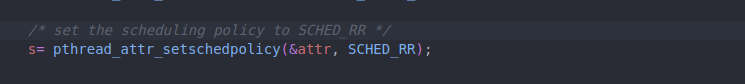
**int pthread\_attr\_setschedpolicy(pthread\_attr\_t \*tattr, int policy);**

tham số policy bao gồm

SCHED\_FIFO (giải thuật first-in-first-out)

SCHED\_RR (giải thuật round-robin)

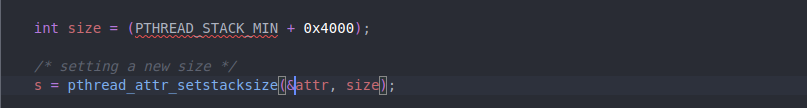
SCHED\_OTHER (giải thuật do người dùng định nghĩa)



*sử dụng giải thuật Round-Robin*

***Thiết lập thuộc tính kích thước cho ngăn xếp***

int pthread\_attr\_setstacksize(pthread\_attr\_t \*tattr, int size);



*Thiết lập kích thước mới cho ngăn xếp*

Ngoài các hàm thiết lập (set) ta còn có các hàm **get** tương tự cho mỗi hàm **set**,các hàm get được dùng để lấy giá trị thuộc tính, cũng như trả về 0 nếu các thiết lập(set) thành công.

**Câu 4: Viết chương trình thực hiện các công việc sau:**

a In ra dòng chữ “Welcome to IT007, I am 20520720”

b Mở tệp abcd.txt bằng vim editor

c Tắt vim khi người dùng nhấn CTRL + C

d Khi người dùng nhấn CTRL + C thì in ra dòng chữ “CTRL-C was pressed!Goodbye!”

