Name: Lương Toàn Bách

ID: 21521845

Class: KHTN2021

OPERATING SYSTEM
LAB 3'S REPORT

SUMMARY

Task			Status	Page
Section 3.4	Section 3.4.1	Concept of process	Done	
		Process in linux	Done	
		Create a process	Done	
		Finish a process	Done	
	Section 3.4.2	Concept of sub-process	Done	
		Sub-process in Linux	Done	
		Create a sub-process	Done	
		Finish a sub-process	Done	
		Merge and Block sub-process	Done	
		Add data to sub-process	Done	
	Section 3.4.3	Signal	Done	
Section 3.5	Section 3.5.1	Exercise_1	Done	
	Section 3.5.2	Exercise_2	Done	
	Section 3.5.3	Excersie_3	Done	
	Secion 3.5.4	Exercise_4	Done	
	Section 3.5.5	Exercise_5	Done	

Self-scrores: 10

Note: Export file to* ***PDF* *and name the file by following format:*
LAB X – <Student ID>.pdf

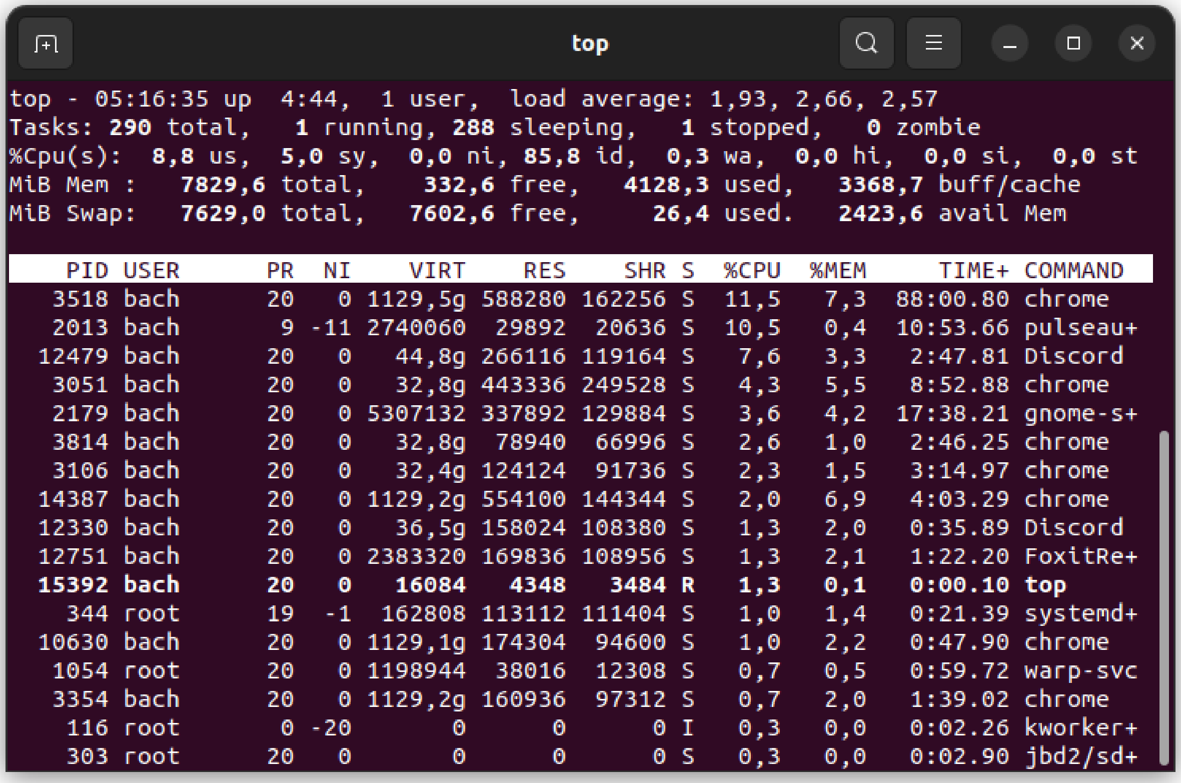
Section 3.4 Process

3.4.1. Process

3.4.1.1. Concept of process

3.4.1.2. Process in Linux

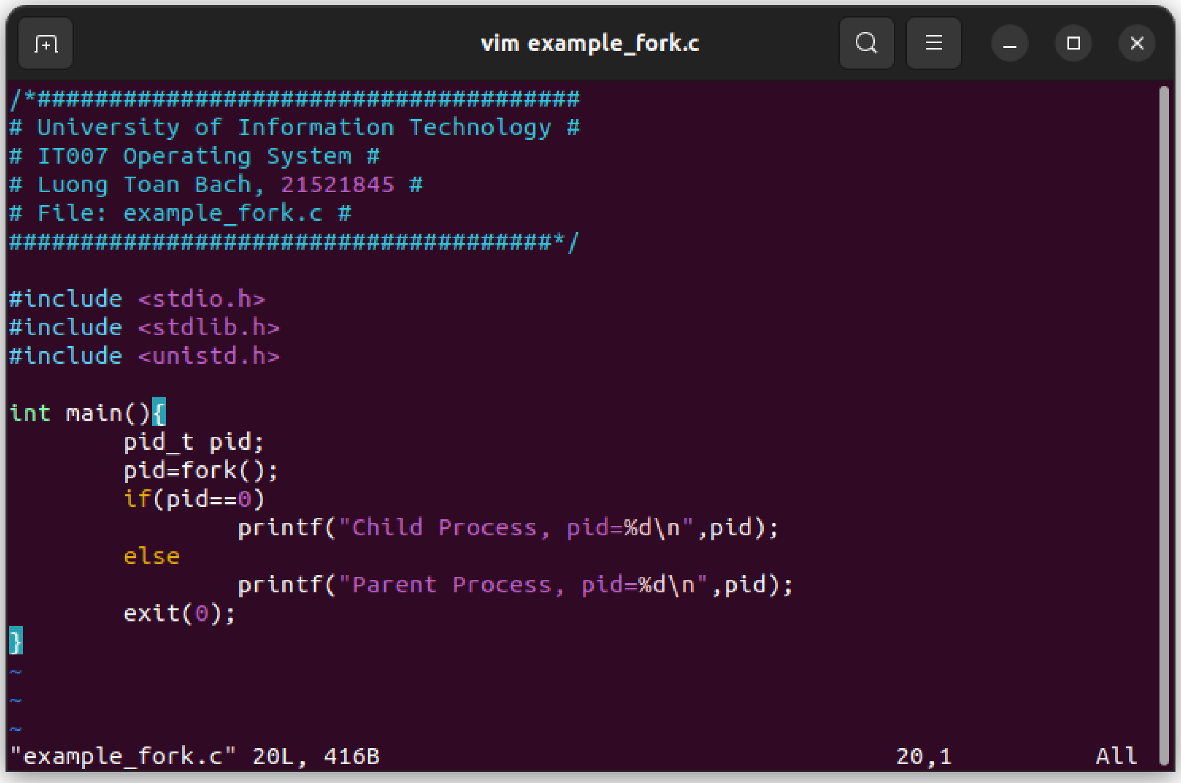
Result of running top command



```
top - 05:16:35 up  4:44,  1 user,  load average: 1,93, 2,66, 2,57
Tasks: 290 total,   1 running, 288 sleeping,   1 stopped,   0 zombie
%Cpu(s):  8,8 us,  5,0 sy,  0,0 ni, 85,8 id,  0,3 wa,  0,0 hi,  0,0 si,  0,0 st
MiB Mem :   7829,6 total,    332,6 free,   4128,3 used,   3368,7 buff/cache
MiB Swap:   7629,0 total,   7602,6 free,     26,4 used.   2423,6 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
   3518 bach      20   0 1129,5g 588280 162256 S  11,5   7,3  88:00.80 chrome
   2013 bach       9 -11 2740060  29892  20636 S  10,5   0,4  10:53.66 pulseau+
  12479 bach      20   0   44,8g 266116 119164 S   7,6   3,3   2:47.81 Discord
   3051 bach      20   0   32,8g 443336 249528 S   4,3   5,5   8:52.88 chrome
   2179 bach      20   0 5307132 337892 129884 S   3,6   4,2  17:38.21 gnome-s+
   3814 bach      20   0   32,8g  78940  66996 S   2,6   1,0   2:46.25 chrome
   3106 bach      20   0   32,4g 124124  91736 S   2,3   1,5   3:14.97 chrome
  14387 bach      20   0 1129,2g 554100 144344 S   2,0   6,9   4:03.29 chrome
  12330 bach      20   0   36,5g 158024 108380 S   1,3   2,0   0:35.89 Discord
  12751 bach      20   0 2383320 169836 108956 S   1,3   2,1   1:22.20 FoxitRe+
  15392 bach      20   0   16084   4348   3484 R   1,3   0,1   0:00.10 top
    344 root      19  -1  162808 113112 111404 S   1,0   1,4   0:21.39 systemd+
  10630 bach      20   0 1129,1g 174304  94600 S   1,0   2,2   0:47.90 chrome
   1054 root      20   0 1198944  38016  12308 S   0,7   0,5   0:59.72 warp-svc
   3354 bach      20   0 1129,2g 160936  97312 S   0,7   2,0   1:39.02 chrome
    116 root       0 -20       0      0      0 I   0,3   0,0   0:02.26 kworker+
    303 root      20   0       0      0      0 S   0,3   0,0   0:02.90 jbd2/sd+
```

3.4.1.3. Create a process

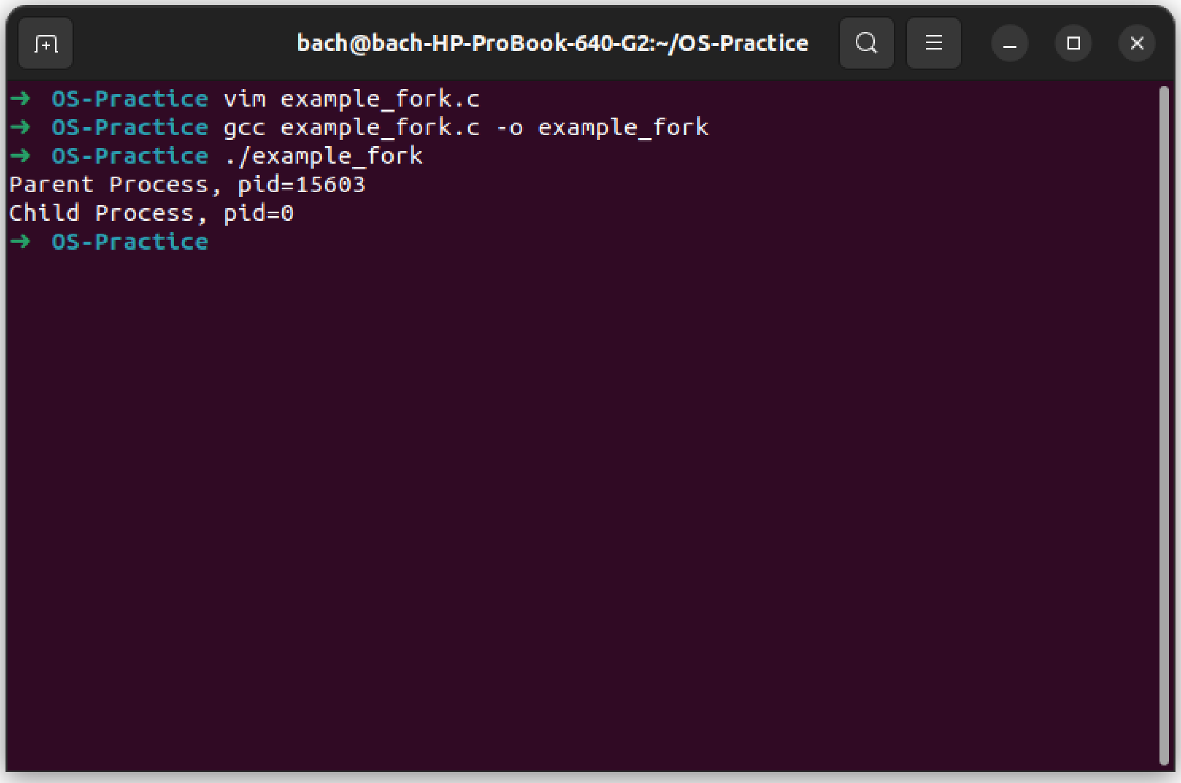
File example_fork.c



vim example_fork.c

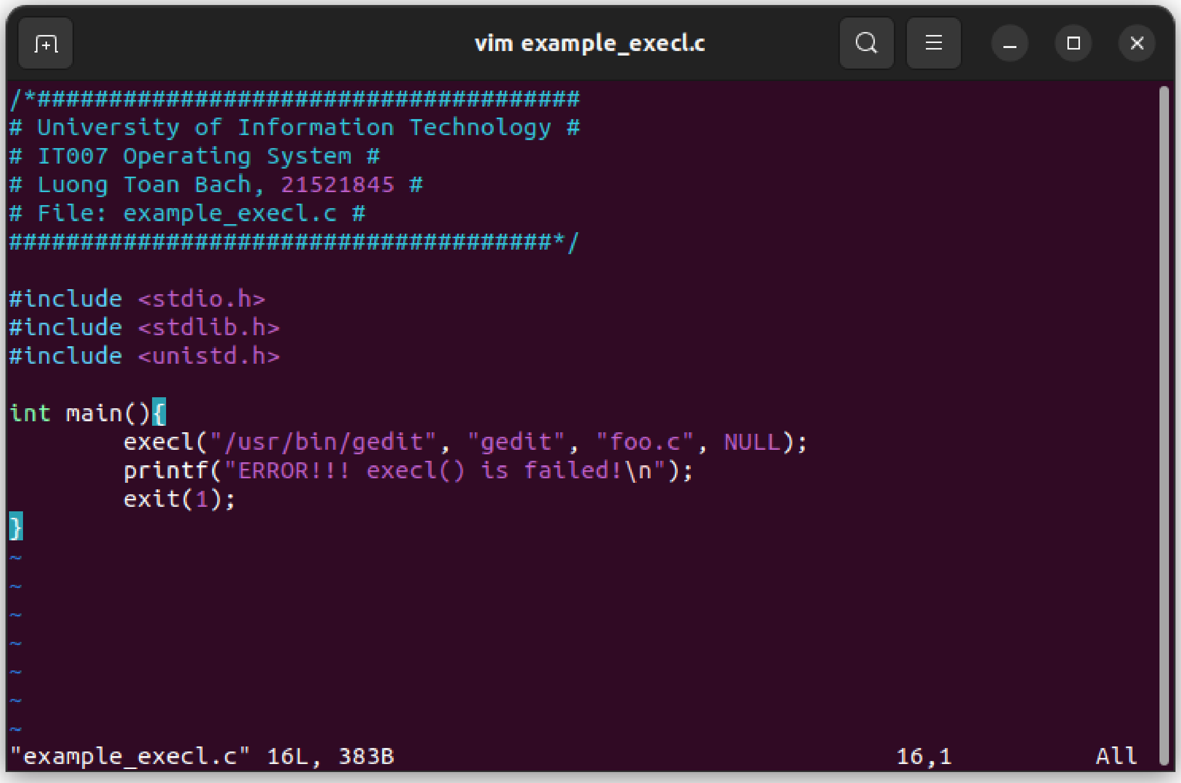
```
/*######################################
# University of Information Technology #
# IT007 Operating System #
# Luong Toan Bach, 21521845 #
# File: example_fork.c #
######################################*/

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

int main(){
        pid_t pid;
        pid=fork();
        if(pid==0)
                printf("Child Process, pid=%d\n",pid);
        else
                printf("Parent Process, pid=%d\n",pid);
        exit(0);
}
~
~
~
"example_fork.c" 20L, 416B                              20,1          All
```

Result of running example_fork.c



```
bach@bach-HP-ProBook-640-G2:~/OS-Practice
➜  OS-Practice vim example_fork.c
➜  OS-Practice gcc example_fork.c -o example_fork
➜  OS-Practice ./example_fork
Parent Process, pid=15603
Child Process, pid=0
➜  OS-Practice
```

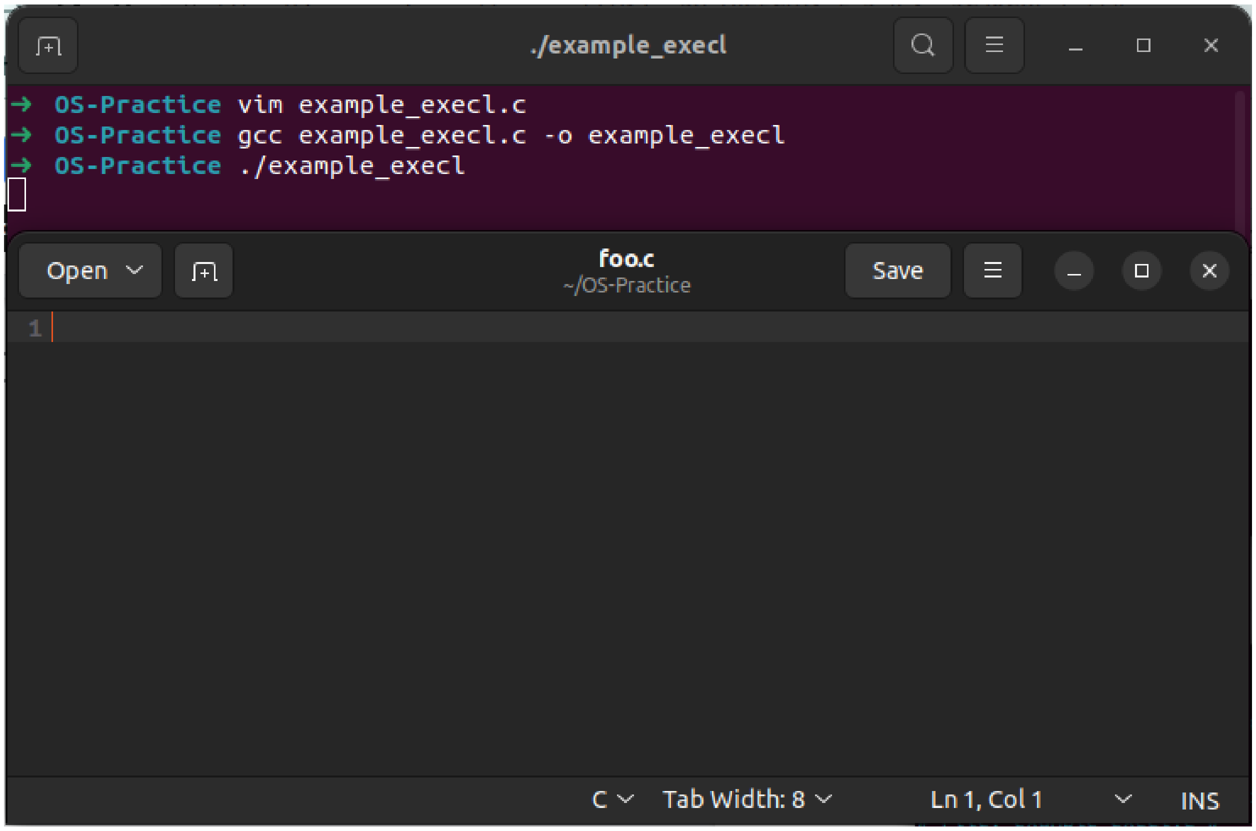
File example_fork.c



vim example_execl.c

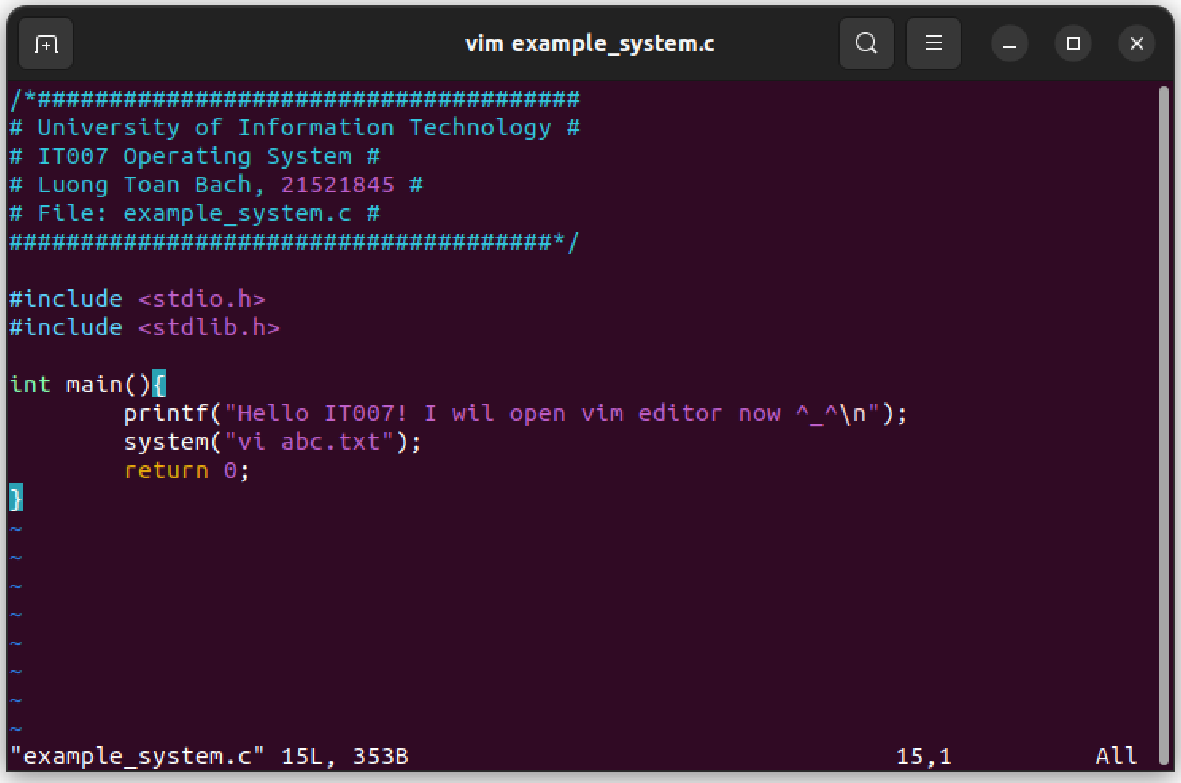
```
/*######################################
# University of Information Technology #
# IT007 Operating System #
# Luong Toan Bach, 21521845 #
# File: example_execl.c #
######################################*/

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

int main(){
        execl("/usr/bin/gedit", "gedit", "foo.c", NULL);
        printf("ERROR!!! execl() is failed!\n");
        exit(1);
}
~
~
~
~
~
~
~
"example_execl.c" 16L, 383B                               16,1          All
```

Result of running example_fork.c



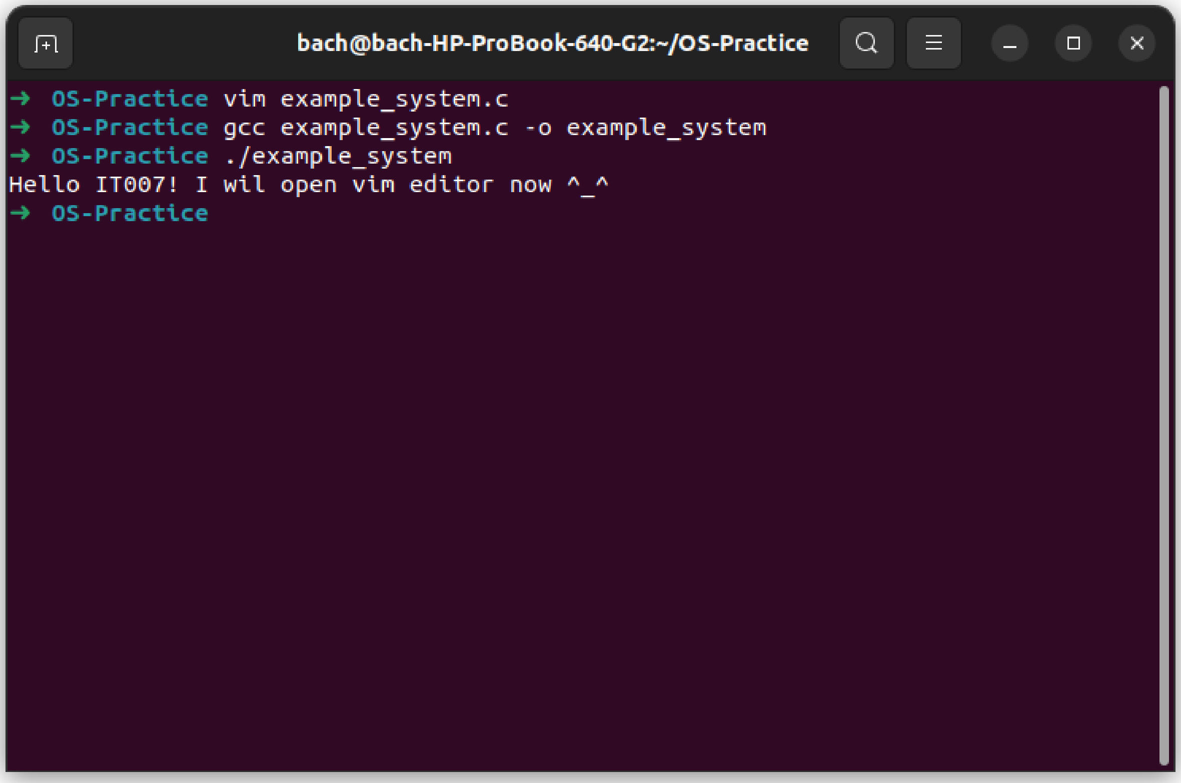
```
➜  OS-Practice vim example_execl.c
➜  OS-Practice gcc example_execl.c -o example_execl
➜  OS-Practice ./example_execl
```

File example_system.c

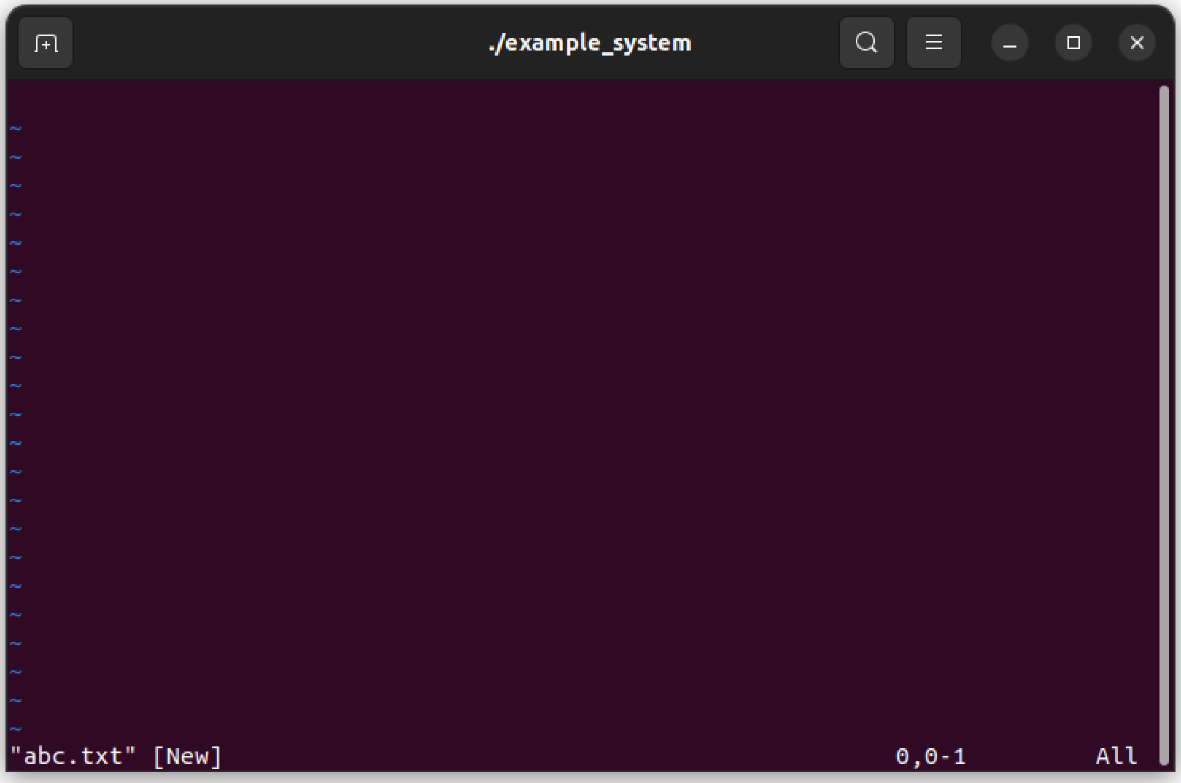


```
/*######################################
# University of Information Technology #
# IT007 Operating System #
# Luong Toan Bach, 21521845 #
# File: example_system.c #
######################################*/

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

int main(){
        printf("Hello IT007! I wil open vim editor now ^_^\n");
        system("vi abc.txt");
        return 0;
}
~
~
~
~
~
~
~
~
"example_system.c" 15L, 353B                              15,1          All
```

Result of running example_system.c



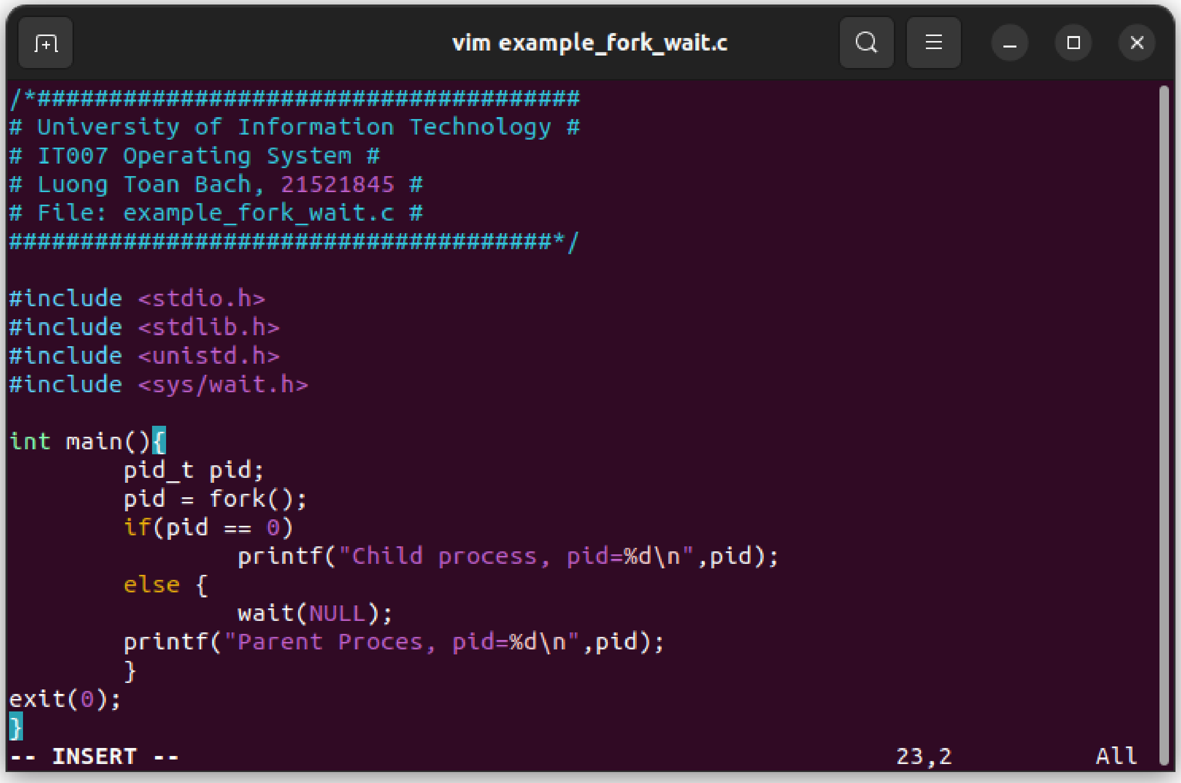
```
➜ OS-Practice vim example_system.c
➜ OS-Practice gcc example_system.c -o example_system
➜ OS-Practice ./example_system
Hello IT007! I wil open vim editor now ^_^
➜ OS-Practice
```



```
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
"abc.txt" [New]                                           0,0-1         All
```

3.4.1.4. Finish a process

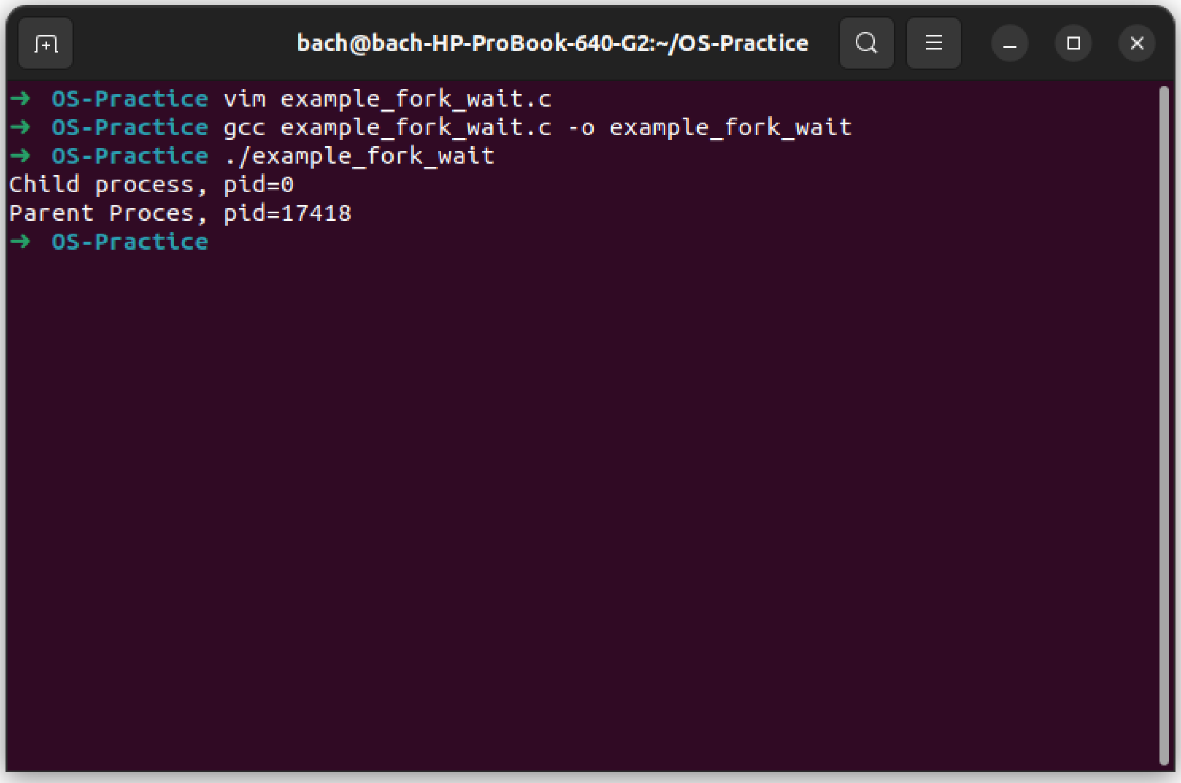
File example_fork_wait.c



vim example_fork_wait.c

```
/*#######################################
# University of Information Technology #
# IT007 Operating System #
# Luong Toan Bach, 21521845 #
# File: example_fork_wait.c #
######################################*/

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

int main(){
        pid_t pid;
        pid = fork();
        if(pid == 0)
                printf("Child process, pid=%d\n",pid);
        else {
                wait(NULL);
        printf("Parent Proces, pid=%d\n",pid);
        }
exit(0);
}
-- INSERT --                                            23,2          All
```

Result of running example_fork_wait.c



```
➜  OS-Practice vim example_fork_wait.c
➜  OS-Practice gcc example_fork_wait.c -o example_fork_wait
➜  OS-Practice ./example_fork_wait
Child process, pid=0
Parent Proces, pid=17418
➜  OS-Practice
```

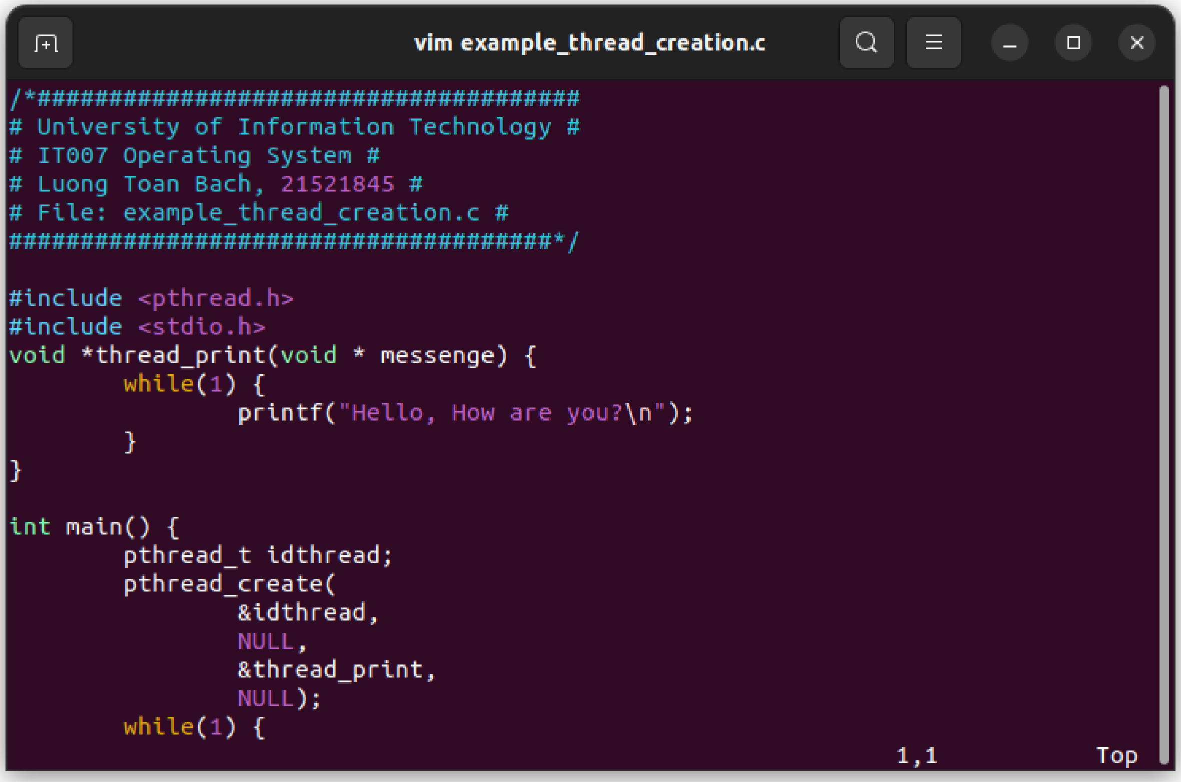
3.4.2. Sub-process

3.4.2.1. Concept of sub-process

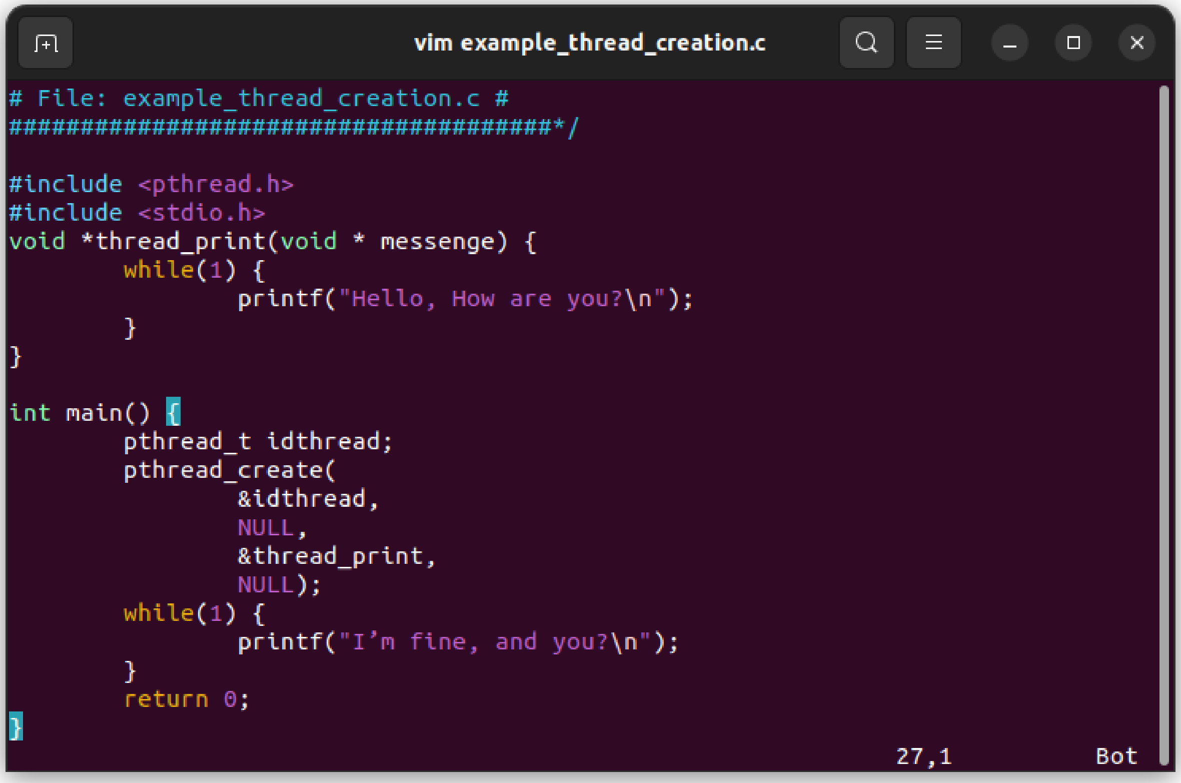
3.4.2.2. Sub-process in Linux

3.4.2.3. Create a sub-process

File example_thread_creation.c

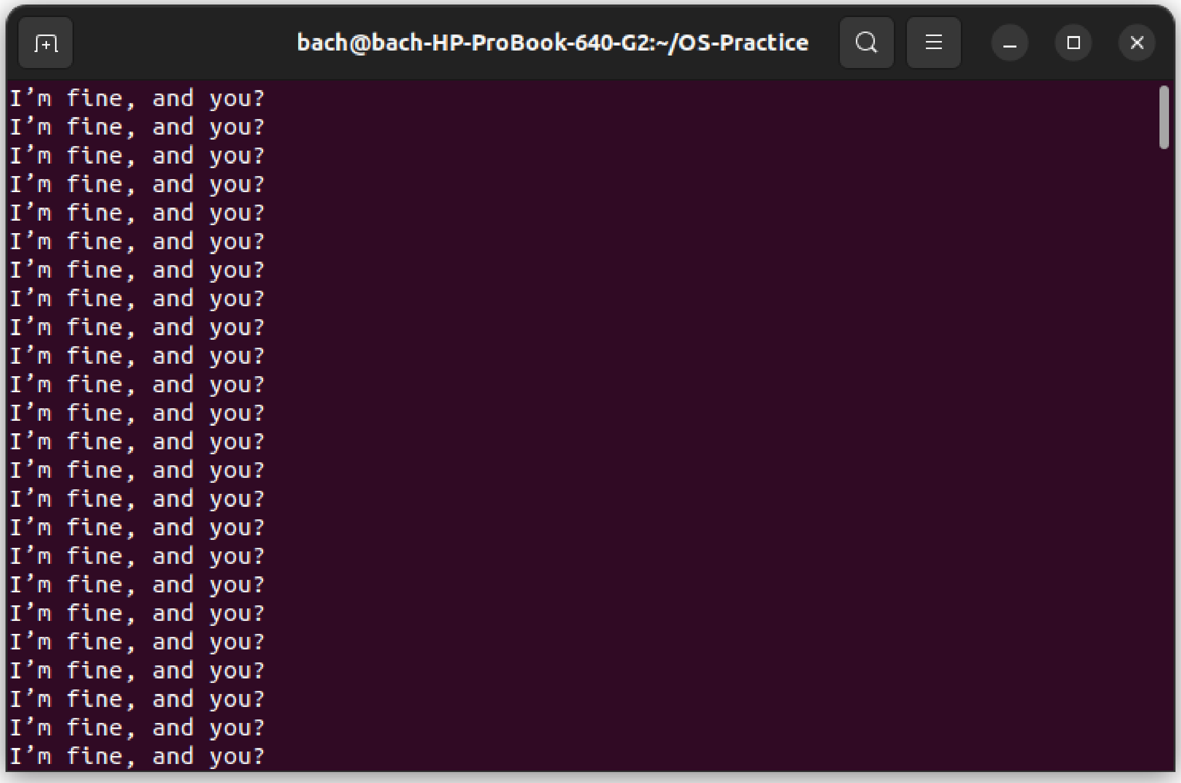


```
/*#######################################
# University of Information Technology #
# IT007 Operating System #
# Luong Toan Bach, 21521845 #
# File: example_thread_creation.c #
######################################*/

#include <pthread.h>
#include <stdio.h>
void *thread_print(void * messenge) {
        while(1) {
                printf("Hello, How are you?\n");
        }
}

int main() {
        pthread_t idthread;
        pthread_create(
                &idthread,
                NULL,
                &thread_print,
                NULL);
        while(1) {
                                                                 1,1           Top
```



```
# File: example_thread_creation.c #
######################################*/

#include <pthread.h>
#include <stdio.h>
void *thread_print(void * messenge) {
        while(1) {
                printf("Hello, How are you?\n");
        }
}

int main() {
        pthread_t idthread;
        pthread_create(
                &idthread,
                NULL,
                &thread_print,
                NULL);
        while(1) {
                printf("I'm fine, and you?\n");
        }
        return 0;
}
                                                                 27,1          Bot
```

Result of running example_thread_creation.c

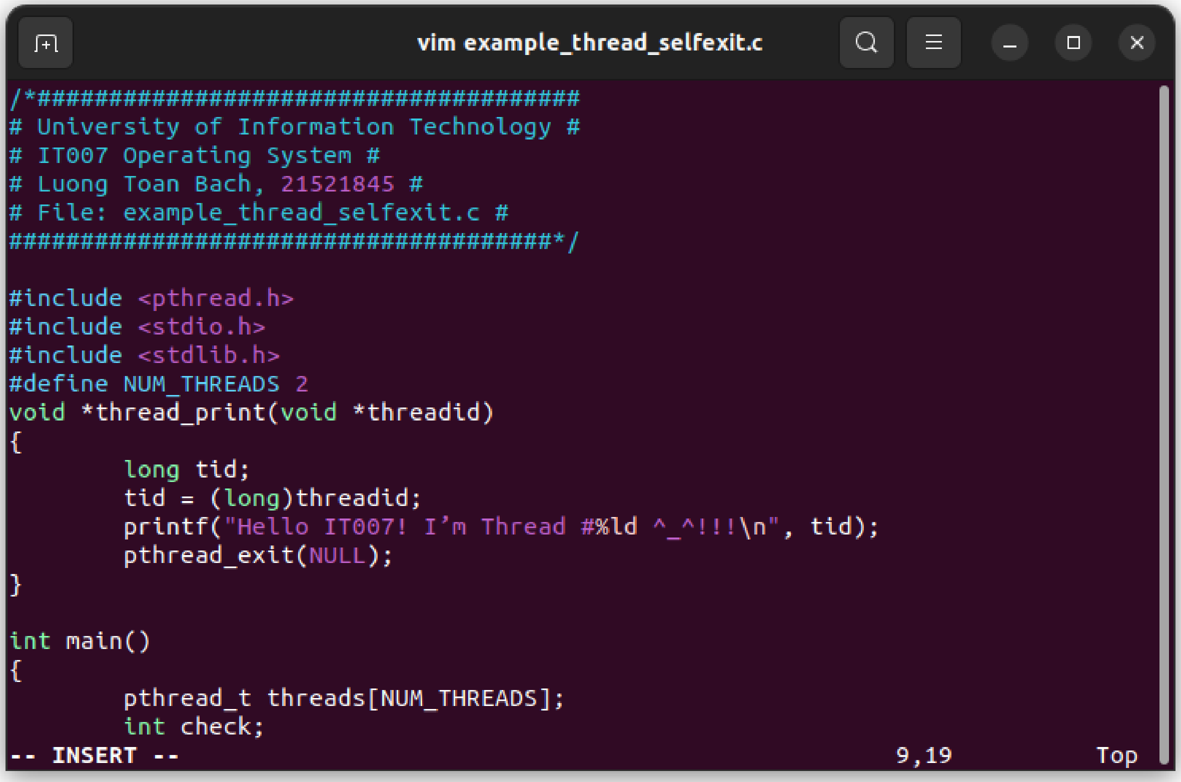


bach@bach-HP-ProBook-640-G2:~/OS-Practice

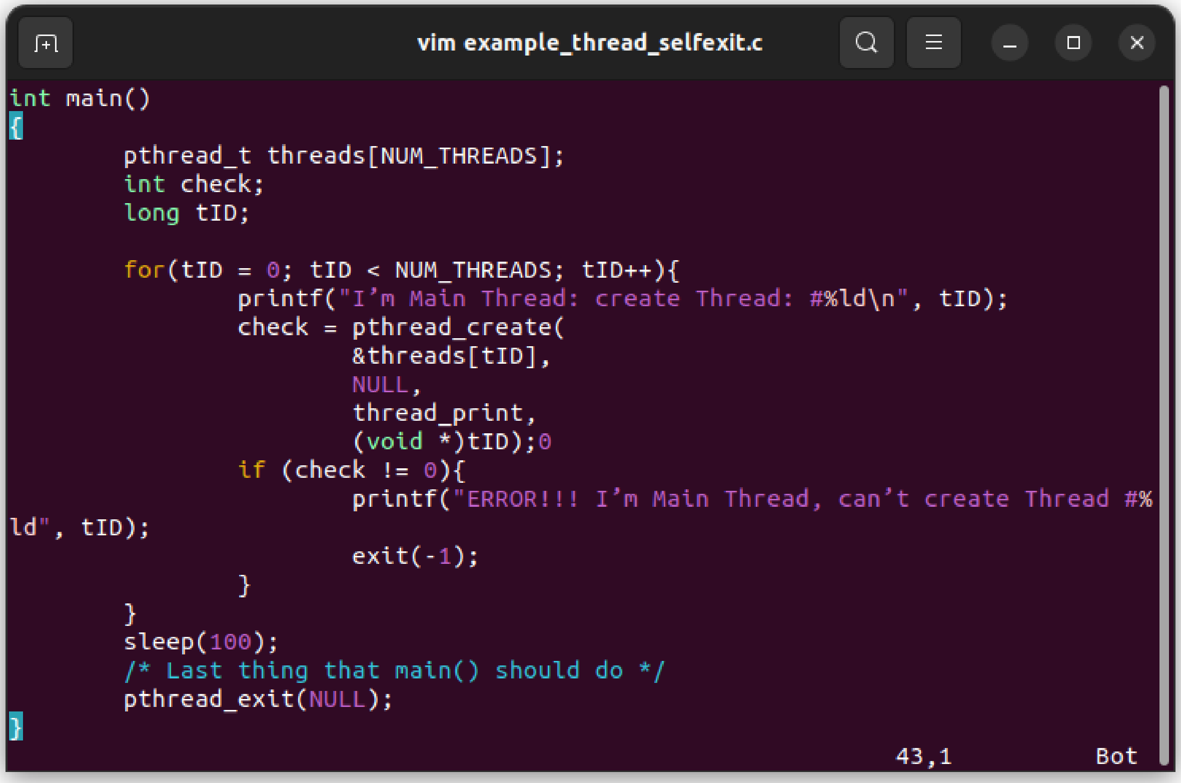
```
I'm fine, and you?
I'm fine, and you?
I'm fine, and you?
I'm fine, and you?
I'm fine, and you?
I'm fine, and you?
I'm fine, and you?
I'm fine, and you?
I'm fine, and you?
I'm fine, and you?
I'm fine, and you?
I'm fine, and you?
I'm fine, and you?
I'm fine, and you?
I'm fine, and you?
I'm fine, and you?
I'm fine, and you?
I'm fine, and you?
I'm fine, and you?
I'm fine, and you?
I'm fine, and you?
I'm fine, and you?
I'm fine, and you?
I'm fine, and you?
```

3.4.2.4. Finish a sub-process

File example_thread_selfexit.c

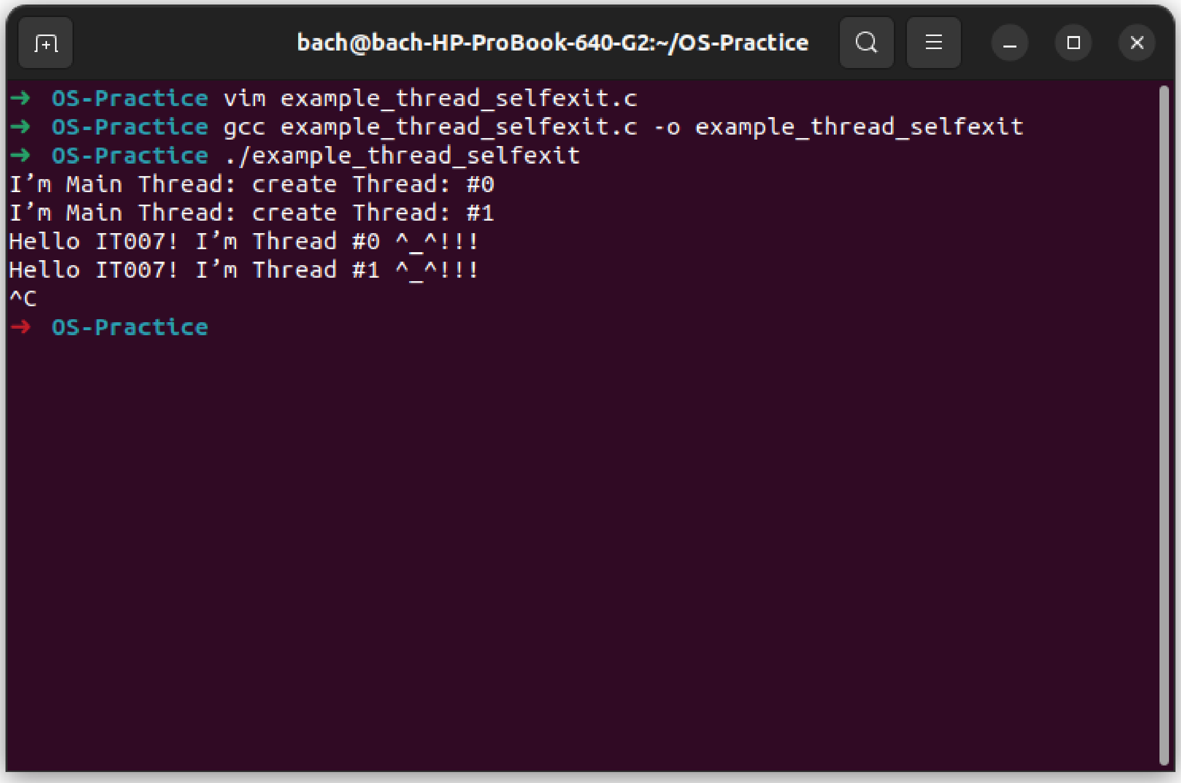


```
/*########################################
# University of Information Technology #
# IT007 Operating System #
# Luong Toan Bach, 21521845 #
# File: example_thread_selfexit.c #
########################################*/

#include <pthread.h>
#include <stdio.h>
#include <stdlib.h>
#define NUM_THREADS 2
void *thread_print(void *threadid)
{
        long tid;
        tid = (long)threadid;
        printf("Hello IT007! I’m Thread #%ld ^_^!!!\n", tid);
        pthread_exit(NULL);
}

int main()
{
        pthread_t threads[NUM_THREADS];
        int check;
-- INSERT --                                                   9,19          Top
```



```
int main()
{
        pthread_t threads[NUM_THREADS];
        int check;
        long tID;

        for(tID = 0; tID < NUM_THREADS; tID++){
                printf("I’m Main Thread: create Thread: #%ld\n", tID);
                check = pthread_create(
                        &threads[tID],
                        NULL,
                        thread_print,
                        (void *)tID);0
                if (check != 0){
                        printf("ERROR!!! I’m Main Thread, can’t create Thread #%
ld", tID);
                        exit(-1);
                }
        }
        sleep(100);
        /* Last thing that main() should do */
        pthread_exit(NULL);
}
                                                               43,1          Bot
```

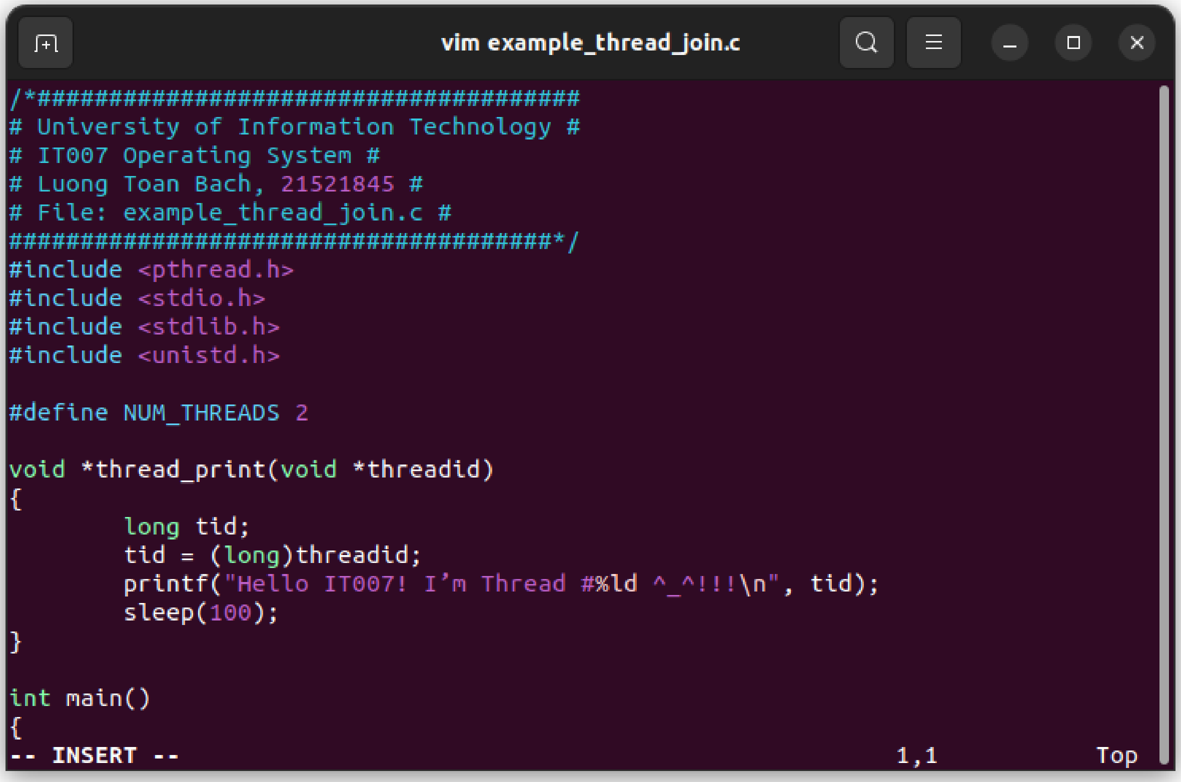
Result of running example_thread_selfexit.c



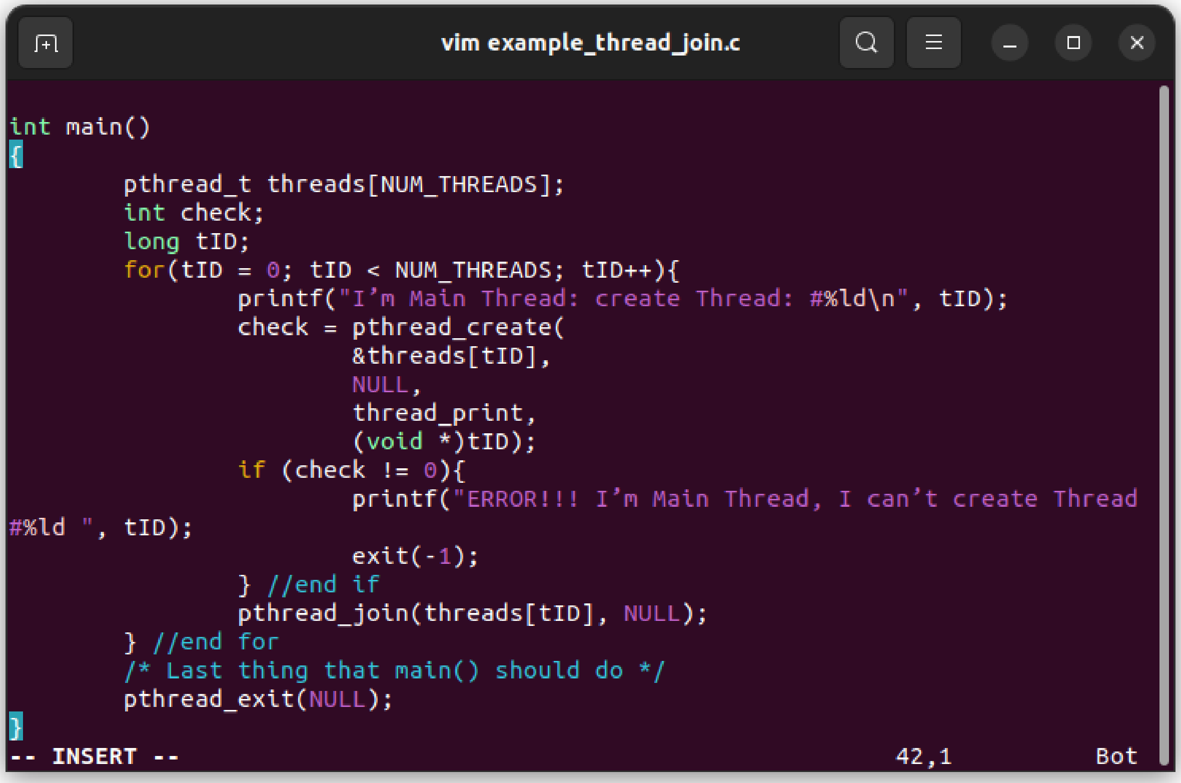
```
➜  OS-Practice vim example_thread_selfexit.c
➜  OS-Practice gcc example_thread_selfexit.c -o example_thread_selfexit
➜  OS-Practice ./example_thread_selfexit
I’m Main Thread: create Thread: #0
I’m Main Thread: create Thread: #1
Hello IT007! I’m Thread #0 ^_^!!!
Hello IT007! I’m Thread #1 ^_^!!!
^C
➜  OS-Practice
```

3.4.2.5. Join and Block sub-process

File example_thread_join.c

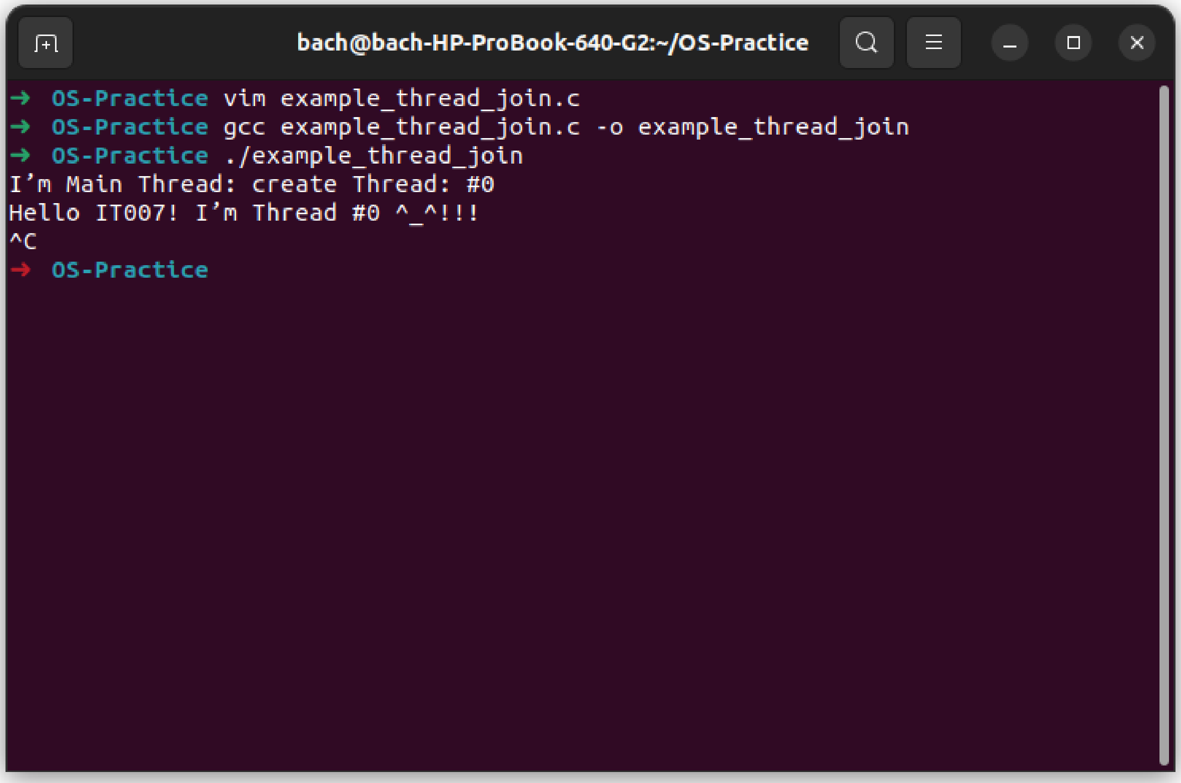


```
/*########################################
# University of Information Technology #
# IT007 Operating System #
# Luong Toan Bach, 21521845 #
# File: example_thread_join.c #
########################################*/
#include <pthread.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

#define NUM_THREADS 2

void *thread_print(void *threadid)
{
        long tid;
        tid = (long)threadid;
        printf("Hello IT007! I'm Thread #%ld ^_^!!!\n", tid);
        sleep(100);
}

int main()
{
-- INSERT --                                                  1,1           Top
```



```
int main()
{
        pthread_t threads[NUM_THREADS];
        int check;
        long tID;
        for(tID = 0; tID < NUM_THREADS; tID++){
                printf("I'm Main Thread: create Thread: #%ld\n", tID);
                check = pthread_create(
                        &threads[tID],
                        NULL,
                        thread_print,
                        (void *)tID);
                if (check != 0){
                        printf("ERROR!!! I'm Main Thread, I can't create Thread
#%ld ", tID);
                        exit(-1);
                } //end if
                pthread_join(threads[tID], NULL);
        } //end for
        /* Last thing that main() should do */
        pthread_exit(NULL);
}
-- INSERT --                                                  42,1          Bot
```

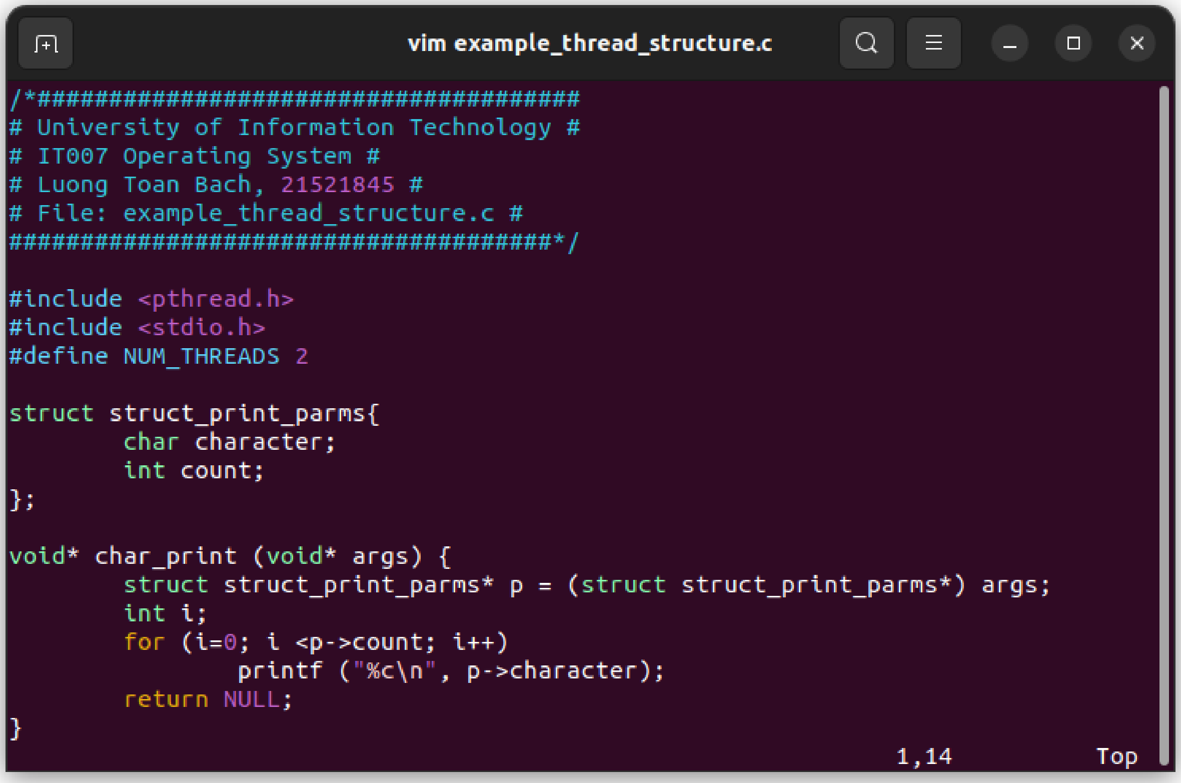
Result of running example_thread_join.c



```
bach@bach-HP-ProBook-640-G2:~/OS-Practice
➜  OS-Practice vim example_thread_join.c
➜  OS-Practice gcc example_thread_join.c -o example_thread_join
➜  OS-Practice ./example_thread_join
I'm Main Thread: create Thread: #0
Hello IT007! I'm Thread #0 ^_^!!!
^C
➜  OS-Practice
```

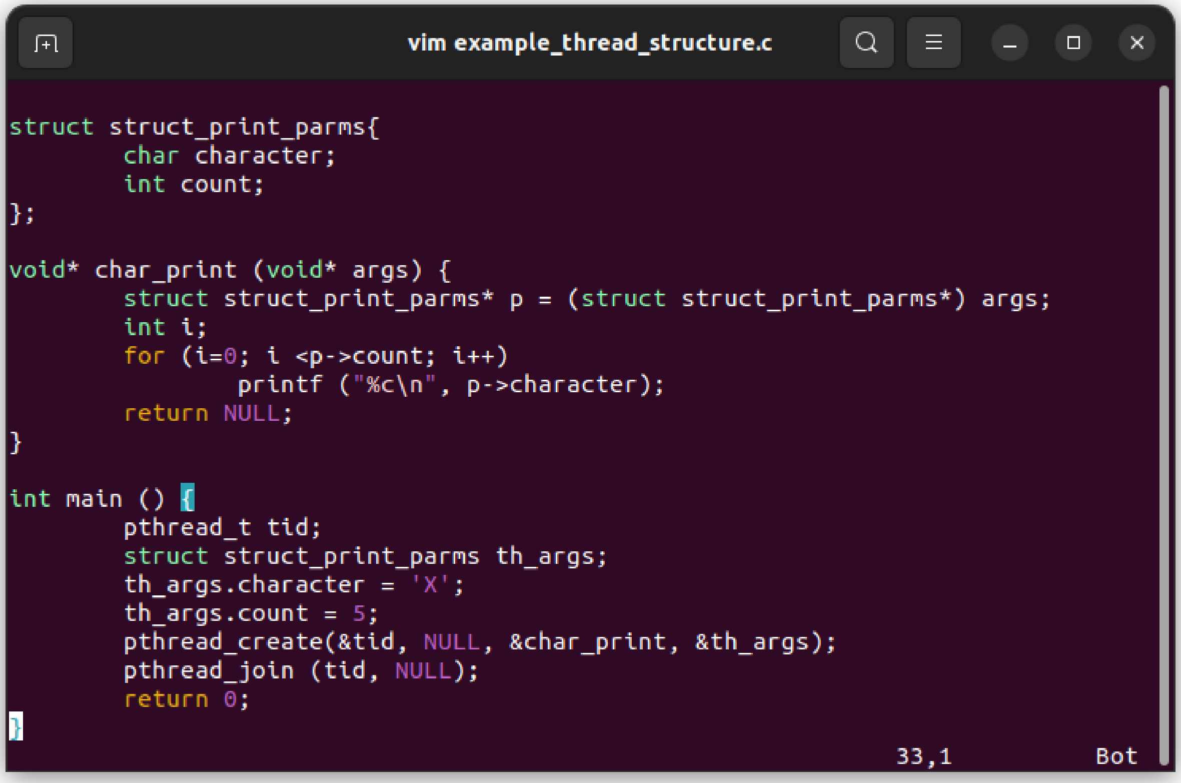
3.4.2.6. Add data to sub-process

File example_thread_structure.c



vim example_thread_structure.c

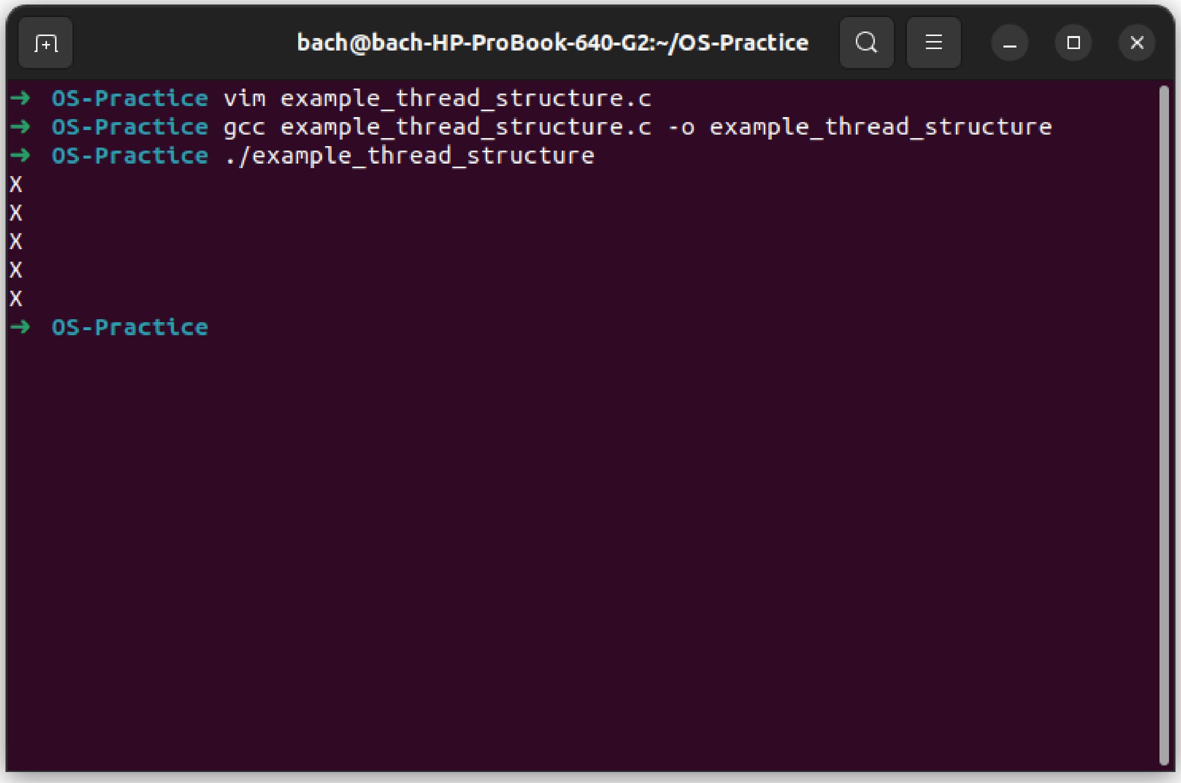
```
/*#######################################
# University of Information Technology #
# IT007 Operating System #
# Luong Toan Bach, 21521845 #
# File: example_thread_structure.c #
######################################*/

#include <pthread.h>
#include <stdio.h>
#define NUM_THREADS 2

struct struct_print_parms{
        char character;
        int count;
};

void* char_print (void* args) {
        struct struct_print_parms* p = (struct struct_print_parms*) args;
        int i;
        for (i=0; i <p->count; i++)
                printf ("%c\n", p->character);
        return NULL;
}
                                                              1,14          Top
```



vim example_thread_structure.c

```

struct struct_print_parms{
        char character;
        int count;
};

void* char_print (void* args) {
        struct struct_print_parms* p = (struct struct_print_parms*) args;
        int i;
        for (i=0; i <p->count; i++)
                printf ("%c\n", p->character);
        return NULL;
}

int main () {
        pthread_t tid;
        struct struct_print_parms th_args;
        th_args.character = 'X';
        th_args.count = 5;
        pthread_create(&tid, NULL, &char_print, &th_args);
        pthread_join (tid, NULL);
        return 0;
}
                                                              33,1          Bot
```

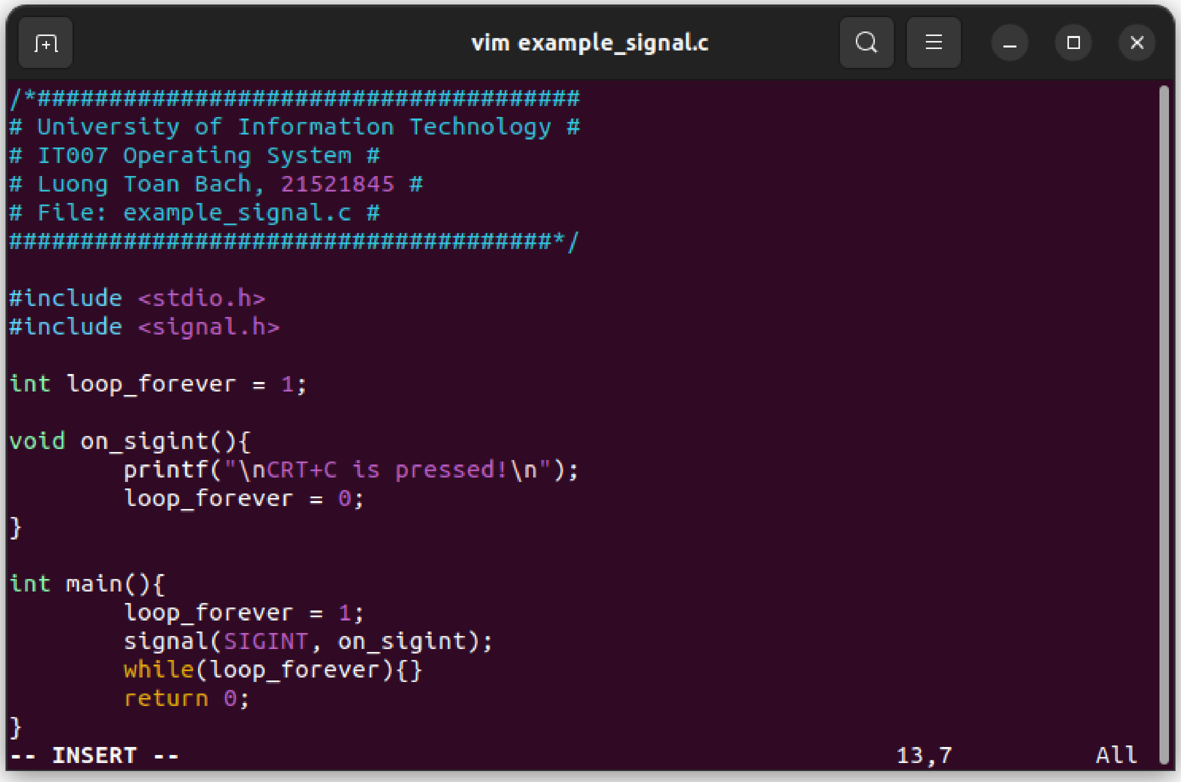
Result of running example_thread_structure.c



```
➜  OS-Practice vim example_thread_structure.c
➜  OS-Practice gcc example_thread_structure.c -o example_thread_structure
➜  OS-Practice ./example_thread_structure
X
X
X
X
X
➜  OS-Practice
```

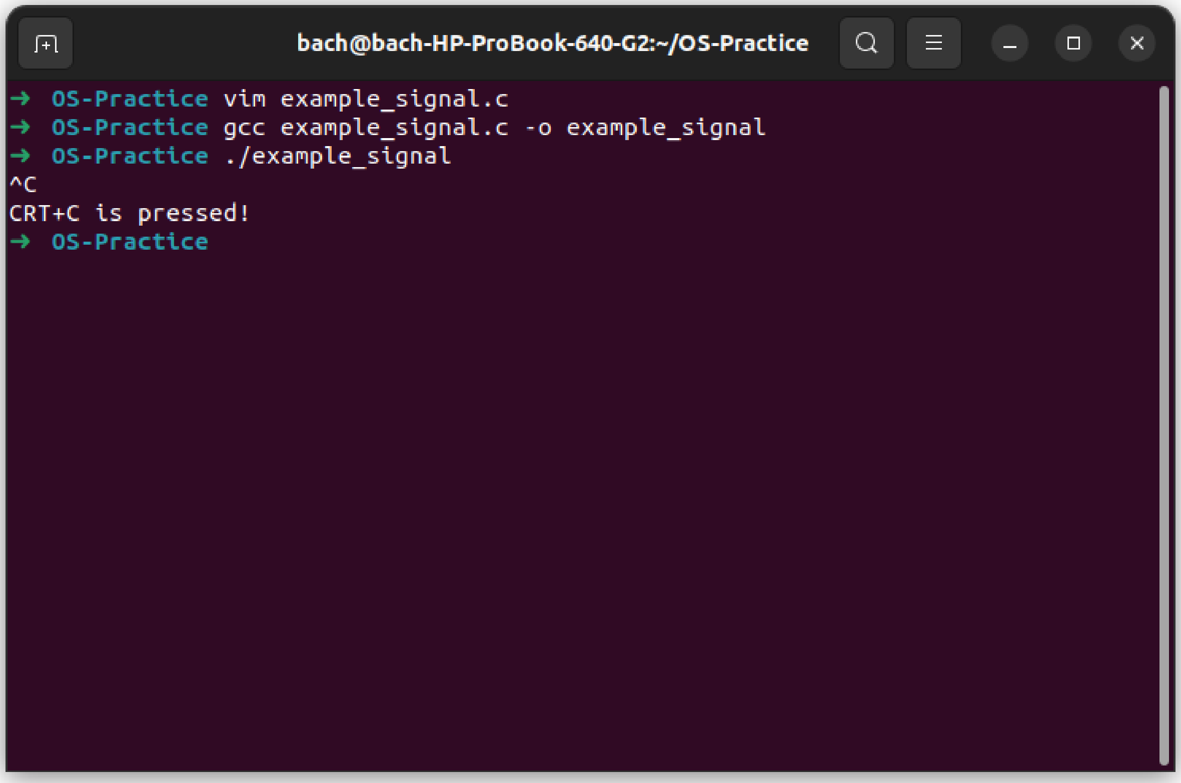
3.4.3. Signal

File example_signal.c


vim example_signal.c

```
/*######################################
# University of Information Technology #
# IT007 Operating System #
# Luong Toan Bach, 21521845 #
# File: example_signal.c #
######################################*/

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

int loop_forever = 1;

void on_sigint(){
        printf("\nCRT+C is pressed!\n");
        loop_forever = 0;
}

int main(){
        loop_forever = 1;
        signal(SIGINT, on_sigint);
        while(loop_forever){}
        return 0;
}
-- INSERT --                                              13,7          All
```

Result of running example_signal.c

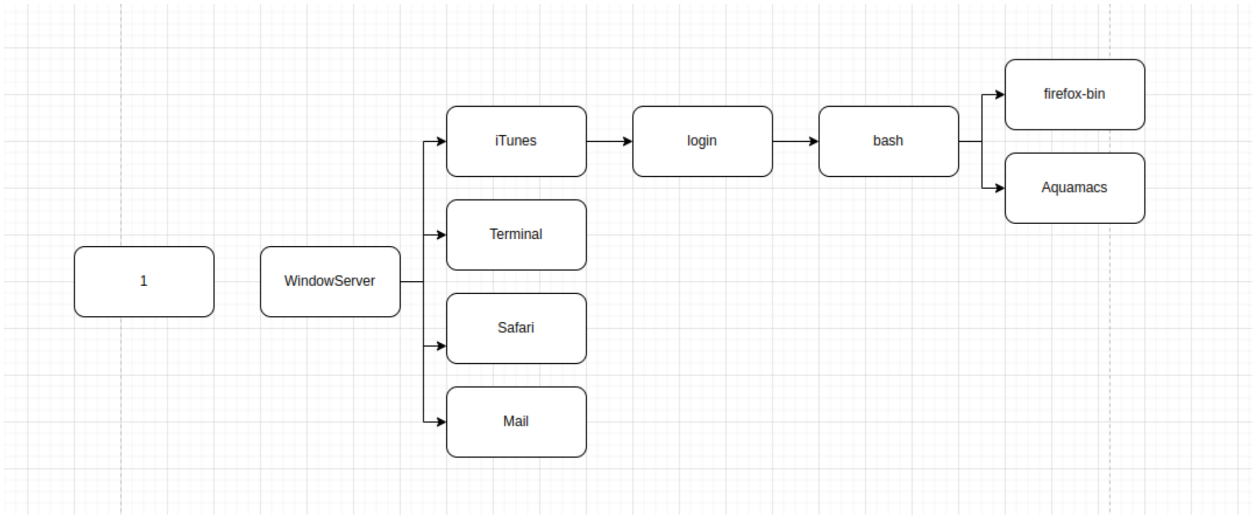


```
➜ OS-Practice vim example_signal.c
➜ OS-Practice gcc example_signal.c -o example_signal
➜ OS-Practice ./example_signal
^C
CRT+C is pressed!
➜ OS-Practice
```

Section 3.5 HomeWork

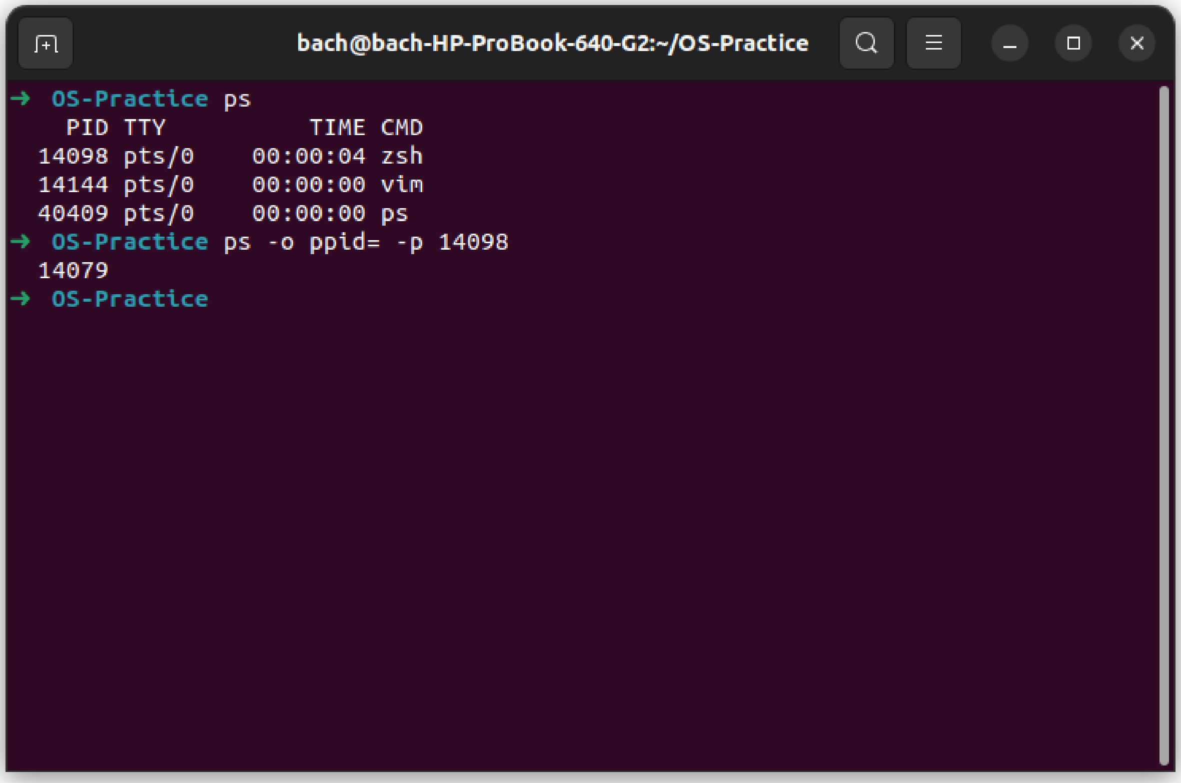
3.5.1.

a.



$$\begin{tikzcd}
 & & & & & \text{firefox-bin} \\
 & & \text{iTunes} \arrow[r] & \text{login} \arrow[r] & \text{bash} \arrow[ur] \arrow[dr] & \\
 & & \text{Terminal} & & & \text{Aquamacs} \\
1 & \text{WindowServer} \arrow[uur] \arrow[ur] \arrow[r] \arrow[dr] & \text{Safari} & & & \\
 & & \text{Mail} & & &
\end{tikzcd}$$

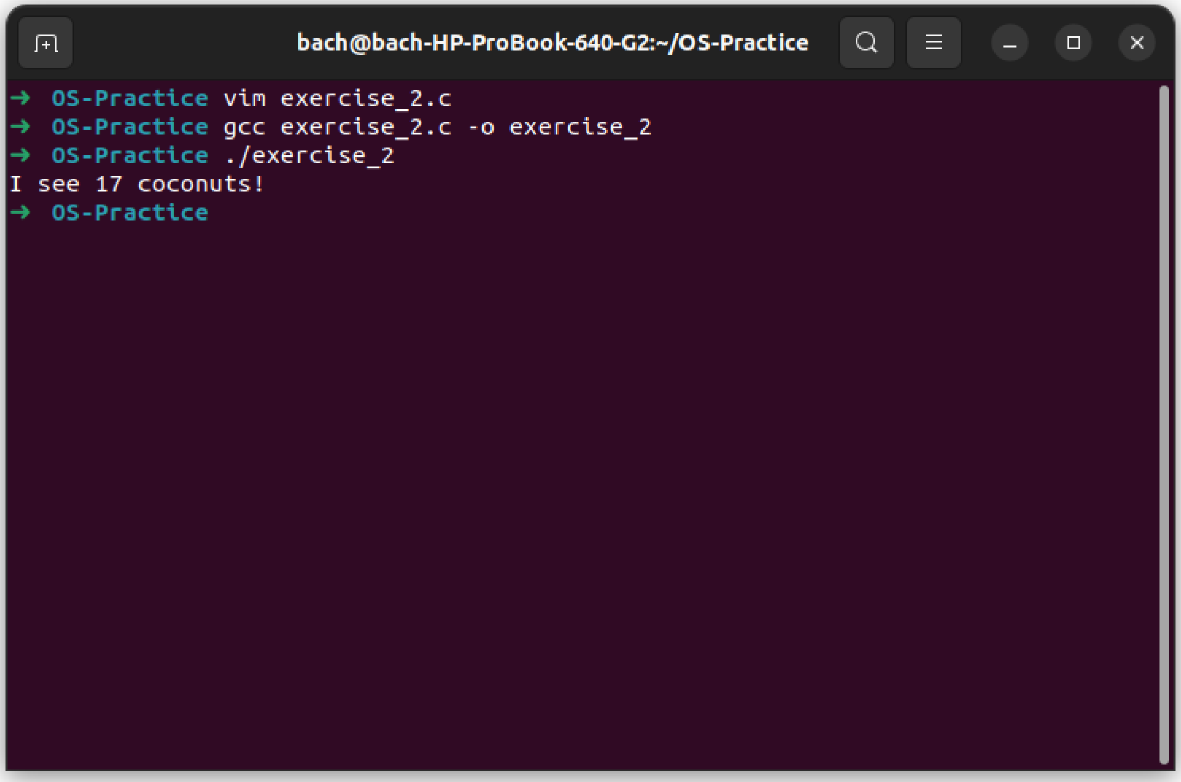
b.



```
bach@bach-HP-ProBook-640-G2:~/OS-Practice
→ OS-Practice ps
    PID TTY          TIME CMD
  14098 pts/0    00:00:04 zsh
  14144 pts/0    00:00:00 vim
  40409 pts/0    00:00:00 ps
→ OS-Practice ps -o ppid= -p 14098
  14079
→ OS-Practice
```

c.

3.5.2.



```
➜  OS-Practice vim exercise_2.c
➜  OS-Practice gcc exercise_2.c -o exercise_2
➜  OS-Practice ./exercise_2
I see 17 coconuts!
➜  OS-Practice
```

Because in sub-process that have the pid == 0(child process) so it had terminated. The parent possess(pid !=0) went to the else statement and change the value of num_coconuts.

3.5.3.

- Properties of sub-process:

Properties	Default value	Meaning
Guradsize	PAGEIZES	The size ensures that the thread does not use more than the allocated space
Scope	PTHREAD_SCOPE_PROCE SS	Use resources within the scope of the process
Detachstate	PTHREAD_CREATE_JOIN ABLE	Threads are merged with other processes
Stackaddr	NULL	New thread has address

		in system-allocated stack
SatckSize	NULL	The next thread will have the size specified by the system
Inheritsched	PTHREAD_INHERIT_SCHED	The child thread will inherit the parent thread's priority schedule
SchedPolicy	SCHED_OTHER	The thread will run according to the thread's priority

Set up properties for sub-process

+ Using command ‘attr’ with ‘pthread_attr_t*’

+ Using command pthread_attr_init(&attr) to reset default value.

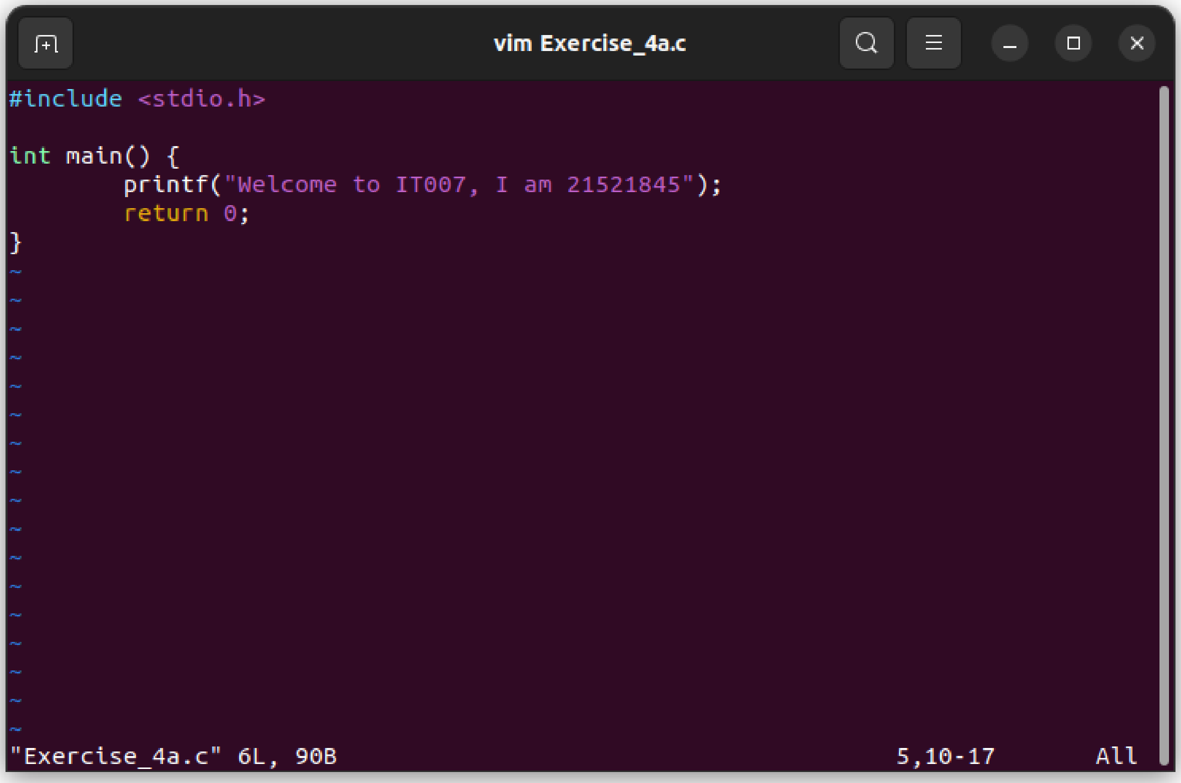
+ Call properties function :

pthread_attr_set(detachstate/Inheritsched/SchedPolicy/scope/....)()

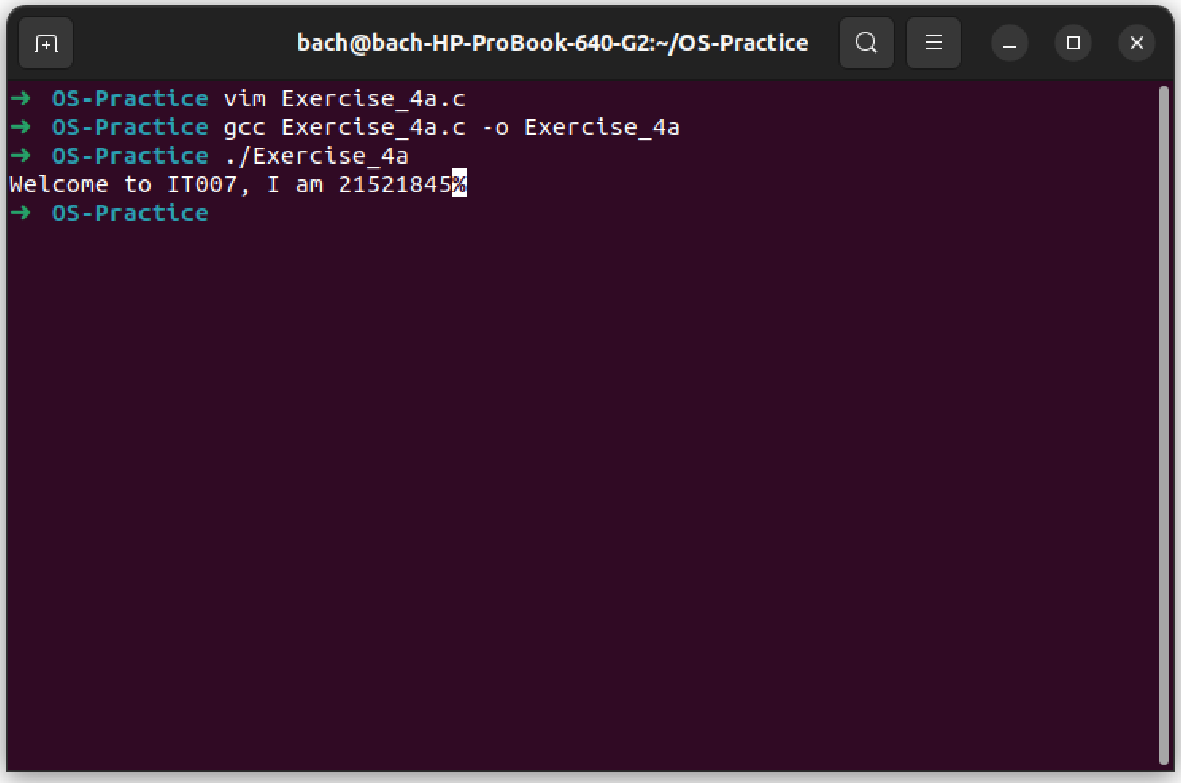
+ Using pthread_attr_destroy(): to destroy properties that not nercesary.

3.5.4.

a.

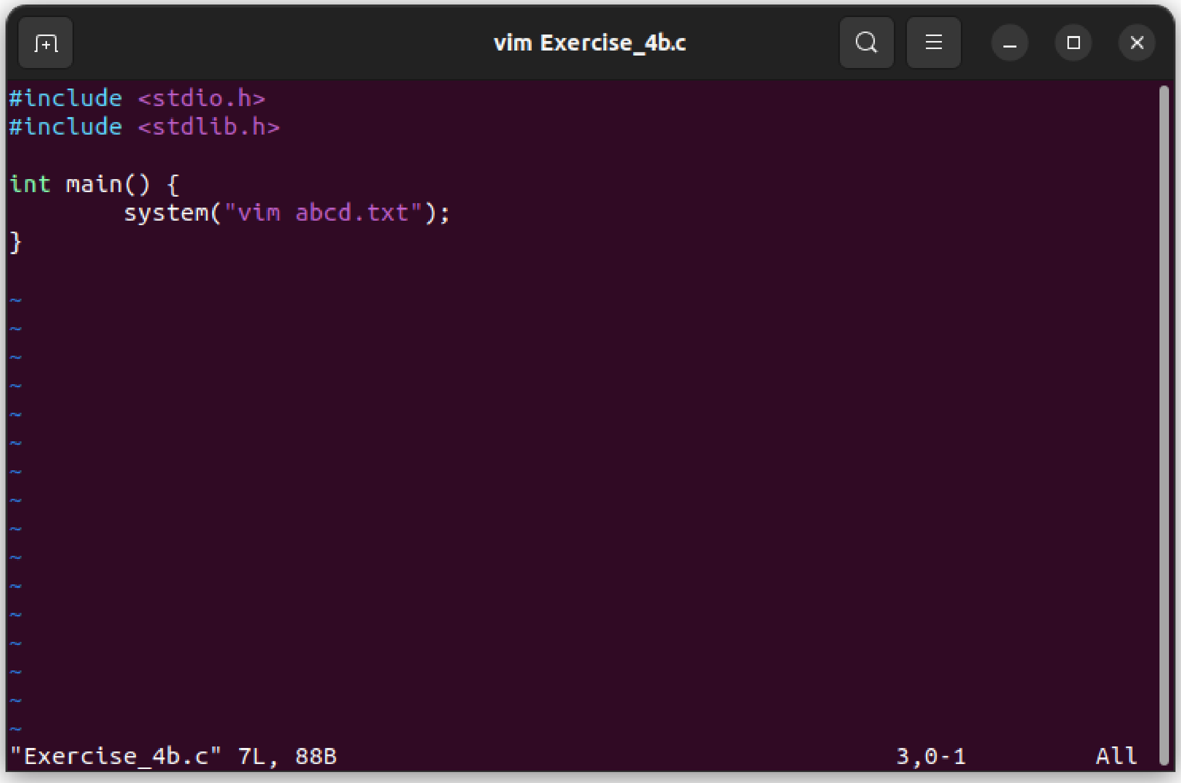


```
#include <stdio.h>

int main() {
        printf("Welcome to IT007, I am 21521845");
        return 0;
}
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
"Exercise_4a.c" 6L, 90B                                 5,10-17       All
```

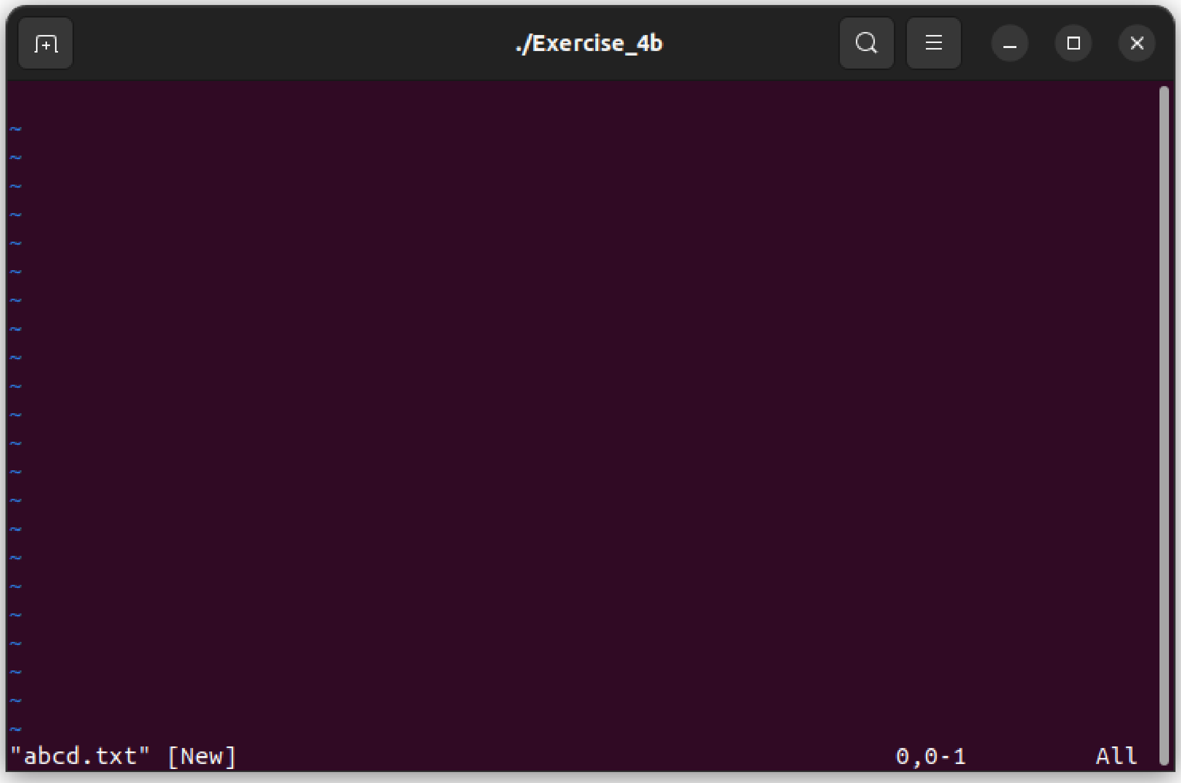


```
➜  OS-Practice vim Exercise_4a.c
➜  OS-Practice gcc Exercise_4a.c -o Exercise_4a
➜  OS-Practice ./Exercise_4a
Welcome to IT007, I am 21521845%
➜  OS-Practice
```

b.

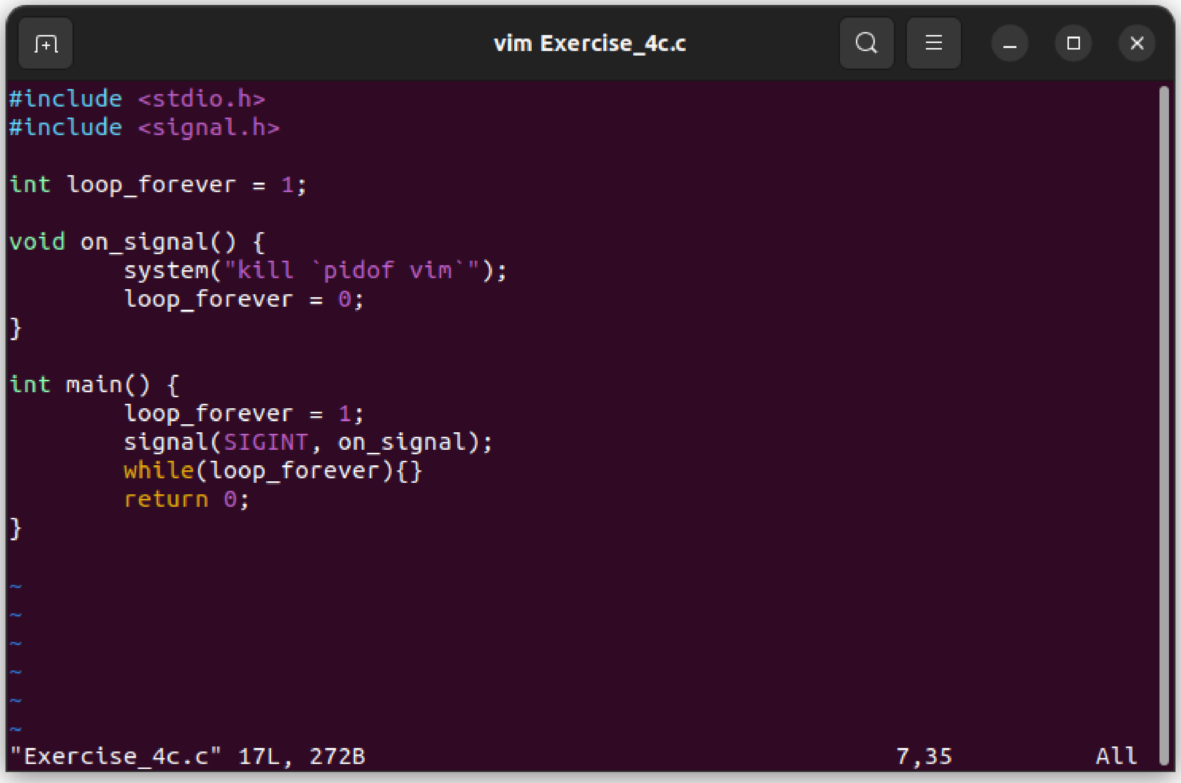


```
#include <stdio.h>
#include <stdlib.h>

int main() {
        system("vim abcd.txt");
}

~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
"Exercise_4b.c" 7L, 88B                                     3,0-1         All
```

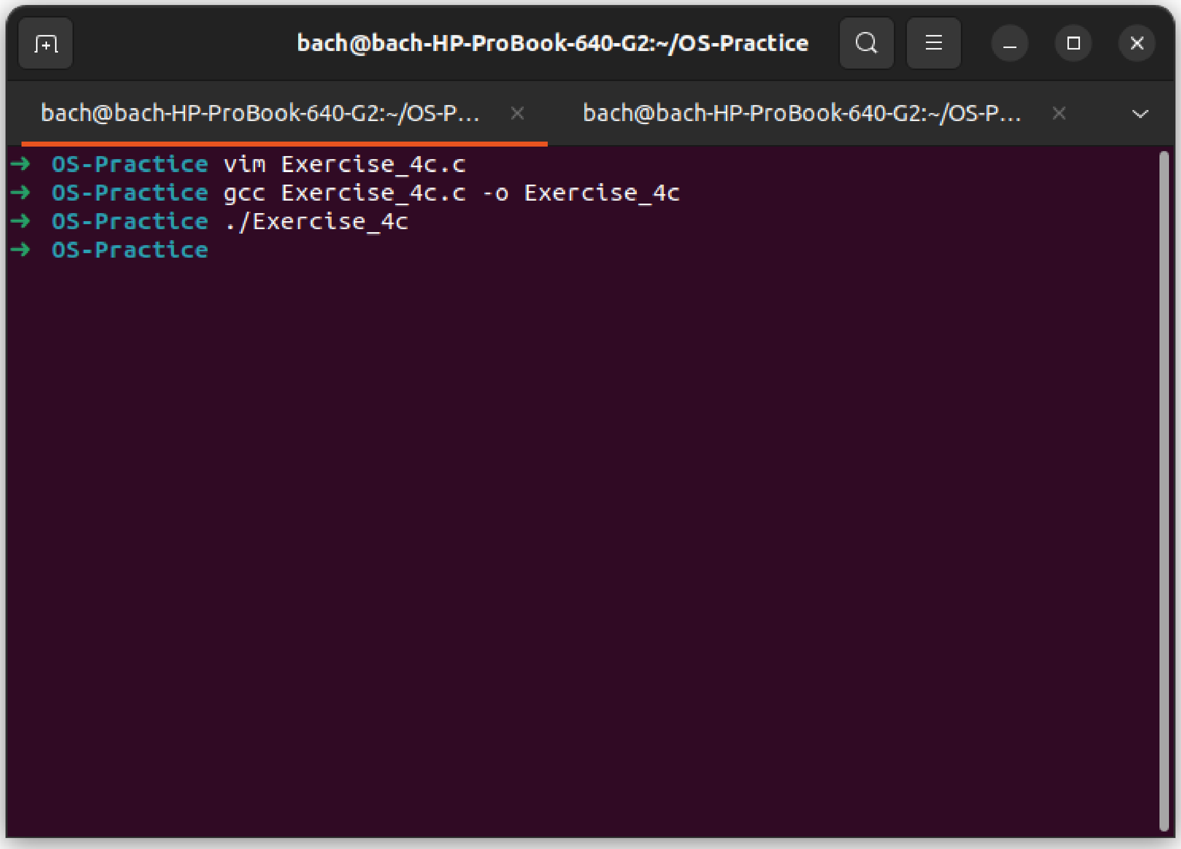


```
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
"abcd.txt" [New]                                              0,0-1         All
```

c.

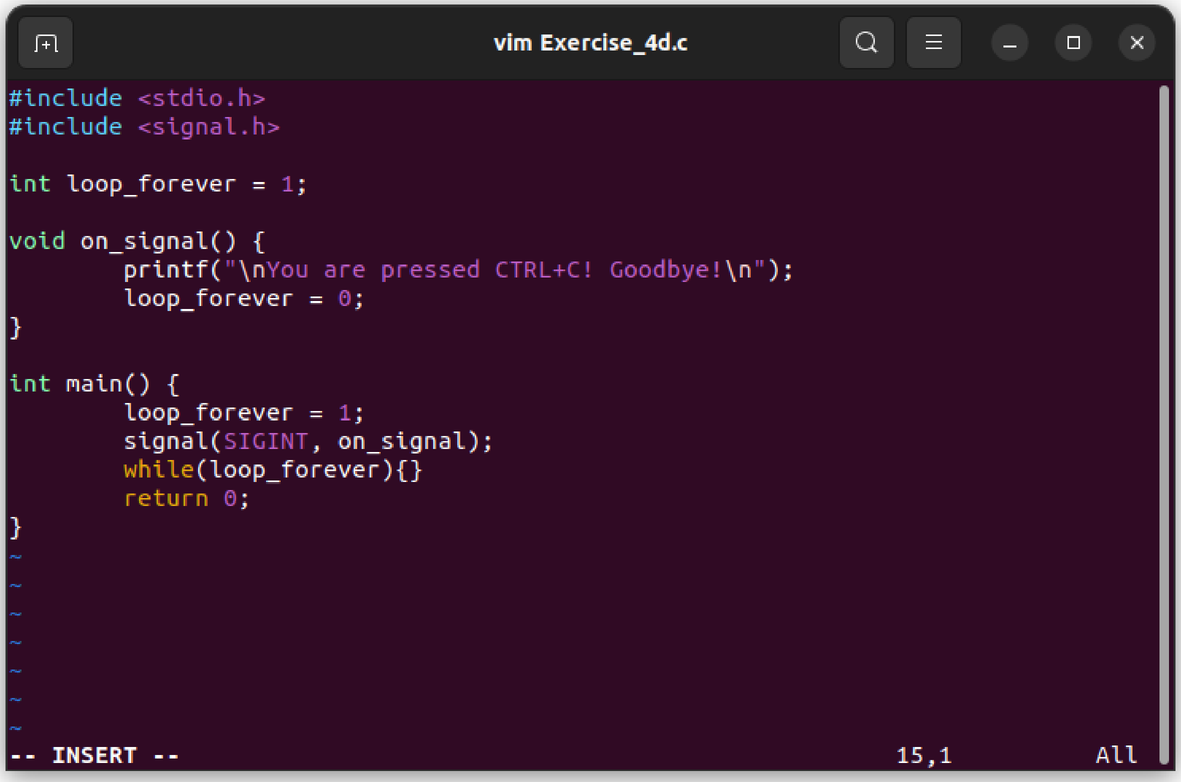


```
#include <stdio.h>
#include <signal.h>

int loop_forever = 1;

void on_signal() {
        system("kill `pidof vim`");
        loop_forever = 0;
}

int main() {
        loop_forever = 1;
        signal(SIGINT, on_signal);
        while(loop_forever){}
        return 0;
}

~
~
~
~
~
~
"Exercise_4c.c" 17L, 272B                            7,35          All
```

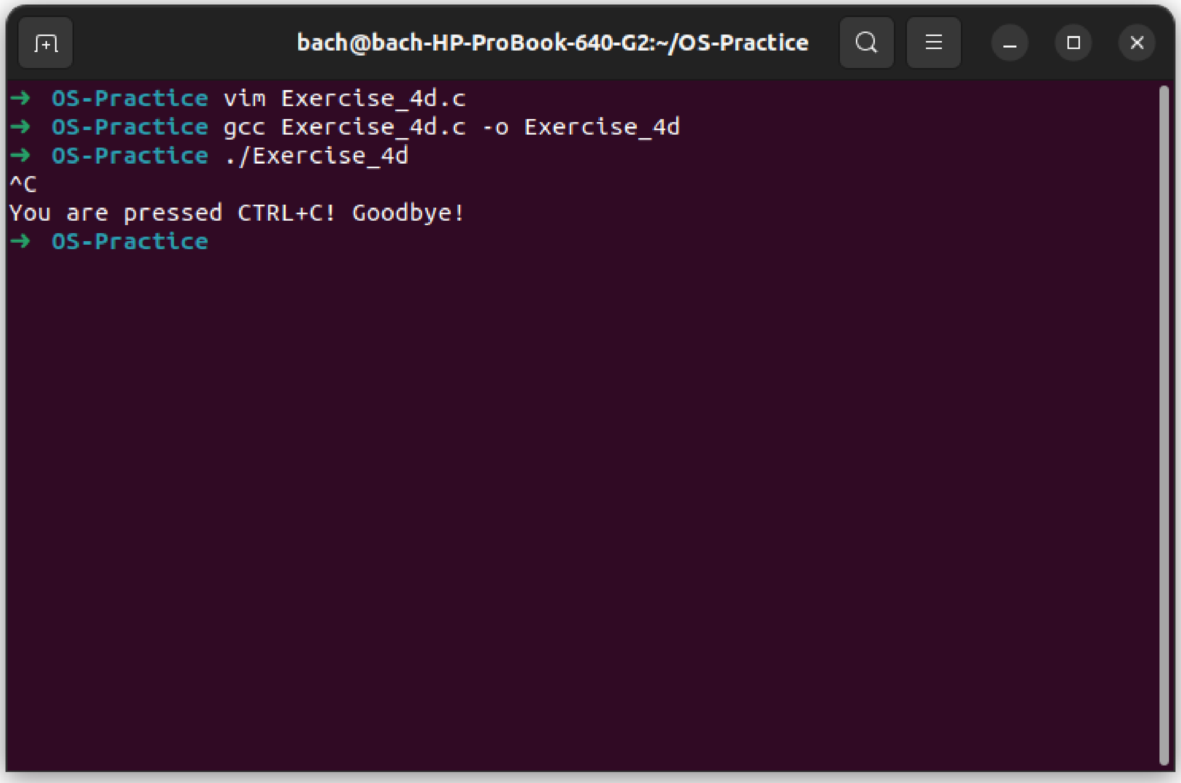

bach@bach-HP-ProBook-640-G2:~/OS-Practice
bach@bach-HP-ProBook-640-G2:~/OS-P...
bach@bach-HP-ProBook-640-G2:~/OS-P...

```
➜  OS-Practice vim Exercise_4c.c
➜  OS-Practice gcc Exercise_4c.c -o Exercise_4c
➜  OS-Practice ./Exercise_4c
➜  OS-Practice
```

d.

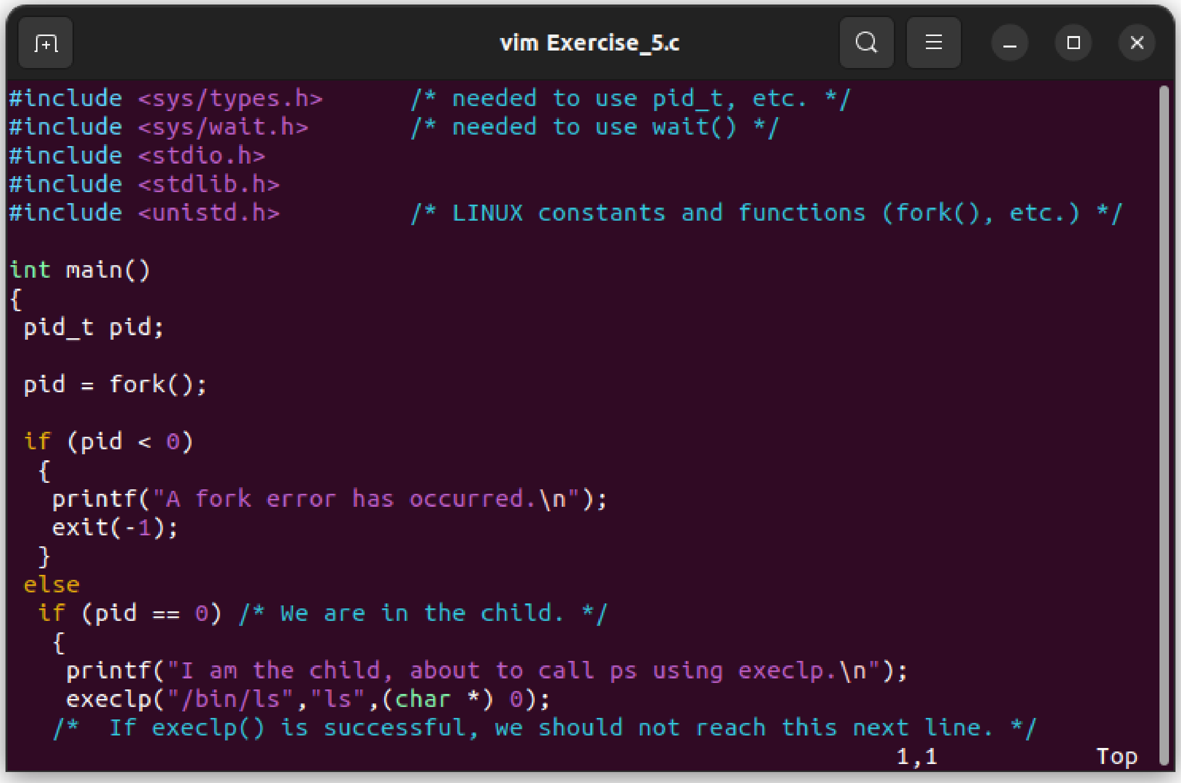


```
#include <stdio.h>
#include <signal.h>

int loop_forever = 1;

void on_signal() {
        printf("\nYou are pressed CTRL+C! Goodbye!\n");
        loop_forever = 0;
}

int main() {
        loop_forever = 1;
        signal(SIGINT, on_signal);
        while(loop_forever){}
        return 0;
}
~
~
~
~
~
~
~
-- INSERT --                                         15,1          All
```

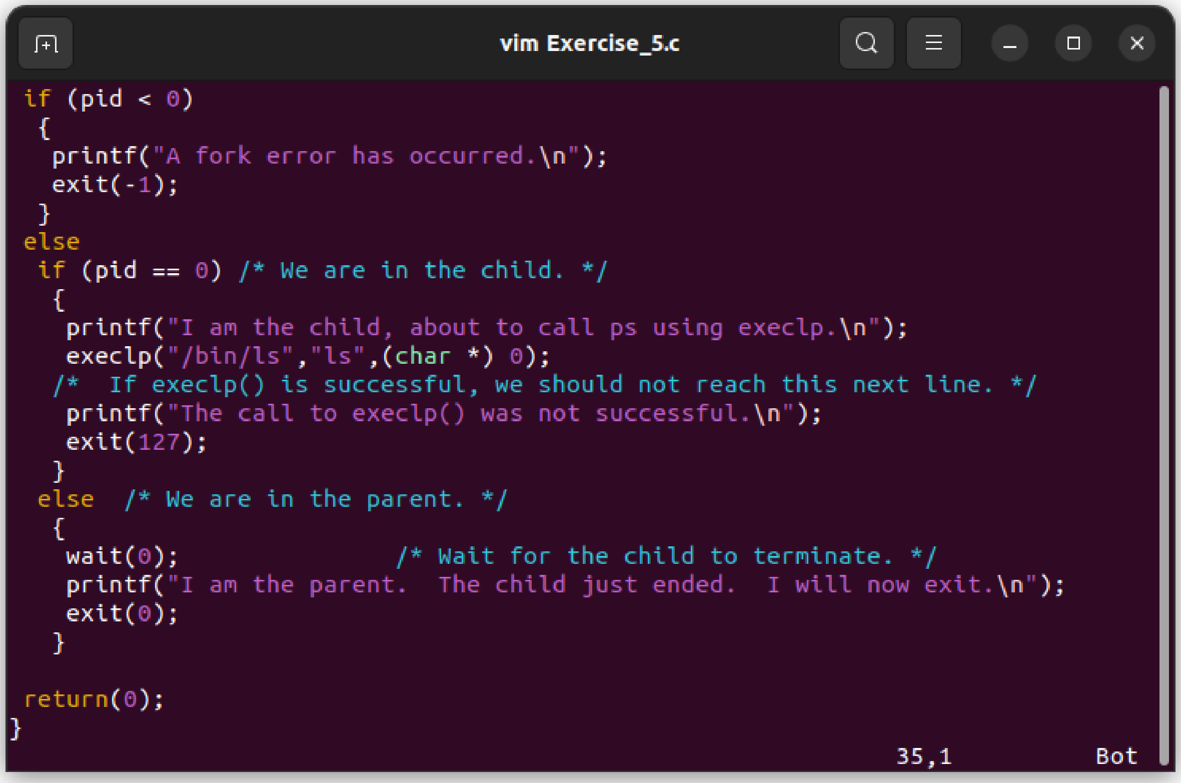


```
➜  OS-Practice vim Exercise_4d.c
➜  OS-Practice gcc Exercise_4d.c -o Exercise_4d
➜  OS-Practice ./Exercise_4d
^C
You are pressed CTRL+C! Goodbye!
➜  OS-Practice
```

3.5.5.
This is an example about how to using execlp()
File Exercise_5.c

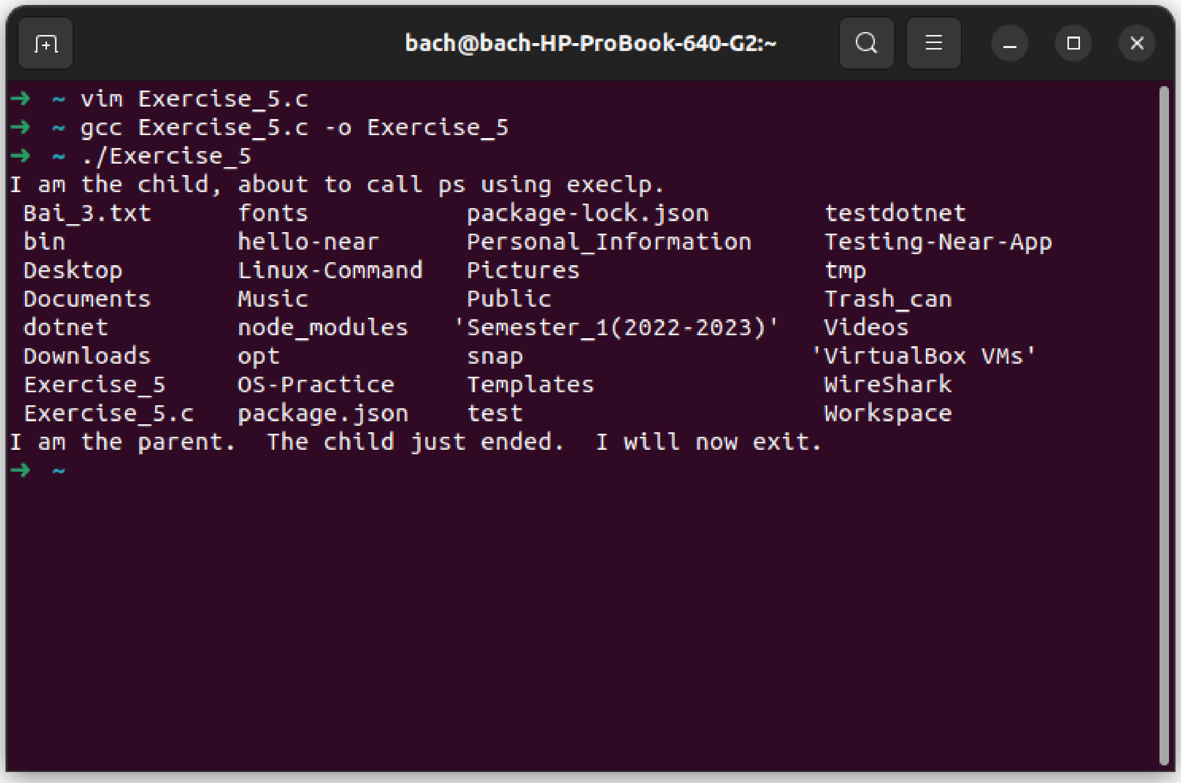


```
#include <sys/types.h>      /* needed to use pid_t, etc. */
#include <sys/wait.h>       /* needed to use wait() */
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>         /* LINUX constants and functions (fork(), etc.) */

int main()
{
 pid_t pid;

 pid = fork();

 if (pid < 0)
  {
   printf("A fork error has occurred.\n");
   exit(-1);
  }
 else
  if (pid == 0) /* We are in the child. */
   {
    printf("I am the child, about to call ps using execlp.\n");
    execlp("/bin/ls","ls",(char *) 0);
   /*  If execlp() is successful, we should not reach this next line. */
                                                              1,1           Top
```



```
 if (pid < 0)
  {
   printf("A fork error has occurred.\n");
   exit(-1);
  }
 else
  if (pid == 0) /* We are in the child. */
   {
    printf("I am the child, about to call ps using execlp.\n");
    execlp("/bin/ls","ls",(char *) 0);
   /*  If execlp() is successful, we should not reach this next line. */
    printf("The call to execlp() was not successful.\n");
    exit(127);
   }
  else  /* We are in the parent. */
   {
    wait(0);                   /* Wait for the child to terminate. */
    printf("I am the parent.  The child just ended.  I will now exit.\n");
    exit(0);
   }

 return(0);
}
                                                              35,1          Bot
```

Result of running Exercise_5.c



```
➜  ~ vim Exercise_5.c
➜  ~ gcc Exercise_5.c -o Exercise_5
➜  ~ ./Exercise_5
I am the child, about to call ps using execlp.
 Bai_3.txt      fonts           package-lock.json        testdotnet
 bin            hello-near      Personal_Information     Testing-Near-App
 Desktop        Linux-Command   Pictures                 tmp
 Documents      Music           Public                   Trash_can
 dotnet         node_modules   'Semester_1(2022-2023)'   Videos
 Downloads      opt             snap                    'VirtualBox VMs'
 Exercise_5     OS-Practice     Templates                WireShark
 Exercise_5.c   package.json    test                     Workspace
I am the parent.  The child just ended.  I will now exit.
➜  ~
```

execXX calls as a group

- The calls with v in the name take an array parameter to specify the argv[] array of the new program. The end of the arguments is indicated by an array element containing NULL.
- The calls with l in the name take the arguments of the new program as a variable-length argument list to the function itself. The end of the arguments is indicated by a (char *)NULL argument. You should always include the type cast, because NULL is allowed to be an integer constant, and default argument conversions when calling a variadic function won't convert that to a pointer.
- The calls with e in the name take an extra argument (or arguments in the l case) to provide the environment of the new program; otherwise, the program inherits the current process's environment. This is provided in the same way as the argv array: an array for execve(), separate arguments for execle().
- The calls with p in the name search the PATH environment variable to find the program if it doesn't have a directory in it (i.e. it doesn't contain a / character). Otherwise, the program name is always treated as a path to the executable.
- FreeBSD 5.2 added another variant: execvP (with uppercase P). This is like execvp(), but instead of getting the search path from the PATH environment variable, it's an explicit parameter to the function: