

Họ và tên: Nguyễn Nguyên Ngọc Anh

Mã số sinh viên: 22520058

Lớp: IT007.O11.1

HỆ ĐIỀU HÀNH BÁO CÁO LAB 6

CHECKLIST

6.4. BÀI TẬP THỰC HÀNH

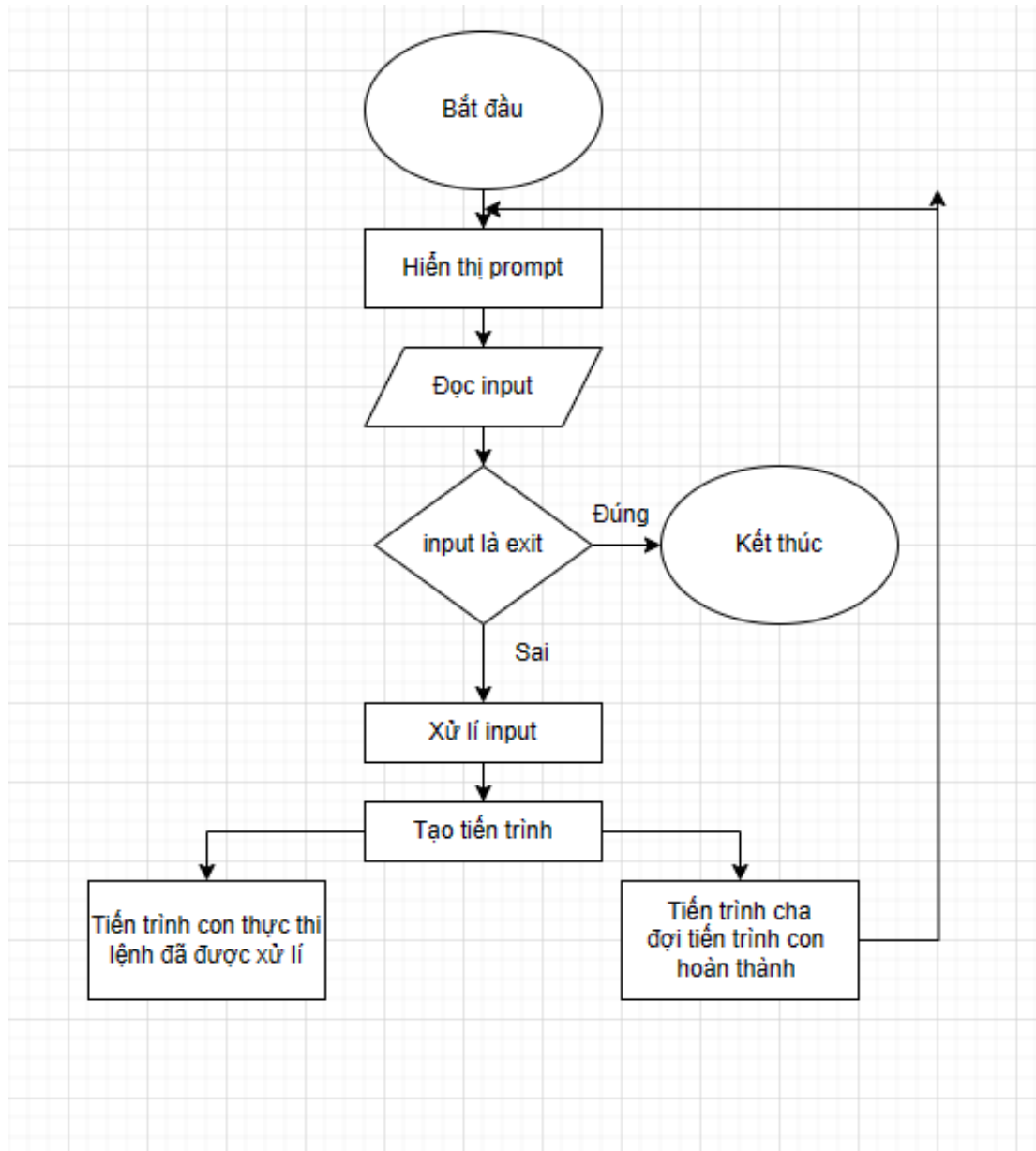
	Câu 1	Câu 2	Câu 3	Câu 4	Câu 5
Trình bày giải thuật	Có <input type="checkbox"/>	Có <input type="checkbox"/>	Có <input type="checkbox"/>	Có <input type="checkbox"/>	Có <input type="checkbox"/>
Chụp hình minh chứng (chạy ít nhất 3 lệnh)	Có <input type="checkbox"/>	Có <input type="checkbox"/>	Có <input type="checkbox"/>	Có <input type="checkbox"/>	Có <input type="checkbox"/>
Giải thích code, kết quả	Có <input type="checkbox"/>	Có <input type="checkbox"/>	Có <input type="checkbox"/>	Có <input type="checkbox"/>	Có <input type="checkbox"/>

Tư chấm điểm: 10

6.4. BÀI TẬP THỰC HÀNH

1. Câu 1

- Trình bày giải thuật :



- Chụp hình minh chứng :

```
Lab6 > C 1.c > main(void)
4  #include <string.h>
5  #include <sys/wait.h>
6  #include <fcntl.h>
7  #include <stdbool.h>
8  #include <stdlib.h>
9  #include <ctype.h>
10 #include <signal.h>
11 #define MAX_LINE 80 // maximum length command
12
13 pid_t pid;
14
15 int main(void)
16 {
17     char *args[MAX_LINE / 2 + 1];
18     int should_run = 1;
19
20     while (should_run)
21     {
22         printf("it007sh>");
23         fflush(stdout);
24
25         // get input
26         char input[MAX_LINE];
27         fgets(input, MAX_LINE, stdin);
28
29         if (strcmp(input, "exit\n") == 0)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
nguyennguyennngocanh-22520058@DESKTOP-7R66M1N:~/Lab6$ ./1
it007sh>pwd
/home/nguyennguyennngocanh-22520058/Lab6
it007sh>ls
1 1.c 2 2.c 3 3.c 4 4.c 5 5.c baongu.txt doc.txt lab6 lab6.c lab6test ngocanh.txt ruoiconbangbaongu.txt test.txt text.txt
it007sh>whoami
nguyennguyennngocanh-22520058
it007sh>
```

- Giải thích code, kết quả :

+ Code :

```
#include <stdio.h>
#include <unistd.h>
#include <string.h>
#include <sys/wait.h>
#include <fcntl.h>
#include <stdbool.h>
#include <stdlib.h>
```

```
#include <ctype.h>
#include <signal.h>
#define MAX_LINE 80 // maximum length command

pid_t pid;

int main(void)
{
    char *args[MAX_LINE / 2 + 1];
    int should_run = 1;

    while (should_run)
    {
        printf("it007sh>");
        fflush(stdout);

        // get input
        char input[MAX_LINE];
        fgets(input, MAX_LINE, stdin);

        if (strcmp(input, "exit\n") == 0)
        {
            should_run = 0;
            continue;
        }
        else
        {
            int i = 0;
```

```
args[i] = strtok(input, " \n");
while (args[i] != NULL)
{
    i++;
    args[i] = strtok(NULL, " \n");
}
// end of args is NULL for execvp() know to end
args[i] = NULL;

pid = fork();

if (pid < 0)
{
    perror("Fork err!");
}
else if (pid == 0)
{
    execvp(args[0], args);
    perror("execvp");
    return 1;
}

else
{
    waitpid(pid, NULL, 0);
}
}
```

```
return 1;  
}
```

1.Khởi tạo:

Chương trình khởi tạo các biến, bao gồm một mảng args để lưu trữ các đối số dòng lệnh và một biến pid để lưu trữ ID của quá trình.

Nó thiết lập một vòng lặp while (should_run) để giữ shell chạy cho đến khi should_run trở thành 0.

2.Nhập lệnh:

Hiển thị dấu nhắc lệnh ("it007sh>") và chờ đợi đầu vào từ người dùng thông qua fgets. Lệnh nhập vào được lưu trong biến input.

3.Kiểm tra lệnh Exit:

Nếu lệnh nhập vào là "exit\n", trong đó \n đại diện cho phím Enter, biến should_run được thiết lập thành 0, và chương trình sẽ thoát khỏi vòng lặp và kết thúc.

4.Phân tích Lệnh:

Nếu đầu vào không phải là "exit", chương trình tiến hành tách chuỗi lệnh nhập thành các đối số riêng lẻ bằng cách sử dụng strtok(). Nó tách chuỗi đầu vào bằng khoảng trắng và ký tự xuống dòng ("\n") và lưu mỗi đoạn trong mảng args.

5.Tạo Tiến trình Con:

Sau khi phân tích lệnh, chương trình tạo một tiến trình con bằng cách sử dụng fork().

Nếu fork() thất bại (pid < 0), một thông báo lỗi được hiển thị bằng perror.

Nếu fork() thành công và quá trình hiện tại là tiến trình con (pid == 0), nó thực thi lệnh nhập bằng execvp(args[0], args). Điều này thay thế quá trình con bằng lệnh được chỉ định.

Nếu execvp() gặp lỗi, nó hiển thị thông báo lỗi bằng perror và trả về 1 để báo lỗi cho quá trình cha.

6.Xử lý Quá trình Cha:

Báo cáo thực hành môn Hệ điều hành - Giảng viên: Trần Hoàng Lộc.

Trong quá trình cha ($pid > 0$), nó đợi quá trình con hoàn thành bằng `waitpid(pid, NULL, 0)`. Điều này đảm bảo rằng quá trình cha đợi quá trình con kết thúc thực thi trước khi tiếp tục.

7. Tiếp tục Vòng Lặp:

Sau khi quá trình con hoàn thành và quá trình cha đợi nó, vòng lặp tiếp tục, yêu cầu lệnh tiếp theo (`it007sh>`).

8. Thoát:

Khi nhập "exit", chương trình thoát khỏi vòng lặp và kết thúc.

+ kết quả:

- **Lệnh pwd :**

Khi thực thi lệnh `pwd` sẽ cho ta biết vị trí đang hoạt động.

- **Lệnh ls:**

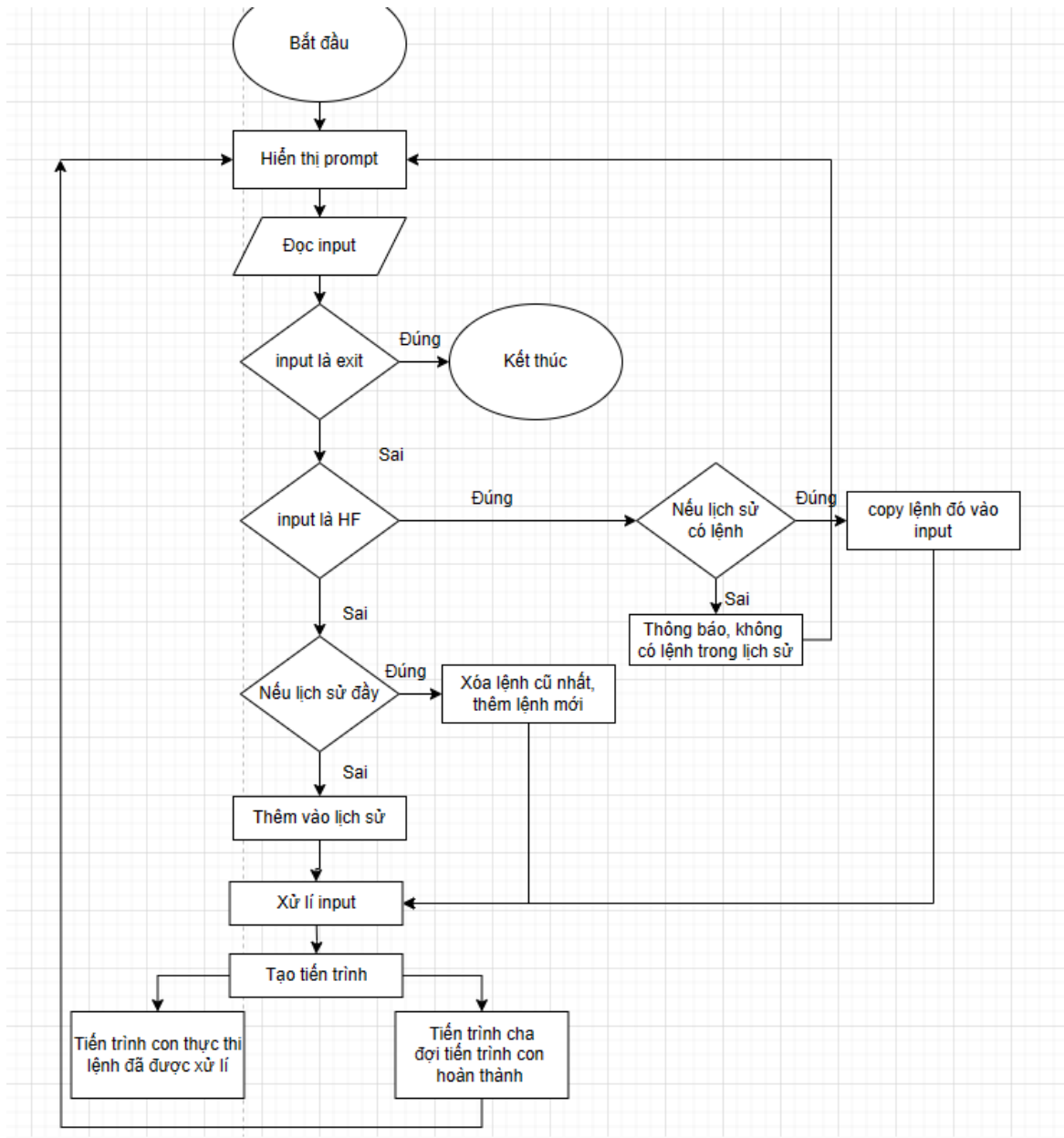
Khi thực thi lệnh `ls` sẽ cho ta biết tất cả file có trong directory đang hoạt động.

- **Lệnh whoami:**

Khi thực thi lệnh `whoami` sẽ cho ta biết tên user đang hoạt động.

2. Câu 2

- **Trình bày giải thuật :**



- Chụp hình minh chứng :


```
Lab6 > C 2.c > main(void)
31 fgets(input, MAX_LINE, stdin);
32
33 if (strcmp(input, "exit\n") == 0)
34 {
35     should_run = 0;
36     continue;
37 }
38
39 // 2.
40 // if input == HF
41 if (strcmp(input, "HF\n") == 0)
42 {
43     // check history if have cmd
44     if (history_count > 0 && (history_count - HF_count) >= 0)
45     {
46         // copy from history to input
47         strcpy(input, history[history_count - HF_count]);
48         printf("current cmd is : %s", input);
49         HF_count++;
50     }
51     else
52     {
53         printf("no recent cmd found!\n");
54     }
55 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
it007sh>echo Hello Thay Tung
Hello Thay Tung
it007sh>ls
1 1.c 2 2.c 3 3.c 4 4.c 5 5.c baongu.txt doc.txt lab6 lab6.c lab6test ngocanh.txt ruoiconbangbaongu.txt test.txt text.txt
it007sh>pwd
/home/nguyennnguyenngocanh-22520058/Lab6
it007sh>HF
current cmd is : pwd
/home/nguyennnguyenngocanh-22520058/Lab6
it007sh>HF
current cmd is : ls
1 1.c 2 2.c 3 3.c 4 4.c 5 5.c baongu.txt doc.txt lab6 lab6.c lab6test ngocanh.txt ruoiconbangbaongu.txt test.txt text.txt
it007sh>HF
current cmd is : echo Hello Thay Tung
Hello Thay Tung
it007sh>
```

- Giải thích code, kết quả :

+ Code :

2.

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

```
#include <unistd.h>
```

```
#include <string.h>
```

```
#include <sys/wait.h>
```

```
#include <fcntl.h>
```

```
#include <stdbool.h>
#include <stdlib.h>
#include <ctype.h>
#include <signal.h>
#define MAX_LINE 80 // maximum length command
#define HISTORY_SIZE 10

char history[HISTORY_SIZE][MAX_LINE];
int history_count = 0;
int HF_count = 1;
pid_t pid;

int main(void)
{
    char *args[MAX_LINE / 2 + 1];
    int should_run = 1;

    while (should_run)
    {
        printf("it007sh>");
        fflush(stdout);

        // get input
        char input[MAX_LINE];
        fgets(input, MAX_LINE, stdin);

        if (strcmp(input, "exit\n") == 0)
        {
```

```
    should_run = 0;
    continue;
}
// 2.
// if input == HF
if (strcmp(input, "HF\n") == 0)
{
    // check history if have cmd
    if (history_count > 0 && (history_count - HF_count) >= 0)
    {
        // copy from history to input
        strcpy(input, history[history_count - HF_count]);
        printf("current cmd is : %s", input);
        HF_count++;
    }
    else
    {
        printf("no recent cmd found!\n");
        continue;
    }
}
else
{
    HF_count = 1;
    // add input to history
    if (history_count < HISTORY_SIZE)
    {
        strcpy(history[history_count], input);
```

```
        history_count++;
    }
    else
        // if history full, remove first in history cmd and add input to the last
        {
            for (int i = 0; i < HISTORY_SIZE; i++)
            {
                // copy i+1 into i to have space in HISTORY_SIZE-1
                strcpy(history[i], history[i + 1]);
            }
            // add input to history
            strcpy(history[HISTORY_SIZE - 1], input);
        }
    }

    int i = 0;
    args[i] = strtok(input, " \n");
    while (args[i] != NULL)
    {
        i++;
        args[i] = strtok(NULL, " \n");
    }
    // end of args is NULL for execvp() know to end
    args[i] = NULL;

    pid = fork();

    if (pid == 0)
```

```
{
    execvp(args[0], args);
    perror("execvp");
    return 1;
}
else if (pid < 0)
{
    perror("Fork err!");
}
else
{
    waitpid(pid, NULL, 0);
}
}
return 1;
}
```

1.Khởi tạo:

Biến history là một mảng 2 chiều để lưu trữ lịch sử các lệnh đã nhập.

history_count là biến đếm số lượng lệnh đã lưu trong lịch sử.

HF_count là biến dùng để thao tác với lệnh trong lịch sử.

2.Lặp:

Chương trình chạy trong một vòng lặp while (should_run) để duy trì chạy shell cho đến khi người dùng nhập lệnh "exit\n".

3.Nhập Lệnh và Kiểm tra Lịch sử:

Người dùng nhập lệnh vào input.

Nếu lệnh nhập là "exit\n", chương trình sẽ dừng.

Nếu lệnh nhập là "HF\n", nó sẽ kiểm tra lịch sử để tìm và lấy lệnh gần nhất và thực hiện lệnh đó.

Nếu lệnh không phải là "HF\n":

Nếu lịch sử chưa đầy (`history_count < HISTORY_SIZE`), lệnh mới sẽ được thêm vào lịch sử.

Nếu lịch sử đã đầy (`history_count == HISTORY_SIZE`), lệnh mới sẽ thêm vào cuối danh sách và lệnh đầu tiên sẽ bị xóa.

4.Xử lý Lệnh:

Sau đó, lệnh nhập được phân tách thành các đối số trong mảng `args`.

Tiến trình con được tạo ra bằng `fork()`, và nó sẽ thực thi lệnh đã nhập bằng `execvp(args[0], args)`.

5.Xử lý Lệnh Con và Chờ Kết thúc:

Nếu `fork()` thành công, quá trình cha sẽ chờ đợi tiến trình con kết thúc bằng `waitpid(pid, NULL, 0)`.

6.Lặp Lại Quá trình:

Sau khi tiến trình con hoàn thành và quá trình cha chờ đợi, vòng lặp tiếp tục, chờ đợi lệnh nhập tiếp theo từ người dùng.

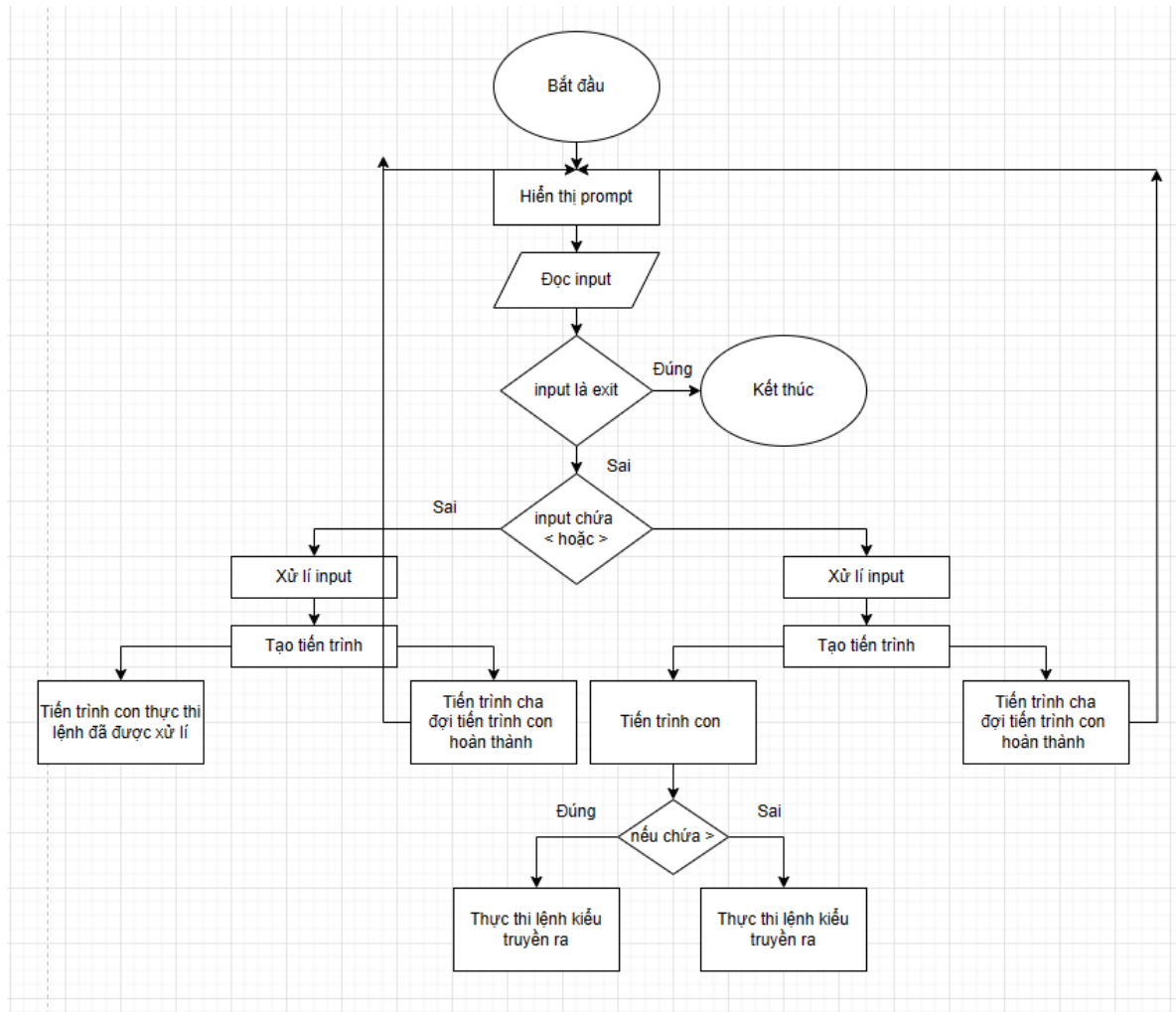
+ kết quả:

Ta đã chạy lần lượt 3 lệnh đó là `<echo Hello Thay Tung>`, `ls`, `pwd`

Khi ta gõ HF thì chương trình sẽ chạy lệnh gần nhất là `pwd`, khi ta gõ HF tiếp tục sẽ thực hiện lệnh `ls`, gõ HF tiếp tục sẽ thực hiện lệnh `echo Hello Thay Tung`.

3. Câu 3

- Trình bày giải thuật :



- Chụp hình minh chứng :

Báo cáo thực hành môn Hệ điều hành - Giảng viên: Trần Hoàng Lộc.

```
lab6.c 1.c 2.c 3.c X 4.c 5.c
Lab6 > C 3.c > main(void)
40 fgets(input, MAX_LINE, stdin);
41
42 if (strcmp(input, "exit\n") == 0)
43 {
44     should_run = 0;
45     continue;
46 }
47
48 int i = 0;
49 args[i] = strtok(input, " \n");
50 while (args[i] != NULL)
51 {
52     i++;
53     args[i] = strtok(NULL, " \n");
54 }
55 // end of args is NULL for execvp() know to end
56 args[i] = NULL;
57 if (has_redirection(args, i))
58 {
59     // create new array input to execvp cmd forward the redirection( "<", ">")
60     int index = 0;
61     char *command_redirection[MAX_LINE / 2 + 1];
62     bool input_redirection = false;
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

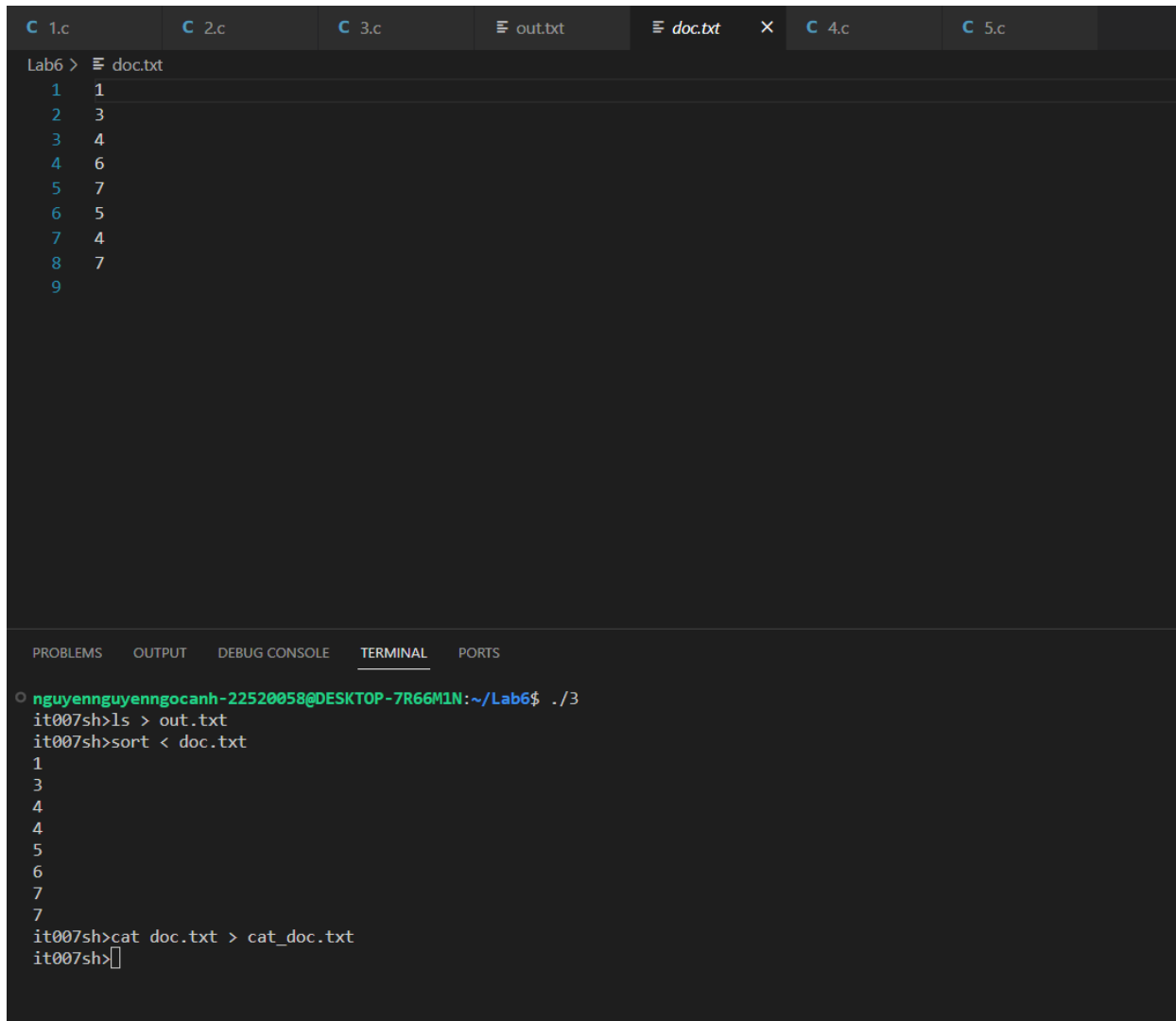
```
nguyennnguyenngocanh-22520058@DESKTOP-7R66M1N:~/Lab6$ ./3
it007sh>ls > out.txt
it007sh>sort < doc.txt
1
3
4
4
5
6
7
7
it007sh>cat doc.txt > cat_doc.txt
it007sh>
```


Báo cáo thực hành môn Hệ điều hành - Giảng viên: Trần Hoàng Lộc.

```
1.c 2.c 3.c out.txt 4.c 5.c
Lab6 > out.txt
1 1
2 1.c
3 2
4 2.c
5 3
6 3.c
7 4
8 4.c
9 5
10 5.c
11 baongu.txt
12 doc.txt
13 lab6
14 lab6.c
15 lab6test
16 ngocanh.txt
17 out.txt
18 ruoiconbangbaongu.txt
19 test.txt
20 text.txt
21

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
○ nguyennguyennngocanh-22520058@DESKTOP-7R66M1N:~/Lab6$ ./3
it007sh>ls > out.txt
it007sh>sort < doc.txt
1
3
4
4
5
6
7
7
it007sh>cat doc.txt > cat_doc.txt
it007sh>
```

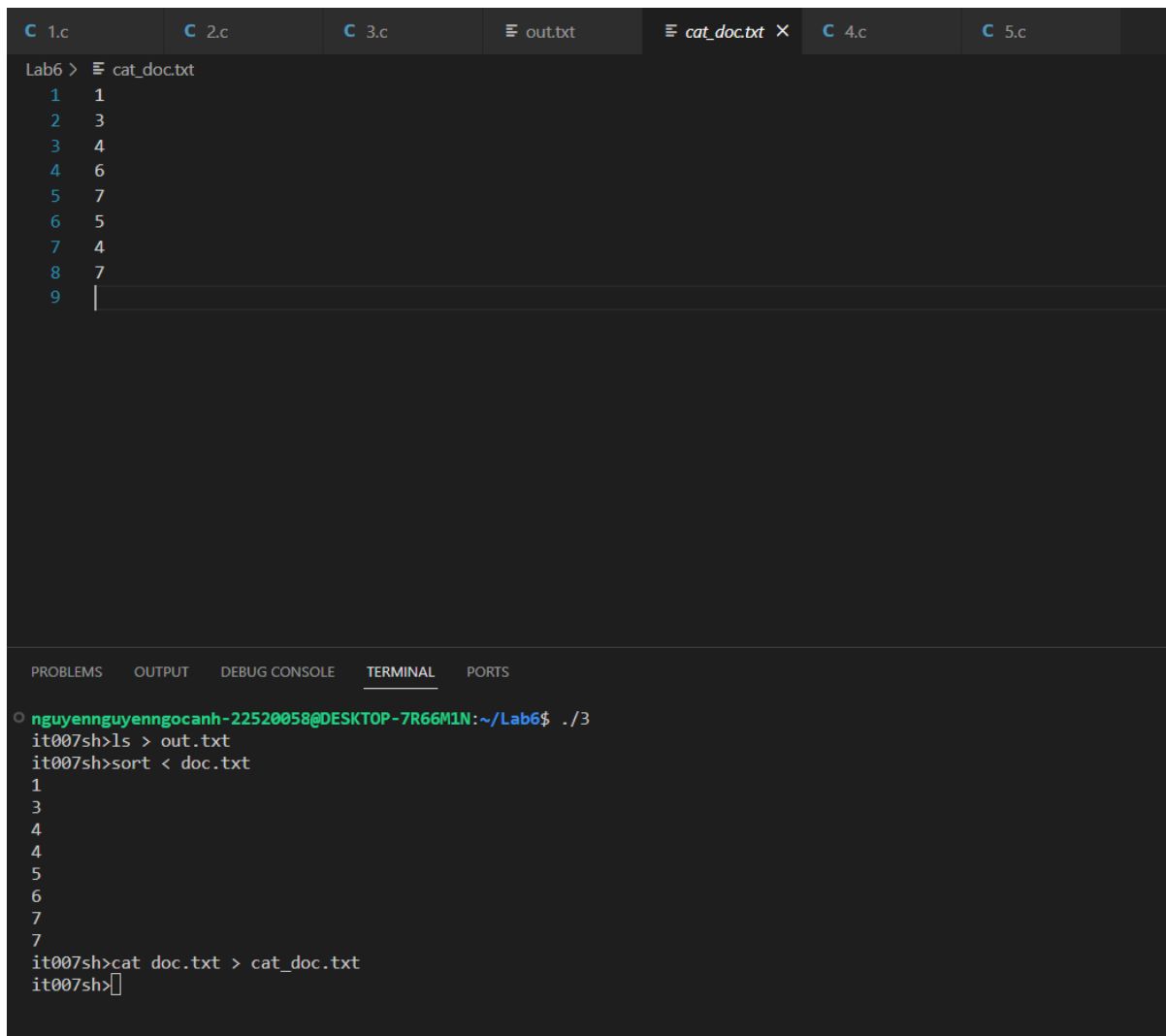
Báo cáo thực hành môn Hệ điều hành - Giảng viên: Trần Hoàng Lộc.



The image shows a Visual Studio Code editor window with a C program in `doc.txt` and a terminal window below it. The C program reads 9 integers from `in.txt` and prints them. The terminal shows the execution of the program, which outputs the numbers 1, 3, 4, 4, 5, 6, 7, 7, and 7.

```
Lab6 > doc.txt
1 1
2 3
3 4
4 6
5 7
6 5
7 4
8 7
9

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
nguyennguyenngocanh-22520058@DESKTOP-7R66M1N:~/Lab6$ ./3
it007sh>ls > out.txt
it007sh>sort < doc.txt
1
3
4
4
5
6
7
7
it007sh>cat doc.txt > cat_doc.txt
it007sh>
```



The screenshot shows a VS Code editor with several tabs: 1.c, 2.c, 3.c, out.txt, cat_doc.txt, 4.c, and 5.c. The active tab is cat_doc.txt, which contains the following content:

```
Lab6 > cat_doc.txt
1 1
2 3
3 4
4 6
5 7
6 5
7 4
8 7
9 |
```

Below the editor, the TERMINAL panel is open, showing the following commands and output:

```
nguyennguyennhocanh-22520058@DESKTOP-7R66M1N:~/Lab6$ ./3
it007sh>ls > out.txt
it007sh>sort < doc.txt
1
3
4
4
5
6
7
7
it007sh>cat doc.txt > cat_doc.txt
it007sh>
```

- Giải thích code, kết quả :

+ Code :

```
#include <stdio.h>
#include <unistd.h>
#include <string.h>
#include <sys/wait.h>
#include <fcntl.h>
#include <stdbool.h>
#include <stdlib.h>
#include <ctype.h>
```

```
#include <signal.h>

#define MAX_LINE 80 // maximum length command

pid_t pid;
int file_in;
int file_out;

// function to check args contains >,<
bool has_redirection(char *args[], int size)
{
    for (int i = 0; i < size; i++)
    {
        if (strcmp(args[i], ">") == 0 || strcmp(args[i], "<") == 0)
            return true;
    }
    return false;
}

int main(void)
{
    char *args[MAX_LINE / 2 + 1];
    int should_run = 1;

    while (should_run)
    {
        printf("it007sh>");
        fflush(stdout);
```

```
// get input
char input[MAX_LINE];
fgets(input, MAX_LINE, stdin);

if (strcmp(input, "exit\n") == 0)
{
    should_run = 0;
    continue;
}

int i = 0;
args[i] = strtok(input, " \n");
while (args[i] != NULL)
{
    i++;
    args[i] = strtok(NULL, " \n");
}
// end of args is NULL for execvp() know to end
args[i] = NULL;
if (has_redirection(args, i))
{
    // create new array input to execvp cmd forward the redirection( "<", ">")
    int index = 0;
    char *command_redirection[MAX_LINE / 2 + 1];
    bool input_redirection = false;

    // check in args if not null and not contain the redirection => add the args
    to new array
```

```
while (args[index] != NULL && strcmp(args[index], ">") != 0 &&
strcmp(args[index], "<") != 0)
{
    command_redirection[index] = args[index];
    index++;
}
// last index is NULL to end the execvp
command_redirection[index] = NULL;

// check the redirection if(>) => bool = false else bool = true
if (strcmp(args[index], ">") == 0)
{
    input_redirection = false;
}
else if (strcmp(args[index], "<") == 0)
{
    input_redirection = true;
}

pid = fork();

if (pid == 0)
{
    // if >
    if (!input_redirection)
    {
        file_out = open(args[index + 1], O_WRONLY | O_CREAT |
O_TRUNC, 0666);
```

```
    if (file_out == -1)
    {
        perror("open");
        return 1;
    }

    dup2(file_out, STDOUT_FILENO);
    close(file_out);

    execvp(command_redirection[0], command_redirection);
}
// if <
else
{
    file_in = open(args[index + 1], O_RDONLY, 0666);
    if (file_in == -1)
    {
        perror("open");
        return 1;
    }

    dup2(file_in, STDIN_FILENO);
    close(file_in);

    execvp(command_redirection[0], command_redirection);
}
perror("execvp!");
return 1;
```

```
    }  
    else if (pid < 0)  
    {  
        perror("fork err!");  
    }  
    else  
    {  
        waitpid(pid, NULL, 0);  
    }  
}  
else  
{  
    pid = fork();  
  
    if (pid == 0)  
    {  
        // 5.  
        // subscribe signal to child_process  
  
        execvp(args[0], args);  
        perror("execvp");  
        return 1;  
    }  
    else if (pid < 0)  
    {  
        perror("Fork err!");  
    }  
    else
```



```
{  
    // receive ctrl c signal and interrupt the process  
    waitpid(pid, NULL, 0);  
}  
}  
}  
return 1;  
}
```

1.Lặp:

Chương trình chạy trong vòng lặp while (should_run) để tiếp tục chạy shell cho đến khi người dùng nhập lệnh "exit\n".

2.Nhập Lệnh và Kiểm tra Chuyển hướng:

Người dùng nhập lệnh vào biến input.

Nếu lệnh nhập là "exit\n", chương trình dừng.

Lệnh được phân tách thành các đối số trong mảng args.

3.Kiểm tra Chuyển hướng:

Hàm has_redirection kiểm tra xem lệnh có chứa > hoặc < không.

Nếu có chuyển hướng, chương trình tách lệnh thành hai phần: phần trước và sau chuyển hướng.

4.Chuyển hướng Đầu vào/Đầu ra:

Nếu có chuyển hướng:

Tạo một tiến trình con bằng fork().

Đối với chuyển hướng > (đầu ra):

Mở hoặc tạo tệp theo tên được cung cấp trong lệnh.

Sử dụng dup2() để đổi hướng đầu ra của tiến trình con sang tệp đã mở.

Thực thi lệnh với execvp.

Đối với chuyển hướng < (đầu vào):

Mở tệp theo tên được cung cấp trong lệnh.

Báo cáo thực hành môn Hệ điều hành - Giảng viên: Trần Hoàng Lộc.

Sử dụng `dup2()` để đổi hướng đầu vào của tiến trình con từ tệp đã mở.

Thực thi lệnh với `execvp`.

Nếu không có chuyển hướng:

Tạo một tiến trình con mới bằng `fork()` và thực thi lệnh đầu vào.

5.Chờ Tiến trình Con Kết thúc:

Quá trình cha chờ tiến trình con kết thúc bằng `waitpid(pid, NULL, 0)`.

6.Lặp Lại Quá trình:

Sau khi tiến trình con hoàn thành và quá trình cha chờ đợi, vòng lặp tiếp tục, chờ đợi lệnh nhập tiếp theo từ người dùng.

+ kết quả:

Khi thực thi lệnh `ls > out.txt` thì tất cả file đang trong thư mục hoạt động sẽ được liệt kê

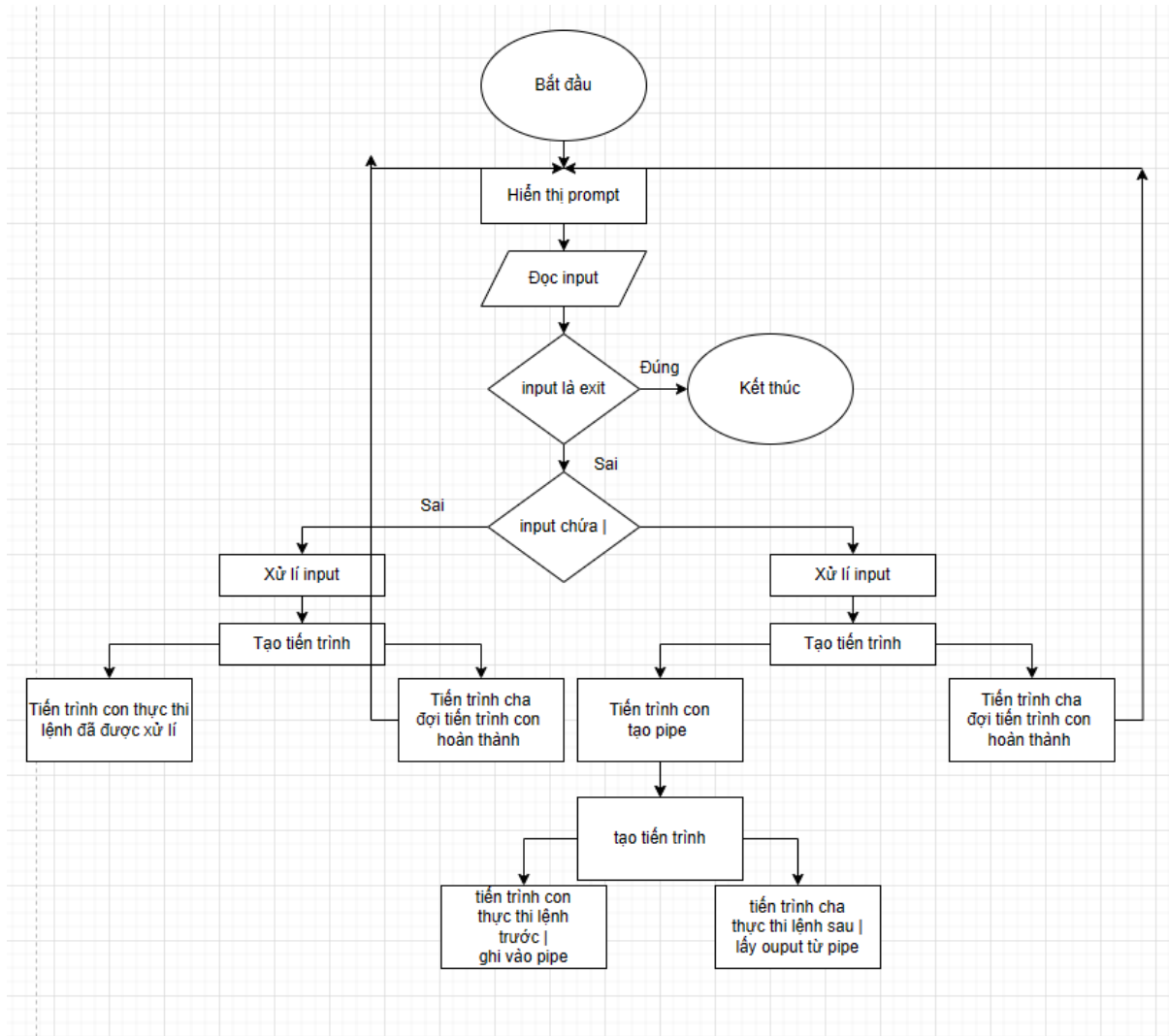
Vào file `out.txt`

Khi thực hiện lệnh `sort < doc.txt` thì sẽ cho ra dãy đã sắp xếp từ dữ liệu có trong file `doc.txt`

Khi thực hiện lệnh `cat doc.txt > cat_doc.txt` thì sẽ đọc tất cả thông tin có trong file `doc.txt` và ghi vào file `cat_doc.txt`

4. Câu 4

- **Trình bày giải thuật :**



- Chụp hình minh chứng :

```
C 4.c x
Lab6 > C 4.c > main(void)
1
2 #include <stdio.h>
3 #include <unistd.h>
4 #include <string.h>
5 #include <sys/wait.h>
6 #include <fcntl.h>
7 #include <stdbool.h>
8 #include <stdlib.h>
9 #include <ctype.h>
10 #include <signal.h>
11 #define MAX_LINE 80 // maximum length command
12
13 // function to check args contains |
14
15 int main(void)
16 {
17     char *args[MAX_LINE / 2 + 1];
18     int should_run = 1;
19
20     while (should_run)
21     {
22         printf("it007sh>");
23         fflush(stdout);
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
nguyennnguyenngocanh-22520058@DESKTOP-7R66M1N:~$ gcc 4.c -o 4
cc1: fatal error: 4.c: No such file or directory
compilation terminated.
nguyennnguyenngocanh-22520058@DESKTOP-7R66M1N:~$ cd Lab6
nguyennnguyenngocanh-22520058@DESKTOP-7R66M1N:~/Lab6$ gcc 4.c -o 4
nguyennnguyenngocanh-22520058@DESKTOP-7R66M1N:~/Lab6$ ./4
it007sh>ps aux | grep firefox
nguyenn+ 677 0.0 0.0 4020 2076 pts/1 S+ 23:39 0:00 grep firefox
it007sh>ps aux | grep chrome
nguyenn+ 709 0.0 0.0 4020 2024 pts/1 S+ 23:39 0:00 grep chrome
it007sh>ps aux | grep edge
nguyenn+ 771 0.0 0.0 4020 2092 pts/1 S+ 23:39 0:00 grep edge
it007sh>
```

- Giải thích code, kết quả :

+ Code :

```
#include <stdio.h>
#include <unistd.h>
#include <string.h>
#include <sys/wait.h>
#include <fcntl.h>
#include <stdbool.h>
#include <stdlib.h>
#include <ctype.h>
```

```
#include <signal.h>

#define MAX_LINE 80 // maximum length command

// function to check args contains |

int main(void)
{
    char *args[MAX_LINE / 2 + 1];
    int should_run = 1;

    while (should_run)
    {
        printf("it007sh>");
        fflush(stdout);

        // get input
        char input[MAX_LINE];
        fgets(input, MAX_LINE, stdin);

        if (strcmp(input, "exit\n") == 0)
        {
            should_run = 0;
            continue;
        }

        int i = 0;
        args[i] = strtok(input, " \n");
        while (args[i] != NULL)
```

```
{
    i++;
    args[i] = strtok(NULL, " \n");
}
// end of args is NULL for execvp() know to end
args[i] = NULL;

pid_t pid = fork();

if (pid < 0)
{
    perror("fork");
}
else if (pid == 0)
{
    // create cmd front pipe, cmd behind pipe
    char *back_pipe[MAX_LINE / 2 + 1];
    char *front_pipe[MAX_LINE / 2 + 1];

    // get args to back_pipe
    back_pipe[0] = args[3];
    if (args[4] != NULL)
        back_pipe[1] = args[4];
    back_pipe[2] = NULL;

    // get args to front_pipe
```

```
front_pipe[0] = args[0];
if (args[1] != NULL)
    front_pipe[1] = args[1];
front_pipe[2] = NULL;

int pipefd[2];

// create pipe
if (pipe(pipefd) == -1)
{
    perror("pipe");
}

pid_t pid_child = fork();

if (pid_child < 0)
{
    perror("fork");
}
else if (pid_child == 0)
{
    close(pipefd[0]);

    dup2(pipefd[1], STDOUT_FILENO);

    close(pipefd[1]);

    execvp(front_pipe[0], front_pipe);
```

```
        perror("execvp");
    }
    else
    {
        close(pipefd[1]);

        dup2(pipefd[0], STDIN_FILENO);

        close(pipefd[0]);

        execvp(back_pipe[0], back_pipe);

        perror("execvp");
    }
}
else
{
    waitpid(pid, NULL, 0);
}
}
return 1;
}
```

1.Lặp:

Chương trình chạy trong vòng lặp while (should_run) để tiếp tục chạy shell cho đến khi người dùng nhập lệnh "exit\n".

2.Nhập Lệnh và Phân Tách Đối số:

Người dùng nhập lệnh vào biến input.

Nếu lệnh nhập là "exit\n", chương trình dừng.

Lệnh được phân tách thành các đối số trong mảng args.

3.Kiểm tra Dấu | (Pipe):

Chương trình kiểm tra xem lệnh nhập vào có chứa dấu "|" không.

4.Xử lý Dấu | (Pipe):

Nếu có dấu "|":

Tạo một pipe bằng cách sử dụng pipe() để tạo một cặp đầu vào/đầu ra.

Tạo hai mảng front_pipe và back_pipe để chứa các phần của lệnh trước và sau dấu "|".

Tạo một tiến trình con (child process) bằng fork() để thực hiện lệnh trước dấu "|".

Trong tiến trình con này:

Đóng đầu đọc của pipe sử dụng close(pipefd[0]).

Sử dụng dup2() để đặt đầu ra chuẩn của tiến trình này là đầu vào của pipe, thông qua dup2(pipefd[1], STDOUT_FILENO).

Thực thi lệnh trước dấu "|" bằng execvp().

Tiếp theo, trong tiến trình cha:

Đóng đầu ghi của pipe close(pipefd[1]).

Sử dụng dup2() để đặt đầu vào chuẩn của tiến trình này là đầu ra của pipe, thông qua dup2(pipefd[0], STDIN_FILENO).

Thực thi lệnh sau dấu "|" bằng execvp().

5.Chờ Tiến trình Con Kết thúc:

Quá trình cha chờ tiến trình con kết thúc bằng waitpid(pid, NULL, 0).

6.Lặp Lại Quá trình:

Sau khi tiến trình con hoàn thành và quá trình cha chờ đợi, vòng lặp tiếp tục, chờ đợi lệnh nhập tiếp theo từ người dùng.

+ kết quả:

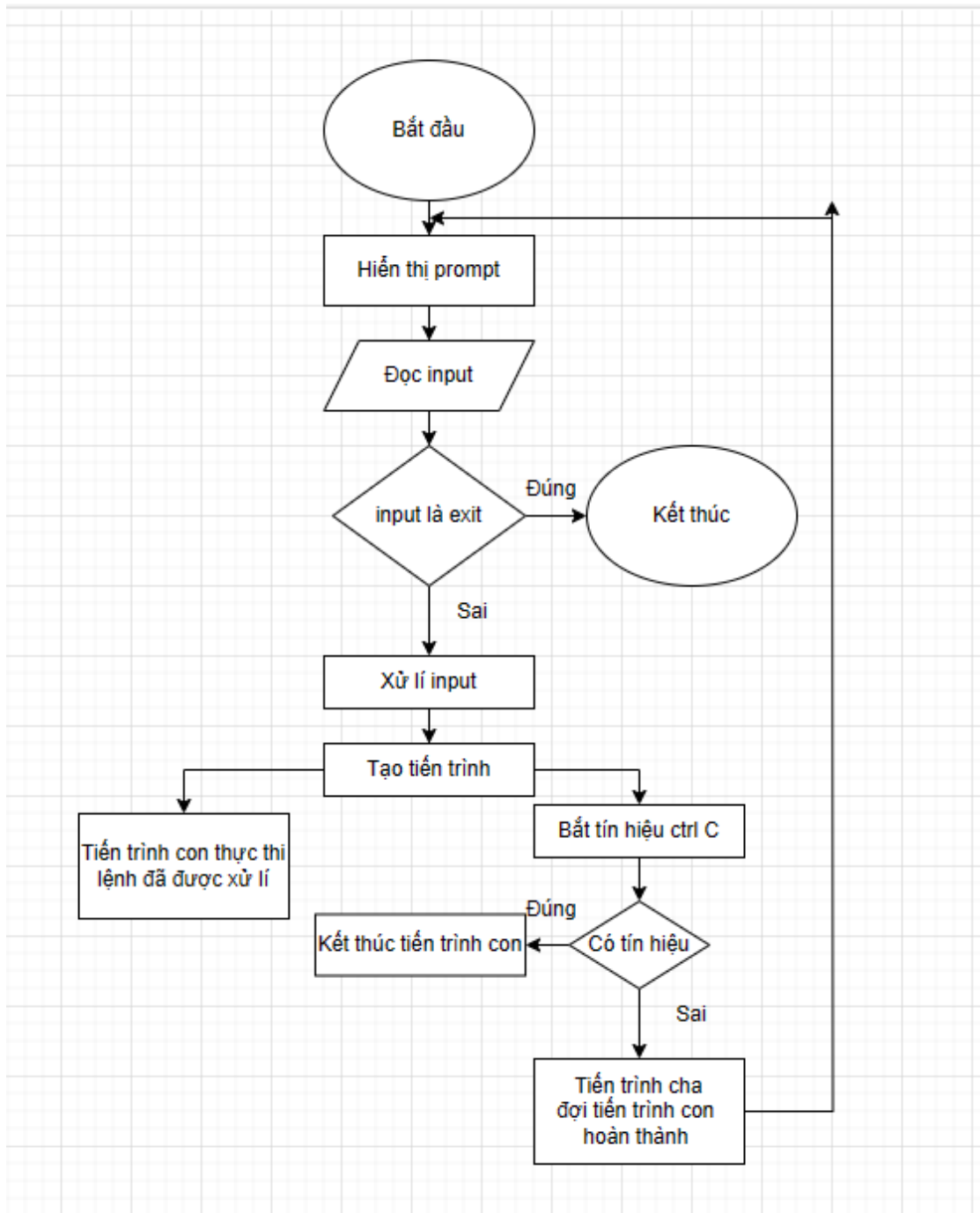
Báo cáo thực hành môn Hệ điều hành - Giảng viên: Trần Hoàng Lộc.

Lệnh ps aux để liệt kê tất cả các tiến trình đang chạy trên hệ thống với thông tin chi tiết về từng tiến trình. Trong khi đó, lệnh grep firefox được sử dụng để lọc kết quả từ đầu ra của lệnh ps aux và chỉ hiển thị các dòng chứa từ "firefox".

Tương tự với ps aux | grep chrome và ps aux | grep edge.

5. Câu 5

- **Trình bày giải thuật :**



- Chụp hình minh chứng :

Lệnh top :

Báo cáo thực hành môn Hệ điều hành - Giảng viên: Trần Hoàng Lộc.

```
C 4.c C 5.c x
Lab6 > C 5.c > main(void)

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Tasks: 20 total, 1 running, 18 sleeping, 1 stopped, 0 zombie
%Cpu(s): 1.6 us, 1.6 sy, 0.0 ni, 95.3 id, 0.0 wa, 0.0 hi, 1.6 si, 0.0 st
MiB Mem : 12689.3 total, 11927.2 free, 355.8 used, 406.3 buff/cache
MiB Swap: 4096.0 total, 4096.0 free, 0.0 used, 12084.6 avail Mem

PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND
1 root 20 0 928 544 476 S 0.0 0.0 0:00.23 init
17 root 20 0 1276 372 20 S 0.0 0.0 0:00.00 init
18 root 20 0 1276 372 20 S 0.0 0.0 0:00.03 init
19 nguyenn+ 20 0 6208 5052 3324 S 0.0 0.0 0:00.12 bash
53 root 20 0 15420 5324 3692 S 0.0 0.0 0:00.00 sshd
128 nguyenn+ 20 0 2884 996 900 S 0.0 0.0 0:00.00 sh
138 nguyenn+ 20 0 980932 118340 40812 S 0.0 0.9 0:28.46 node
285 nguyenn+ 20 0 721956 66292 38496 S 0.0 0.5 0:07.42 node
327 root 20 0 16880 10632 8512 S 0.0 0.1 0:00.01 sshd
351 nguyenn+ 20 0 17320 7940 5464 S 0.0 0.1 0:03.02 sshd
352 nguyenn+ 20 0 5040 3636 3148 S 0.0 0.0 0:00.03 bash
406 nguyenn+ 20 0 1022660 154924 40488 S 0.0 1.2 0:22.66 node
417 nguyenn+ 20 0 848816 55464 38088 S 0.0 0.4 0:05.01 node
488 nguyenn+ 20 0 89012 41348 15252 S 0.0 0.3 0:12.81 cpptools
562 nguyenn+ 20 0 6228 5384 3576 S 0.0 0.0 0:00.18 bash
1009 nguyenn+ 20 0 4256908 14400 9412 S 0.0 0.1 0:00.24 cpptools-srv
1061 nguyenn+ 20 0 2768 1520 1420 T 0.0 0.0 0:00.00 5
2435 nguyenn+ 20 0 3204 1004 916 S 0.0 0.0 0:00.00 sleep
2628 nguyenn+ 20 0 2768 1012 920 S 0.0 0.0 0:00.00 5
2865 nguyenn+ 20 0 7780 3664 3060 R 0.0 0.0 0:00.00 top

it007sh>
```

Lệnh htop :

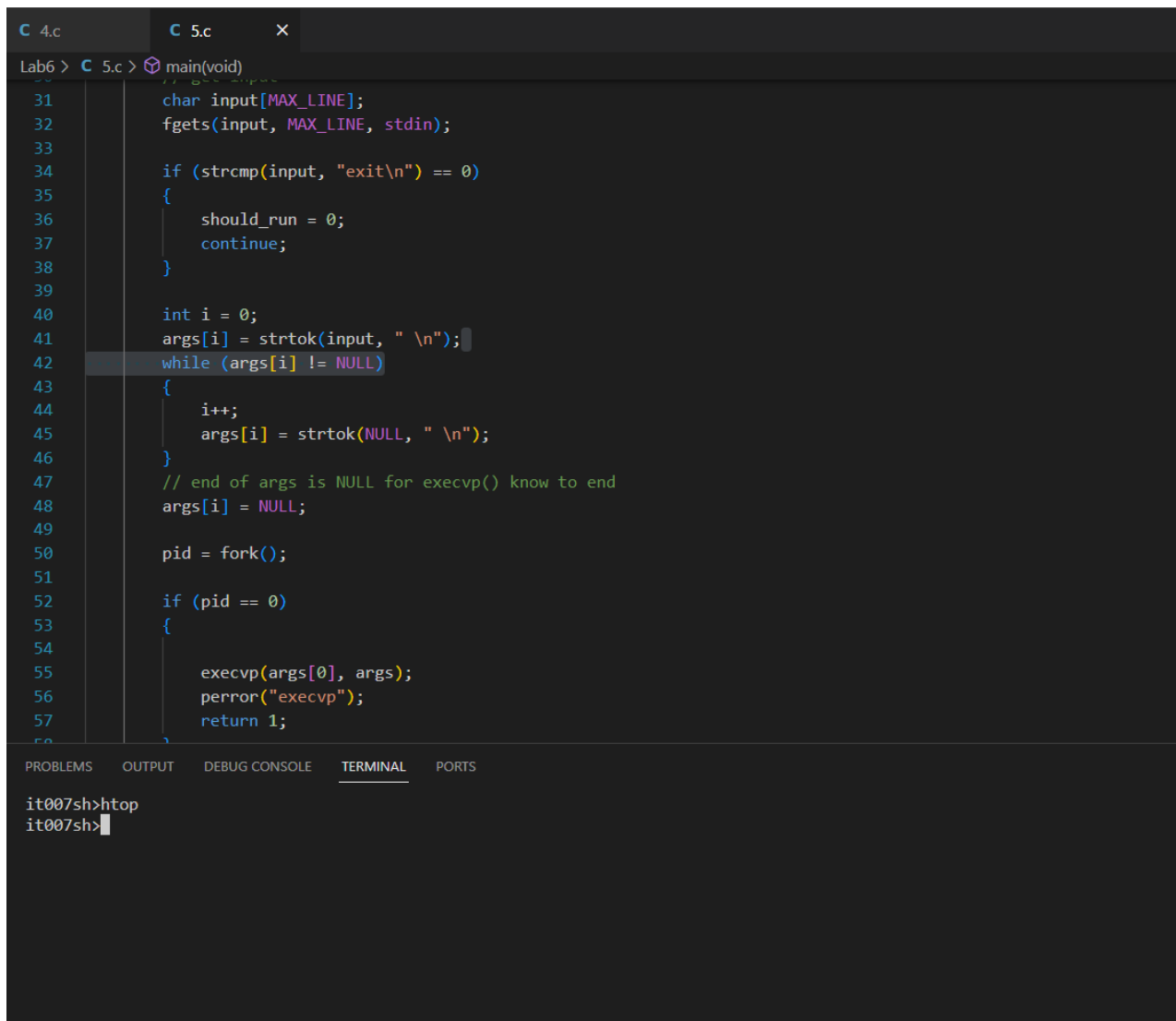
Báo cáo thực hành môn Hệ điều hành - Giảng viên: Trần Hoàng Lộc.

```
Lab6 > C 5.c > main(void)

0[ | 2.6% Tasks: 20, 69 thr; 1 running
1[ 0.0% Load average: 0.11 0.06 0.07
2[ 1.3% Uptime: 00:21:23
3[ 1.3%
Mem[ ||||| 357M/12.4G
Swp[ 0K/4.00G

PID USER PRI NI VIRT RES SHR S CPU% MEM% TIME+ Command
1 root 20 0 928 544 476 S 0.0 0.0 0:00.23 /init
7 root 20 0 928 544 476 S 0.0 0.0 0:00.00 /init
17 root 20 0 1276 372 20 S 0.0 0.0 0:00.00 /init
18 root 20 0 1276 372 20 S 0.0 0.0 0:00.03 /init
19 nguyenngu 20 0 6208 5052 3324 S 0.0 0.0 0:00.12 -bash
53 root 20 0 15420 5324 3692 S 0.0 0.0 0:00.00 sshd: /usr/sbin/sshd [listener] 0 of 10-100 startups
128 nguyenngu 20 0 2884 996 900 S 0.0 0.0 0:00.00 sh /home/nguyennguyenngocanh-22520058/.vscode-server/bin/0ee08df0cf4527e40edc9aa28f4b5bd38bbff2b2/bin/co
138 nguyenngu 20 0 958M 115M 40812 S 0.7 0.9 0:28.86 /home/nguyennguyenngocanh-22520058/.vscode-server/bin/0ee08df0cf4527e40edc9aa28f4b5bd38bbff2b2/node /hom
175 nguyenngu 20 0 958M 115M 40812 S 0.0 0.9 0:00.00 /home/nguyennguyenngocanh-22520058/.vscode-server/bin/0ee08df0cf4527e40edc9aa28f4b5bd38bbff2b2/node /hom
176 nguyenngu 20 0 958M 115M 40812 S 0.0 0.9 0:00.98 /home/nguyennguyenngocanh-22520058/.vscode-server/bin/0ee08df0cf4527e40edc9aa28f4b5bd38bbff2b2/node /hom
177 nguyenngu 20 0 958M 115M 40812 S 0.0 0.9 0:00.91 /home/nguyennguyenngocanh-22520058/.vscode-server/bin/0ee08df0cf4527e40edc9aa28f4b5bd38bbff2b2/node /hom
178 nguyenngu 20 0 958M 115M 40812 S 0.0 0.9 0:01.06 /home/nguyennguyenngocanh-22520058/.vscode-server/bin/0ee08df0cf4527e40edc9aa28f4b5bd38bbff2b2/node /hom
179 nguyenngu 20 0 958M 115M 40812 S 0.0 0.9 0:00.95 /home/nguyennguyenngocanh-22520058/.vscode-server/bin/0ee08df0cf4527e40edc9aa28f4b5bd38bbff2b2/node /hom
204 nguyenngu 20 0 958M 115M 40812 S 0.0 0.9 0:00.00 /home/nguyennguyenngocanh-22520058/.vscode-server/bin/0ee08df0cf4527e40edc9aa28f4b5bd38bbff2b2/node /hom
248 nguyenngu 20 0 958M 115M 40812 S 0.0 0.9 0:01.65 /home/nguyennguyenngocanh-22520058/.vscode-server/bin/0ee08df0cf4527e40edc9aa28f4b5bd38bbff2b2/node /hom
249 nguyenngu 20 0 958M 115M 40812 S 0.0 0.9 0:01.91 /home/nguyennguyenngocanh-22520058/.vscode-server/bin/0ee08df0cf4527e40edc9aa28f4b5bd38bbff2b2/node /hom
250 nguyenngu 20 0 958M 115M 40812 S 0.0 0.9 0:01.69 /home/nguyennguyenngocanh-22520058/.vscode-server/bin/0ee08df0cf4527e40edc9aa28f4b5bd38bbff2b2/node /hom
251 nguyenngu 20 0 958M 115M 40812 S 0.0 0.9 0:01.73 /home/nguyennguyenngocanh-22520058/.vscode-server/bin/0ee08df0cf4527e40edc9aa28f4b5bd38bbff2b2/node /hom
285 nguyenngu 20 0 705M 66772 38496 S 0.7 0.5 0:07.65 /home/nguyennguyenngocanh-22520058/.vscode-server/bin/0ee08df0cf4527e40edc9aa28f4b5bd38bbff2b2/node /hom
286 nguyenngu 20 0 705M 66772 38496 S 0.0 0.5 0:00.00 /home/nguyennguyenngocanh-22520058/.vscode-server/bin/0ee08df0cf4527e40edc9aa28f4b5bd38bbff2b2/node /hom
287 nguyenngu 20 0 705M 66772 38496 S 0.0 0.5 0:00.29 /home/nguyennguyenngocanh-22520058/.vscode-server/bin/0ee08df0cf4527e40edc9aa28f4b5bd38bbff2b2/node /hom
288 nguyenngu 20 0 705M 66772 38496 S 0.0 0.5 0:00.29 /home/nguyennguyenngocanh-22520058/.vscode-server/bin/0ee08df0cf4527e40edc9aa28f4b5bd38bbff2b2/node /hom
289 nguyenngu 20 0 705M 66772 38496 S 0.0 0.5 0:00.41 /home/nguyennguyenngocanh-22520058/.vscode-server/bin/0ee08df0cf4527e40edc9aa28f4b5bd38bbff2b2/node /hom
290 nguyenngu 20 0 705M 66772 38496 S 0.0 0.5 0:00.19 /home/nguyennguyenngocanh-22520058/.vscode-server/bin/0ee08df0cf4527e40edc9aa28f4b5bd38bbff2b2/node /hom
291 nguyenngu 20 0 705M 66772 38496 S 0.0 0.5 0:00.00 /home/nguyennguyenngocanh-22520058/.vscode-server/bin/0ee08df0cf4527e40edc9aa28f4b5bd38bbff2b2/node /hom
292 nguyenngu 20 0 705M 66772 38496 S 0.0 0.5 0:00.00 /home/nguyennguyenngocanh-22520058/.vscode-server/bin/0ee08df0cf4527e40edc9aa28f4b5bd38bbff2b2/node /hom
293 nguyenngu 20 0 705M 66772 38496 S 0.0 0.5 0:00.00 /home/nguyennguyenngocanh-22520058/.vscode-server/bin/0ee08df0cf4527e40edc9aa28f4b5bd38bbff2b2/node /hom
294 nguyenngu 20 0 705M 66772 38496 S 0.0 0.5 0:00.00 /home/nguyennguyenngocanh-22520058/.vscode-server/bin/0ee08df0cf4527e40edc9aa28f4b5bd38bbff2b2/node /hom
295 nguyenngu 20 0 705M 66772 38496 S 0.0 0.5 0:00.00 /home/nguyennguyenngocanh-22520058/.vscode-server/bin/0ee08df0cf4527e40edc9aa28f4b5bd38bbff2b2/node /hom
327 root 20 0 16880 10632 8512 S 0.0 0.1 0:00.01 sshd: nguyennguyenngocanh-22520058 [priv]
351 nguyenngu 20 0 17320 7940 5464 S 0.0 0.1 0:03.09 sshd: nguyennguyenngocanh-22520058@notty
352 nguyenngu 20 0 5040 3636 3148 S 0.0 0.0 0:00.03 bash

F1:help F2:Setup F3:Search F4:Filter F5:Tree F6:SortBy F7:Nice F8:Nice + F9:Kill F10:Quit
```



```
C 4.c C 5.c X
Lab6 > C 5.c > main(void)
31 char input[MAX_LINE];
32 fgets(input, MAX_LINE, stdin);
33
34 if (strcmp(input, "exit\n") == 0)
35 {
36     should_run = 0;
37     continue;
38 }
39
40 int i = 0;
41 args[i] = strtok(input, " \n");
42 while (args[i] != NULL)
43 {
44     i++;
45     args[i] = strtok(NULL, " \n");
46 }
47 // end of args is NULL for execvp() know to end
48 args[i] = NULL;
49
50 pid = fork();
51
52 if (pid == 0)
53 {
54     execvp(args[0], args);
55     perror("execvp");
56     return 1;
57 }
```

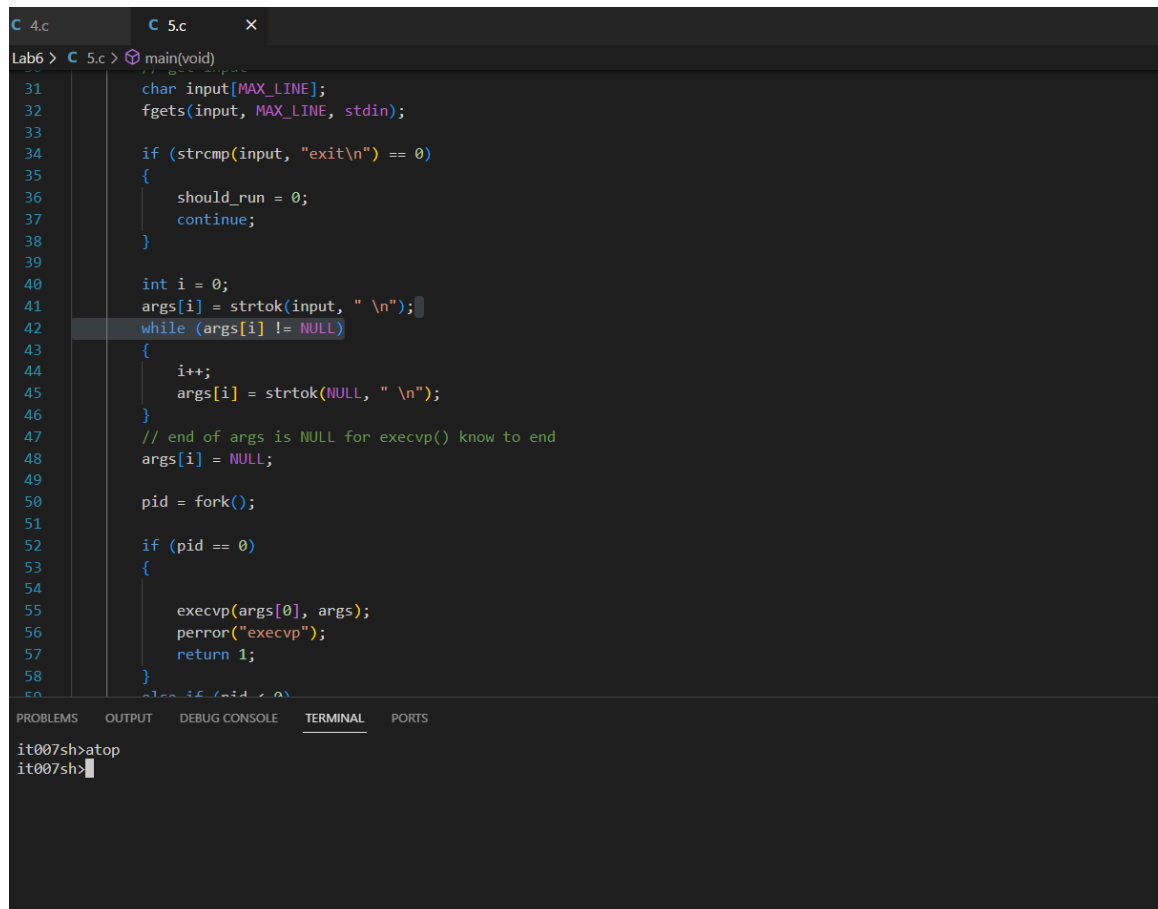
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
it007sh>htop
it007sh>
```

Lệnh atop :

Báo cáo thực hành môn Hệ điều hành - Giảng viên: Trần Hoàng Lộc.

```
C 4.c C 5.c X
Lab6 > C 5.c > main(void)
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
ATOP - DESKTOP-7R6M1IN 2023/12/26 23:57:15 ----- 23m24s elapsed
PRC sys 24.72s user 59.00s #proc 20 #tslpi 86 #tslpu 0 #zombie 0 clones 3450 no procacct
CPU sys 2% user 5% irq 1% idle 389% wait 3% steal 0% guest 0% curf 2.81GHz
cpu sys 1% user 1% irq 0% idle 96% cpu000 w 2% steal 0% guest 0% curf 2.81GHz
cpu sys 1% user 1% irq 0% idle 98% cpu001 w 0% steal 0% guest 0% curf 2.81GHz
cpu sys 1% user 1% irq 0% idle 97% cpu002 w 1% steal 0% guest 0% curf 2.81GHz
cpu sys 1% user 1% irq 0% idle 98% cpu003 w 0% steal 0% guest 0% curf 2.81GHz
CPL avgl 0.01 avg5 0.04 avg15 0.05 cs w 444445 intr 152371 numcpu numnode 0
MEM tot 12.4G free 11.6G cache 321.8M dirty 0.0M buff 43.8M slab 62.1M slrec 41.0M shmem 0.1M shrss 0.0M vmcom 609.5M vmlin 10.2G
SWP tot 4.0G free 4.0G swac 0.0M
PAG scan 0 steal 0 stall 0 compact 0 numamig 0 migrate 38e3 swin 0 swout 0 oomkill 0
DSK sdb busy 4% read 13275 write 3167 discrd 289 KiB/r 26 KiB/w 69 MB/s 0.2 MBw/s 0.2 avio 3.77 ms
DSK sda busy 1% read 57 write 915 discrd 0 KiB/r 3 KiB/w 326 MB/s 0.0 MBw/s 0.2 avio 18.8 ms
NET transport tcpip 29434 tcpo 33005 udpip 82 udpo 82 tcpao 65 tcppo 11 tcprs 23 tcpie 0 udpie 0
NET network ipip 29617 ipi 31210 ipfrw 0 deliv 29516 icmpi 0 icmpo 14
NET eth0 0% pcki 9755 pcko 11313 sp 10 Gbps si 14 Kbps so 27 Kbps erri 0 erro 0 drpi 0 drpo 0
NET lo ---- pcki 19898 pcko 19898 sp 0 Mbps si 31 Kbps so 31 Kbps erri 0 erro 0 drpi 0 drpo 0
Window resized to 167x42...
PID SYSCPU USRCPU RDElay VGROW RGROW RUID EUID ST EXC THR S CPUNR CPU CMD 1/1
138 9.40s 20.14s 3.58s 958.7M 116.0M nguyennng nguyennng N- - 11 S 0 2% node
406 6.00s 17.62s 0.79s 998.7M 151.5M nguyennng nguyennng N- - 12 S 1 2% node
488 3.24s 9.69s 0.06s 86.9M 40.4M nguyennng nguyennng N- - 14 S 0 1% cpptools
285 2.51s 6.00s 0.26s 705.5M 66.8M nguyennng nguyennng N- - 12 R 2 1% node
417 0.79s 4.24s 0.26s 828.9M 54.2M nguyennng nguyennng N- - 13 S 2 0% node
351 2.21s 1.01s 0.12s 16.9M 7.8M nguyennng nguyennng N- - 1 S 2 0% sshd
1009 0.12s 0.14s 0.00s 4.1G 14.1M nguyennng nguyennng N- - 12 S 3 0% cpptools-srv
1 0.23s 0.00s 0.00s 928.0K 544.0K root root N- - 2 S 3 0% init
562 0.07s 0.07s 0.00s 6.1M 5.3M nguyennng nguyennng N- - 1 S 2 0% bash
19 0.04s 0.08s 0.00s 6.1M 4.9M nguyennng nguyennng N- - 1 S 0 0% bash
352 0.02s 0.01s 0.00s 4.9M 3.6M nguyennng nguyennng N- - 1 S 0 0% bash
18 0.03s 0.00s 0.00s 1.2M 372.0K root root N- - 1 S 1 0% init
327 0.01s 0.00s 0.00s 16.5M 10.4M root root N- - 1 S 0 0% sshd
3274 0.01s 0.00s 0.00s 8.4M 3.9M nguyennng nguyennng N- - 1 R 3 0% atop
53 0.00s 0.00s 0.00s 15.1M 5.2M root root N- - 1 S 1 0% sshd
1061 0.00s 0.00s 0.00s 2.7M 1.5M nguyennng nguyennng N- - 1 T 2 0% 5
2628 0.00s 0.00s 0.00s 2.7M 1.4M nguyennng nguyennng N- - 1 S 1 0% 5
3114 0.00s 0.00s 0.00s 3.1M 1.0M nguyennng nguyennng N- - 1 S 2 0% sleep
128 0.00s 0.00s 0.00s 2.8M 996.0K nguyennng nguyennng N- - 1 S 2 0% sh
17 0.00s 0.00s 0.00s 1.2M 372.0K root root N- - 1 S 0 0% init
```



```
Lab6 > C 5.c > main(void)
31     char input[MAX_LINE];
32     fgets(input, MAX_LINE, stdin);
33
34     if (strcmp(input, "exit\n") == 0)
35     {
36         should_run = 0;
37         continue;
38     }
39
40     int i = 0;
41     args[i] = strtok(input, " \\n");
42     while (args[i] != NULL)
43     {
44         i++;
45         args[i] = strtok(NULL, " \\n");
46     }
47     // end of args is NULL for execvp() know to end
48     args[i] = NULL;
49
50     pid = fork();
51
52     if (pid == 0)
53     {
54
55         execvp(args[0], args);
56         perror("execvp");
57         return 1;
58     }
59     else if (pid < 0)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
it007sh>atop
it007sh>
```

- Giải thích code, kết quả :

+ Code :

```
#include <stdio.h>
#include <unistd.h>
#include <string.h>
#include <sys/wait.h>
#include <fcntl.h>
#include <stdbool.h>
#include <stdlib.h>
#include <ctype.h>
#include <signal.h>
#define MAX_LINE 80 // maximum length command
```



```
pid_t pid;

void handle_sigint(int sig)
{
    kill(pid, SIGINT);
}

int main(void)
{
    char *args[MAX_LINE / 2 + 1];
    int should_run = 1;

    while (should_run)
    {
        printf("it007sh>");
        fflush(stdout);

        // get input
        char input[MAX_LINE];
        fgets(input, MAX_LINE, stdin);

        if (strcmp(input, "exit\n") == 0)
        {
            should_run = 0;
            continue;
        }
    }
}
```

```
int i = 0;
args[i] = strtok(input, " \n");
while (args[i] != NULL)
{
    i++;
    args[i] = strtok(NULL, " \n");
}
// end of args is NULL for execvp() know to end
args[i] = NULL;

pid = fork();

if (pid == 0)
{

    execvp(args[0], args);
    perror("execvp");
    return 1;
}
else if (pid < 0)
{
    perror("Fork err!");
}
else
{
    signal(SIGINT, handle_sigint);
    waitpid(pid, NULL, 0);
}
```

```
}  
  
    return 1;  
  
}
```

1. Khởi tạo :

args: Mảng các chuỗi chứa các đối số được nhập từ bàn phím.

should_run: Biến kiểm soát vòng lặp, quyết định khi nào chương trình kết thúc.

2. Vòng lặp chính (while (should_run) { ... }):

Hiển thị dấu nhắc lệnh: it007sh>.

Nhận input từ người dùng bằng cách sử dụng fgets.

Kiểm tra xem người dùng có muốn thoát chương trình bằng lệnh "exit\n" không.

Nếu có, should_run sẽ được đặt thành 0 để kết thúc vòng lặp và thoát chương trình.

3. Tách input thành các đối số (args):

Sử dụng strtok để tách input thành các từ riêng biệt và lưu vào mảng args.

4. Tạo một child process để thực hiện lệnh nhập từ người dùng:

Sử dụng fork() để tạo một child process mới.

Trong child process, sử dụng execvp để thực hiện lệnh được nhập từ người dùng, dựa vào args. Hàm execvp sẽ thay thế tiến trình hiện tại bằng một tiến trình mới tương ứng với lệnh nhập từ người dùng.

5. Dời cho child process kết thúc:

Trong parent process, sử dụng waitpid để chờ cho child process kết thúc. Khi child process kết thúc, parent process tiếp tục vòng lặp để tiếp tục nhận input mới từ người dùng.

6. Xử lý tín hiệu SIGINT (Ctrl+C):

Sử dụng signal(SIGINT, handle_sigint) để gán signal handler handle_sigint cho tín hiệu SIGINT. Khi người dùng ấn Ctrl+C, signal handler này sẽ gửi một tín hiệu SIGINT đến child process hiện đang chạy thông qua hàm kill(pid, SIGINT).

7. Lặp Lại Quá trình:

Sau khi tiến trình con hoàn thành và quá trình cha chờ đợi, vòng lặp tiếp tục, chờ đợi lệnh nhập tiếp theo từ người dùng.

+ kết quả:

Khi thực thi lệnh top , sẽ xuất hiện 1 bảng thông tin về các tài nguyên hệ thống sau đó ta nhấn tổ hợp phím Ctrl + C để thoát tiến trình con đang chạy và không thoát khỏi chương trình C.

Tương tự với lệnh htop, atop

lệnh htop là lệnh top với phiên bản có giao diện đồ họa dễ nhìn hơn

lệnh atop khi chạy sẽ hiển thị thông tin về tình trạng hoạt động hiện tại mà còn lưu trữ dữ liệu lịch sử về tài nguyên hệ thống