

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

Họ và tên: Phạm Nguyên Anh, Phan Hoàng Mạnh Cường , Phạm Đăng Hoàng

Mã số sinh viên: 22520069, 22520179, 22520472

Lớp: IT007.O15.1

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	<input checked="" type="checkbox"/>				
Chụp hình minh chứng (chạy ít nhất 3 lệnh)	<input checked="" type="checkbox"/>				
Giải thích code, kết quả	<input checked="" type="checkbox"/>				

Tư chấm điểm: 10

*Lưu ý: Xuất báo cáo theo định dạng PDF, đặt tên theo cú pháp:

<Tên nhóm>_LAB6.pdf

6.4. BÀI TẬP THỰC HÀNH

1. Câu 1 và Câu 2 :

1. Thực thi command trong tiến trình con.
2. Tạo tính năng sử dụng lại các câu lệnh trong quá khứ

Code :

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

#define MAX_HISTORY_SIZE 10
#define MAX_INPUT_SIZE 100

char history[MAX_HISTORY_SIZE][MAX_INPUT_SIZE];
int historyIndex = 0;

void execute_command(char *command)
{
    strncpy(history[historyIndex % MAX_HISTORY_SIZE], command, MAX_INPUT_SIZE - 1);
    historyIndex++;
    pid_t pid = fork();

    if (pid < 0)
    {
        perror("Fork failed");
    }
    else if (pid == 0)
    {
        char *args[MAX_INPUT_SIZE];
        int i = 0;
        char *token = strtok(command, " ");
        while (token != NULL)
        {
            args[i++] = token;
            token = strtok(NULL, " ");
        }
        args[i] = NULL;
        execvp(args[0], args);
    }
}
```

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

void display_history ()
{
    printf("Command history:\n");
    int i;
    for (i = 0; i < MAX_HISTORY_SIZE; i++)
    {
        if (strlen(history[i]) > 0)
        {
            printf("%d. %s\n", i + 1, history[i]);
        }
    }
}
void print_prompt()
{
    printf("\nit007sh> ");
}

int main()
{
    char input[MAX_INPUT_SIZE];

    while (1)
    {
        print_prompt();
        fgets(input, MAX_INPUT_SIZE, stdin);
        input[strcspn(input, "\n")] = '\0';

        if (strcmp(input, "HF") == 0)
        {
            display_history();
        }
        else
        {
            execute_command(input);
        }
    }
    return 0;
}
```

Trình bày giải thuật :

- Tạo hàm execute_command để thực hiện lệnh được nhập.
- Tạo hàm display_history để hiển thị lịch sử các câu lệnh được nhập.
- Tạo hàm print_prompt để in ra dấu nhắc.
- Hàm main trong vòng lặp while(1) thực hiện các câu lệnh :
 - Nhập lệnh.
 - Nếu “HF” được nhập thì sẽ in lịch sử.
 - Nếu lệnh khác được nhập thì sẽ thực hiện lệnh đó.

The screenshot shows a terminal window with the following content:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

it007sh> cat abc.txt
Hello IT007
it007sh> HF
Command history:
1. cat abc.txt

it007sh> ls
22520069 abc.txt b c Desktop Downloads hello.c LAB3 lab6 myfile Myweb Public ssh test Videos
a a.c b.c c.c Documents hello hello.h LAB5 main.c Music Pictures snap Templates test.c Xin

it007sh> HF
Command history:
1. cat abc.txt
2. ls

it007sh> pwd
/home/nguyenanh_22520069

it007sh> HF
Command history:
1. cat abc.txt
2. ls
3. pwd

it007sh>
```

Giải thích code, kết quả :

- *execute_command(char command):
 - Thực hiện lệnh nhập từ người dùng.
 - Lưu trữ lệnh vào mảng lịch sử.
 - Sử dụng fork để tạo một tiến trình con.
 - Trong tiến trình con, sử dụng execvp để thực hiện lệnh nhập từ người dùng.
 - Trong tiến trình cha, sử dụng waitpid để đợi tiến trình con kết thúc.
- display_history():
 - Hiển thị lịch sử các lệnh đã nhập.
- print_prompt():
 - In ra dấu nhắc để người dùng nhập lệnh.
- main():
 - Sử dụng vòng lặp vô hạn để liên tục đọc và thực thi lệnh từ người dùng.

- Nếu lệnh là "HF" (history), hiển thị lịch sử bằng cách gọi hàm displayHistory().
- Ngược lại, thực hiện lệnh bằng cách gọi hàm execute_command(input).

2. Câu 3 và Câu 4

Câu 3: *Chuyển hướng vào ra*

Code:

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

#define BUFFER_SIZE 1024

int main() {
    char cmd[BUFFER_SIZE];
    char *cmd_tokens[10];
    pid_t pid;

    while (1) {
        printf("it007sh>");
        fgets(cmd, BUFFER_SIZE, stdin);
        cmd[strlen(cmd) - 1] = '\0';

        // Check if the user wants to exit
        if (strcmp(cmd, "exit") == 0) {
            break;
        }

        // Tách lệnh thành các đối số
        int i = 0;
        char *token = strtok(cmd, " ");
        while (token != NULL) {
            cmd_tokens[i++] = token;
            token = strtok(NULL, " ");
        }
        cmd_tokens[i] = NULL;
    }
}
```

```
// Tìm toán tử redirection
int redirect_input = 0;
int input_index = -1;
int redirect_output = 0;
int output_index = -1;

for (int j = 0; j < i; ++j) {
    if (strcmp(cmd_tokens[j], "<") == 0) {
        redirect_input = 1;
        input_index = j;
    }
    if (strcmp(cmd_tokens[j], ">") == 0) {
        redirect_output = 1;
        output_index = j;
    }
}

// Fork child process
pid = fork();

if (pid == 0) {
    // Child process

    // Chuyển hướng đầu vào nếu có
    if (redirect_input) {
        int input_fd = open(cmd_tokens[input_index + 1], O_RDONLY);
        if (input_fd == -1) {
            perror("open");
            exit(1);
        }
        dup2(input_fd, STDIN_FILENO);
        close(input_fd);

        // Loại bỏ toán tử chuyển hướng và tên tệp từ danh sách đối số
        for (int k = input_index; k < i - 2; ++k) {
            cmd_tokens[k] = cmd_tokens[k + 2];
        }
        cmd_tokens[i - 2] = NULL;
        cmd_tokens[i - 1] = NULL;
    }

    // Chuyển hướng đầu ra nếu có
    if (redirect_output) {
```

```
        int output_fd = open(cmd_tokens[output_index + 1], O_WRONLY |  
O_CREAT | O_TRUNC, 0666);  
        if (output_fd == -1) {  
            perror("open");  
            exit(1);  
        }  
        dup2(output_fd, STDOUT_FILENO);  
        close(output_fd);  
  
        // Loại bỏ toán tử chuyển hướng và tên tệp từ danh sách đối số  
        for (int k = output_index; k < i - 2; ++k) {  
            cmd_tokens[k] = cmd_tokens[k + 2];  
        }  
        cmd_tokens[i - 2] = NULL;  
        cmd_tokens[i - 1] = NULL;  
    }  
  
    // Thực hiện Lệnh  
    execvp(cmd_tokens[0], cmd_tokens);  
    perror("execvp");  
    exit(1);  
} else {  
    // Parent process  
    wait(NULL);  
}  
}  
  
return 0;  
}
```

Trình bày giải thuật :

- Sử dụng hàm fork() để tạo một tiến trình con chạy lệnh
- Sử dụng hàm execvp() để thực thi lệnh trong tiến trình con.
- Hàm wait() được sử dụng để đảm bảo tiến trình cha không kết thúc trước tiến trình con.
- Sử dụng các hàm như open(), creat(), dup2(), và close() để thực hiện chuyển hướng đầu vào/đầu ra.

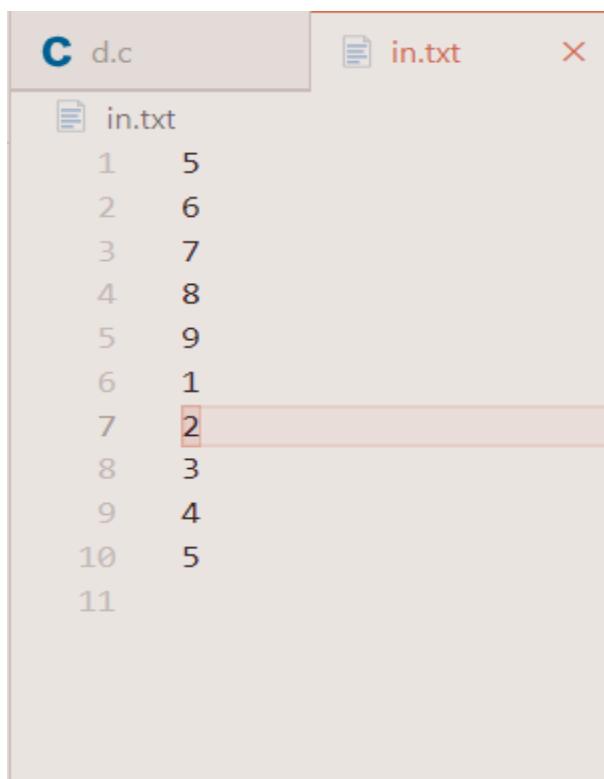
Kết quả chạy code :

The screenshot shows a terminal window with the following session:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
nguyenanh_22520069@nguyenanh22520069:~/VirtualBox:~$ gcc -o d d.c
nguyenanh_22520069@nguyenanh22520069:~/VirtualBox:~$ ./d
ite007sh>ls > output.txt
ite007sh>sort < in.txt
1
2
3
4
5
6
7
8
9
ite007sh>cat < abc.txt
abc.txt
ite007sh>echo "Hello, World!" > greeting.txt
ite007sh>
```

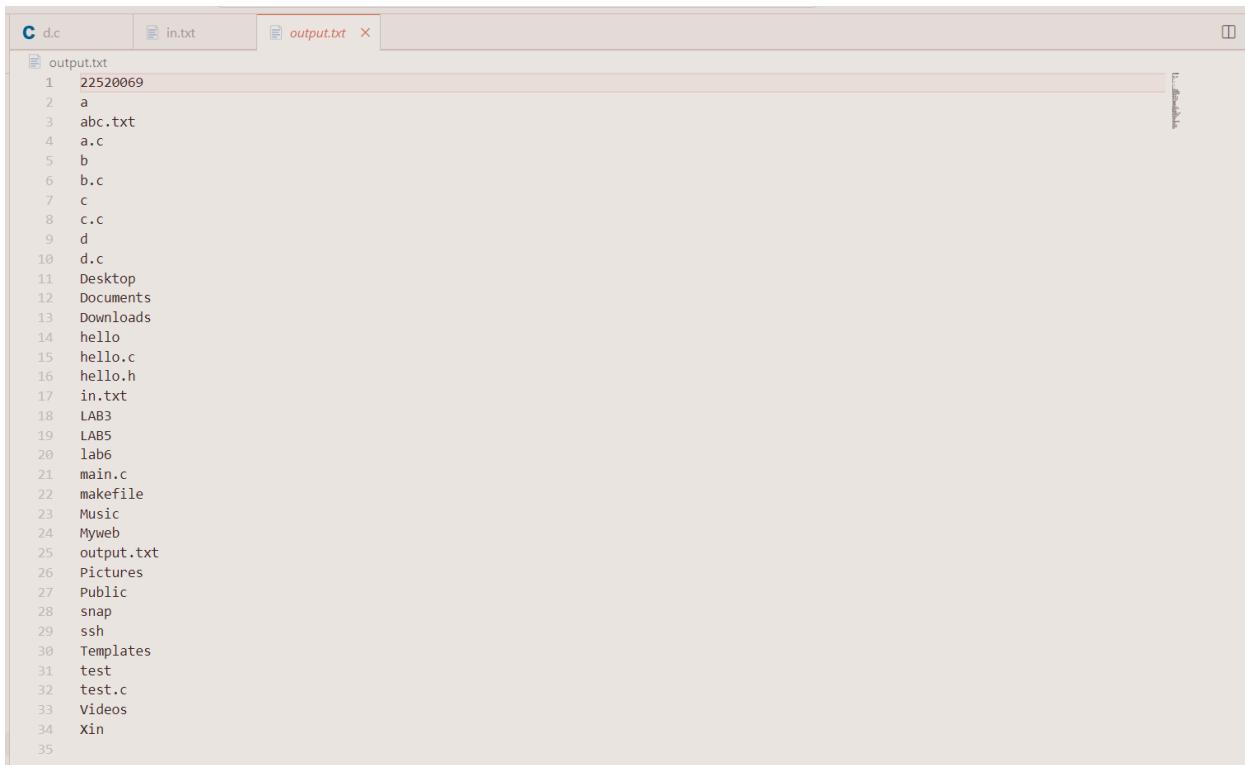
Ln 11, Col 1 Spaces: 4 UTF-8 LF Plain Text

Thực hiện lệnh ls rồi gán vào file output.txt, sort file in.txt, gán Hello, world! vào greeting.txt



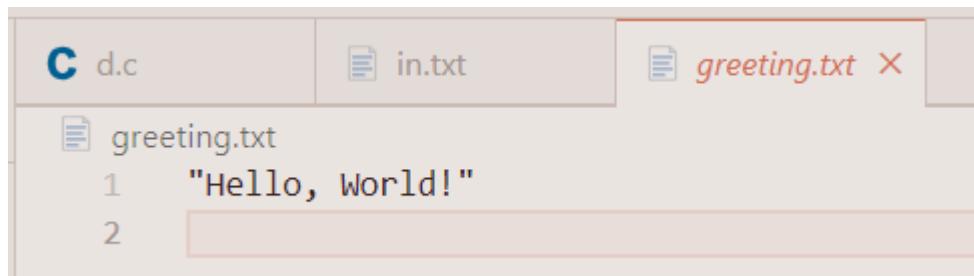
File in.txt chưa được sort.

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



```
C d.c      in.txt      output.txt X
output.txt
1 22520069
2 a
3 abc.txt
4 a.c
5 b
6 b.c
7 c
8 c.c
9 d
10 d.c
11 Desktop
12 Documents
13 Downloads
14 hello
15 hello.c
16 hello.h
17 in.txt
18 LAB3
19 LABS
20 lab6
21 main.c
22 makefile
23 Music
24 Myweb
25 output.txt
26 Pictures
27 Public
28 snap
29 ssh
30 Templates
31 test
32 test.c
33 Videos
34 Xin
35
```

File output.txt được tạo với nội dung khi thực hiện của lệnh ls



```
C d.c      in.txt      greeting.txt X
greeting.txt
1 "Hello, World!"
```

File greeting.txt được tạo với nội dung "Hello, world!"

Giải thích code, kết quả :

- Chương trình này sẽ hiển thị dấu nhắc it007sh> và chờ người dùng nhập vào một lệnh.
- Nếu người dùng nhập exit, chương trình sẽ kết thúc.
- Nếu không, chương trình sẽ tách lệnh thành các đối số bằng cách sử dụng khoảng trắng làm dấu phân cách.
- Chương trình sau đó tìm kiếm các toán tử chuyển hướng (< và >). Nếu tìm thấy, nó sẽ ghi nhớ vị trí của chúng và tệp liên quan.
- Chương trình sau đó tạo một tiến trình con bằng cách sử dụng fork().
- Trong tiến trình con:

- Nếu có toán tử chuyển hướng đầu vào (<), nó mở tệp đầu vào, chuyển hướng STDIN đến tệp này, và loại bỏ toán tử chuyển hướng và tên tệp khỏi danh sách đối số.
 - Nếu có toán tử chuyển hướng đầu ra (>), nó mở (hoặc tạo) tệp đầu ra, chuyển hướng STDOUT đến tệp này, và loại bỏ toán tử chuyển hướng và tên tệp khỏi danh sách đối số.
 - Cuối cùng, nó thực hiện lệnh bằng cách sử dụng execvp().
7. Trong tiến trình cha, nó chờ đợi tiến trình con kết thúc bằng cách sử dụng wait().

Câu 4: Giao tiếp sử dụng cơ chế đường ống

Code:

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

#define BUFFER_SIZE 1024

int main() {
    char cmd[BUFFER_SIZE];
    char *cmd_tokens[10];
    pid_t pid;
    int pipe_fd[2];

    while (1) {
        printf("it007sh>");
        fgets(cmd, BUFFER_SIZE, stdin);
        cmd[strlen(cmd) - 1] = '\0';

        if (strcmp(cmd, "exit") == 0) {
            break;
        }

        int i = 0;
        char *token = strtok(cmd, " ");
        while (token != NULL) {
            cmd_tokens[i++] = token;
            token = strtok(NULL, " ");
        }
    }
}
```

```
cmd_tokens[i] = NULL;

int pipe_index = -1;

for (int j = 0; j < i; ++j) {
    if (strcmp(cmd_tokens[j], "|") == 0) {
        pipe_index = j;
        break;
    }
}

pid = fork();

if (pid == 0) {
    if (pipe_index != -1) {
        if (pipe(pipe_fd) == -1) {
            perror("pipe");
            exit(1);
        }

        pid_t pid2 = fork();

        if (pid2 == 0) {
            close(pipe_fd[0]);

            dup2(pipe_fd[1], STDOUT_FILENO);

            cmd_tokens[pipe_index] = NULL;
            execvp(cmd_tokens[0], cmd_tokens);
            perror("execvp");
            exit(1);
        } else if (pid2 > 0) {
            close(pipe_fd[1]);

            dup2(pipe_fd[0], STDIN_FILENO);

            execvp(cmd_tokens[pipe_index + 1], &cmd_tokens[pipe_index + 1]);
            perror("execvp");
            exit(1);
        } else {
            perror("fork");
            exit(1);
        }
    }
}
```

```
        } else {
            execvp(cmd_tokens[0], cmd_tokens);
            perror("execvp");
            exit(1);
        }
    } else {
        wait(NULL);
    }
}

return 0;
}
```

Trình bày giải thuật:

- Sử dụng hàm fork() để tạo một tiến trình con chạy lệnh
- Hàm system() để thực thi lệnh trong tiến trình con
- Hàm wait() để đảm bảo tiến trình cha không kết thúc trước tiến trình con
- Sử dụng các hàm như pipe(), dup2() và close() để thực hiện chuyển hướng đầu vào/đầu ra thông qua đường ống.

Kết quả chạy code :

The screenshot shows a terminal window with the following content:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
nguyenanh_22520069@nguyenanh_22520069-VirtualBox:~/e
it007sh>ls -l | wc -l
39
it007sh>cat in.txt | wc -l
10
it007sh>ls -l | sort -k 5,5 -n -r
-rw-rw-r-x 1 nguyenanh_22520069 nguyenanh_22520069 16752 Thg 12 17 13:26 lab6
-rw-rw-r-x 1 nguyenanh_22520069 nguyenanh_22520069 16704 Thg 12 17 13:51 c
-rw-rw-r-x 1 nguyenanh_22520069 nguyenanh_22520069 16664 Thg 12 17 14:09 b
-rw-rw-r-x 1 nguyenanh_22520069 nguyenanh_22520069 16552 Thg 12 24 10:37 e
-rw-rw-r-x 1 nguyenanh_22520069 nguyenanh_22520069 16552 Thg 12 24 10:30 d
-rw-rw-r-x 1 nguyenanh_22520069 nguyenanh_22520069 16472 Thg 11 8 22:56 test
-rw-rw-r-x 1 nguyenanh_22520069 nguyenanh_22520069 16360 Thg 12 17 13:36 a
-rw-rw-r-x 1 nguyenanh_22520069 nguyenanh_22520069 15960 Thg 10 27 13:44 hello
drwxr-xr-x 2 nguyenanh_22520069 nguyenanh_22520069 4096 Thg 9 29 18:44 Desktop
drwxr-xr-x 2 nguyenanh_22520069 nguyenanh_22520069 4096 Thg 9 29 17:48 Videos
drwxr-xr-x 2 nguyenanh_22520069 nguyenanh_22520069 4096 Thg 9 29 17:48 Templates
drwxr-xr-x 2 nguyenanh_22520069 nguyenanh_22520069 4096 Thg 9 29 17:48 Public
drwxr-xr-x 2 nguyenanh_22520069 nguyenanh_22520069 4096 Thg 9 29 17:48 Pictures
drwxr-xr-x 2 nguyenanh_22520069 nguyenanh_22520069 4096 Thg 9 29 17:48 Music
drwxr-xr-x 2 nguyenanh_22520069 nguyenanh_22520069 4096 Thg 9 29 17:48 Downloads
drwxr-xr-x 2 nguyenanh_22520069 nguyenanh_22520069 4096 Thg 9 29 17:48 Documents
drwxr-xr-x 5 nguyenanh_22520069 nguyenanh_22520069 4096 Thg 9 29 19:27 Myweb
drwxr-xr-x 3 nguyenanh_22520069 nguyenanh_22520069 4096 Thg 10 17 19:49 22520069
drwxr-xr-x 2 nguyenanh_22520069 nguyenanh_22520069 4096 Thg 12 2 14:14 LAB5
drwxr-xr-x 2 nguyenanh_22520069 nguyenanh_22520069 4096 Thg 11 10 18:45 LAB3
drwxr-xr-x 2 nguyenanh_22520069 nguyenanh_22520069 4096 Thg 10 13 15:31 ssh
drwxr----- 3 nguyenanh_22520069 nguyenanh_22520069 4096 Thg 9 29 17:48 snap
-rw-rw-r-- 1 nguyenanh_22520069 nguyenanh_22520069 3008 Thg 12 24 10:27 d.c
-rw-rw-r-- 1 nguyenanh_22520069 nguyenanh_22520069 2010 Thg 12 24 10:56 e.c
-rw-rw-r-- 1 nguyenanh_22520069 nguyenanh_22520069 1625 Thg 11 8 22:56 test.c
-rw-rw-r-- 1 nguyenanh_22520069 nguyenanh_22520069 1602 Thg 12 17 14:08 b.c
-rw-rw-r-- 1 nguyenanh_22520069 nguyenanh_22520069 1568 Thg 12 17 13:51 c.c
-rw-rw-r-- 3 nguyenanh_22520069 nguyenanh_22520069 1076 Thg 12 17 13:36 a.c
-rw-rw-r-- 1 nguyenanh_22520069 nguyenanh_22520069 327 Thg 10 27 13:50 hello.h
-rw-rw-r-- 1 nguyenanh_22520069 nguyenanh_22520069 315 Thg 10 27 13:37 hello.c
-rw-rw-r-- 1 nguyenanh_22520069 nguyenanh_22520069 262 Thg 10 27 13:51 main.c
-rw-rw-r-- 1 nguyenanh_22520069 nguyenanh_22520069 210 Thg 12 24 10:30 output.txt
-rw-rw-r-- 1 nguyenanh_22520069 nguyenanh_22520069 88 Thg 10 27 14:12 makefile
-rw-rw-r-- 1 nguyenanh_22520069 nguyenanh_22520069 20 Thg 12 24 10:28 in.txt
```

Thực hiện việc đếm số lượng tệp tin trong thư mục hiện tại, đếm số dòng trong một tệp tin văn bản và hiển thị danh sách các tệp tin và thư mục, sau đó sắp xếp theo kích thước giảm dần.

Giải thích code, kết quả :

1. Chương trình này sẽ hiển thị dấu nhắc `it007sh>` và chờ người dùng nhập vào một lệnh.
2. Nếu người dùng nhập `exit`, chương trình sẽ kết thúc.
3. Nếu không, chương trình sẽ tách lệnh thành các đối số bằng cách sử dụng khoảng trắng làm dấu phân cách.
4. Chương trình sau đó tìm kiếm toán tử `|`. Nếu tìm thấy, nó sẽ ghi nhớ vị trí của nó.
5. Chương trình sau đó tạo một tiến trình con bằng cách sử dụng `fork()`.
6. Trong tiến trình con:
 - o Nếu có toán tử `|`, nó tạo một ống bằng cách sử dụng `pipe()`, sau đó tạo một tiến trình con khác.
 - Trong tiến trình con thứ hai, nó đóng đầu đọc của ống, chuyển hướng `STDOUT` đến đầu ghi của ống, và thực hiện phần lệnh trước toán tử `|`.
 - Trong tiến trình con đầu tiên, nó đóng đầu ghi của ống, chuyển hướng `STDIN` đến đầu đọc của ống, và thực hiện phần lệnh sau toán tử `|`.
 - o Nếu không có toán tử `|`, nó chỉ đơn giản thực hiện lệnh bằng cách sử dụng `execvp()`.
7. Trong tiến trình cha, nó chờ đợi tiến trình con kết thúc bằng cách sử dụng `wait()`.

3. Câu 5

Code:

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

```
C lab6.c
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <string.h>
4 #include <unistd.h>
5 #include <signal.h>
6 #include <sys/wait.h>
7
8 #define MAX_LINE 80
9
10 volatile sig_atomic_t child_running = 0;
11
12 void handle_sigchld(int sig) {
13     child_running = 0;
14 }
15
16 void handle_sigint(int sig) {
17 }
18
19
20 int main(void) {
21     char input[MAX_LINE];
22
23     signal(SIGINT, handle_sigint);
24
25     while (1) {
26         if (!child_running) {
27             signal(SIGCHLD, handle_sigchld);
28         }
29
30         printf("T1007sh> ");
31         fflush(stdout);
32
33         if (fgets(input, sizeof(input), stdin) == NULL) {
34             break;
35         }
36
37
38         input[strcspn(input, "\n")] = '\0';
39
40         if (strcmp(input, "exit") == 0) {
41             break;
42         }
43
44         pid_t pid = fork();
45
46         if (pid < 0) {
47             fprintf(stderr, "Fork failed\n");
48             return 1;
49         } else if (pid == 0) {
50
51             execvp(input, input, (char *)NULL);
52             perror("execvp");
53             exit(EXIT_FAILURE);
54         } else {
55
56             child_running = 1;
57
58             wait(NULL);
59             signal(SIGCHLD, SIG_DFL);
60         }
61     }
62
63     return 0;
64 }
65
```

Trình bày giải thuật :

- Tạo hàm handle_sigint(int sig): Xử lý tín hiệu SIGINT (Ctrl+C). Có thể mở rộng để xử lý các tác vụ khi tín hiệu này được nhận.
 - Tạo hàm handle_sigchld(int sig) và hàm sẽ được gọi khi một quy trình con kết thúc. Đặt child_running về 0 để thông báo rằng không có quy trình con đang chạy.
 - Hàm main():
 - Tạo hàm với vòng lặp vô hạn chờ lệnh
 - Hiển thị dấu nhắc "IT007sh>".
Nếu lệnh là "exit", thoát khỏi vòng lặp và kết thúc chương trình.
 - Trong tiến trình con : Sử dụng execlp() để thực hiện lệnh đã nhập từ người dùng.
- Trong tiến trình cha : Chờ tiến trình con và đặt lại xử lý tín hiệu SIGCHLD về giá trị mặc định

Kết quả chạy code :

```
Tasks: 21 total, 1 running, 20 sleeping, 0 stopped, 0 zombie
%CPU(s): 0.1 us, 0.1 sy, 0.0 ni, 99.8 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
MiB Mem : 3590.2 total, 3115.1 free, 323.3 used, 151.7 buff/cache
MiB Swap: 1024.0 total, 1024.0 free, 0.0 used, 3121.7 avail Mem

PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND
23 phamdan+ 20 0 952040 87196 40260 S 1.0 2.4 0:08.14 node
114 phamdan+ 20 0 970908 99808 39192 S 1.0 2.7 0:38.98 node
36 phamdan+ 20 0 601660 52968 36192 S 0.3 1.4 0:01.79 node
52 phamdan+ 20 0 714372 70532 38264 S 0.3 1.9 0:04.54 node
1 root 20 0 1804 1188 1104 S 0.0 0.0 0:00.01 init
11 root 20 0 1824 88 0 S 0.0 0.0 0:00.00 init
12 root 20 0 1824 104 0 S 0.0 0.0 0:00.00 init
13 phamdan+ 20 0 2884 1012 920 S 0.0 0.0 0:00.00 sh
14 phamdan+ 20 0 2884 1048 944 S 0.0 0.0 0:00.00 sh
19 phamdan+ 20 0 2884 944 848 S 0.0 0.0 0:00.00 sh
34 root 20 0 1824 88 0 S 0.0 0.0 0:00.00 init
35 root 20 0 1824 104 0 S 0.0 0.0 0:00.70 init
43 root 20 0 1824 88 0 S 0.0 0.0 0:00.00 init
44 root 20 0 1824 104 0 S 0.0 0.0 0:00.88 init
45 phamdan+ 20 0 598720 52528 36284 S 0.0 1.4 0:02.25 node
97 phamdan+ 20 0 849328 55692 37860 S 0.0 1.5 0:02.72 node
6207 phamdan+ 20 0 6232 5396 3580 S 0.0 0.1 0:00.01 bash
6242 phamdan+ 20 0 2768 1528 1428 S 0.0 0.0 0:00.00 lab6
6634 phamdan+ 20 0 6232 5300 3484 S 0.0 0.1 0:00.01 bash
6684 phamdan+ 20 0 2768 920 828 S 0.0 0.0 0:00.00 lab6
6721 phamdan+ 20 0 7780 3608 3012 R 0.0 0.1 0:00.00 top
```

```
IT007sh> ps
 PID TTY      TIME CMD
 6634 pts/4  00:00:00 bash
 6684 pts/4  00:00:00 lab6
 6876 pts/4  00:00:00 ps
IT007sh> df
Filesystem 1K-blocks      Used Available Use% Mounted on
/dev/sdb       263174212  2403292 247332764  1% /
none          1838164      4   1838160  1% /mnt/wsl
tools         171541500 142495424 29046076 84% /init
none          1838164      0   1838164  0% /run
none          1838164      0   1838164  0% /run/lock
none          1838164      0   1838164  0% /run/shm
none          1838164      0   1838164  0% /run/user
tmpfs         1838164      0   1838164  0% /sys/fs/cgroup
drivers        171541500 142495424 29046076 84% /usr/lib/wsl/drivers
lib           171541500 142495424 29046076 84% /usr/lib/wsl/lib
drvfs         171541500 142495424 29046076 84% /mnt/c
drvfs         163839996 110678728 53161268 68% /mnt/f
drvfs         163838972 101874324 61964648 63% /mnt/g
IT007sh> exit
phamdanghoang-22520472@LAPTOP-7G0T6M1L:~$
```

Giải thích kết quả:

- Chạy chương trình và sử dụng lệnh top hiển thị ra bảng thống kê các thông tin từ hệ thống
- Sử dụng lệnh ‘Ctrl+C’ lệnh thực thi sẽ kết thúc và mời người dùng
- Tương tự sử dụng lệnh ps và df hiện thị thông tin về quy trình đang chạy và không gian đĩa sử dụng/còn trống trên hệ thống.
- Dùng lệnh exit để kết thúc chương trình.
- Tiết trình con (fork): Tiết trình con sẽ thực thi lệnh được nhập từ người dùng.
- Tiết trình cha đợi tiết trình con kết thúc (wait)
- Sau khi tiết trình con kết thúc, đặt child_running về 0 để báo hiệu rằng không có tiết trình con nào đang chạy.
- Sử dụng signal để đặt xử lý tín hiệu SIGCHLD về mặc định.
- Lặp lại quá trình nhập lệnh.
- Quay lại vòng lặp để chờ người dùng nhập lệnh mới.