

Name: Nguyễn Khoa Nguyễn

ID: 24521190

Class: IT007.Q15.1

OPERATING SYSTEM LAB 6'S REPORT

SUMMARY

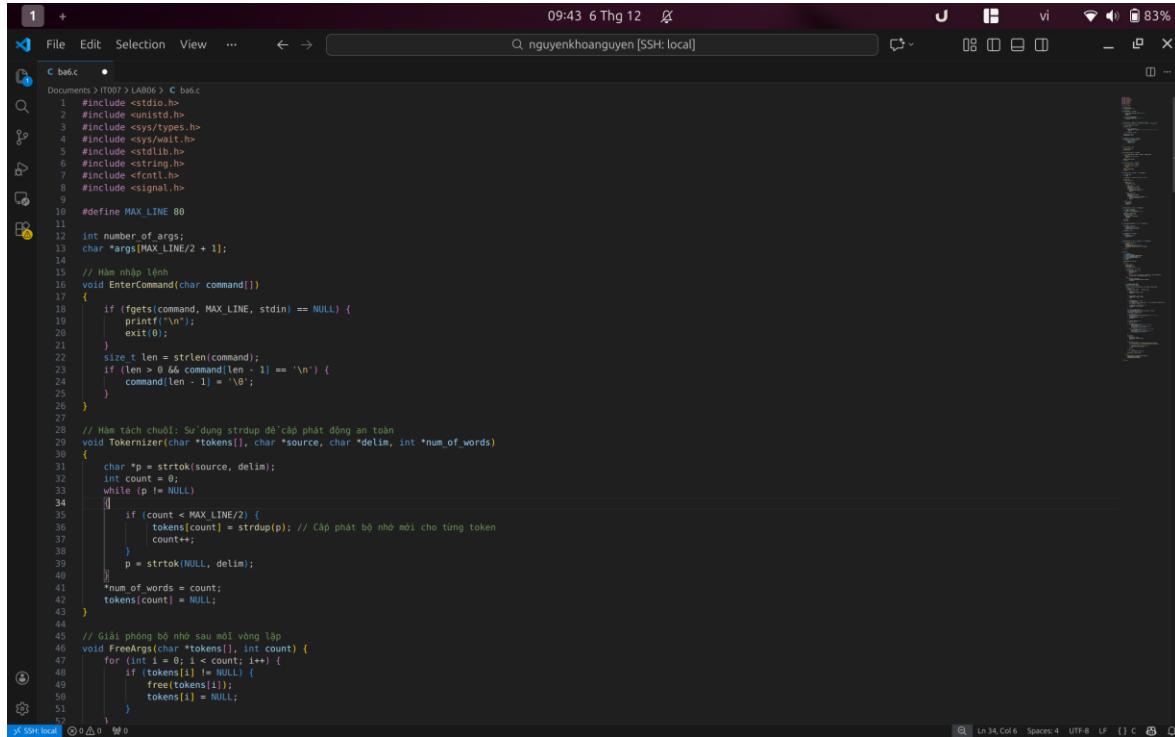
	Task	Status	Page
Section 6.4	Ex 1	Done	6
	Ex 2	Done	8
	Ex 3	Done	9
	Ex 4	Done	13
	Ex 5	Done	15

Self-scores:

*Note: Export file to **PDF** and name the file by following format:
Student ID _LABx.pdf

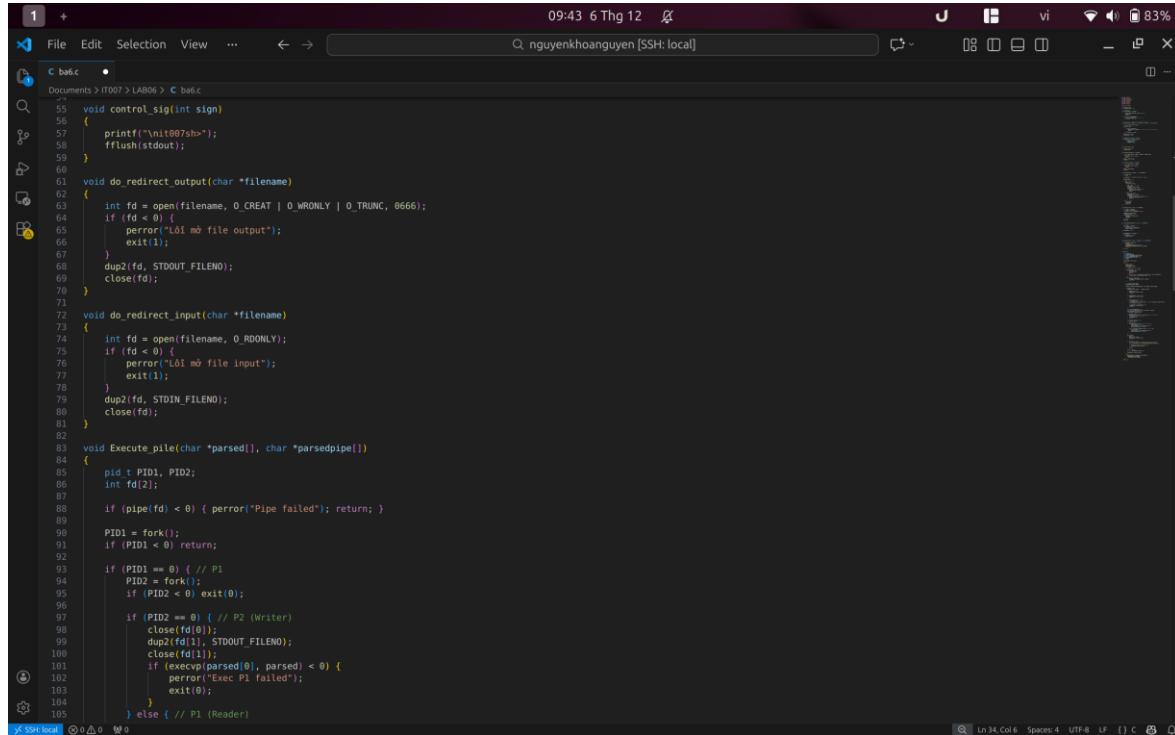
Section 6.4

1. Source code



```
C ba6.c
Documents > I007 > LAB06 > C ba6.c
1 #include <stdio.h>
2 #include <unistd.h>
3 #include <sys/types.h>
4 #include <sys/stat.h>
5 #include <stdlib.h>
6 #include <string.h>
7 #include <errno.h>
8 #include <signal.h>
9
10 #define MAX_LINE 80
11
12 int number_of_args;
13 char *args[MAX_LINE/2 + 1];
14
15 // Hàm nhập lệnh
16 void EnterCommand(char command[])
17 {
18     if (fgets(command, MAX_LINE, stdin) == NULL) {
19         printf("\n");
20         exit(0);
21     }
22     size_t len = strlen(command);
23     if (len > 0 && command[len - 1] == '\n') {
24         command[len - 1] = '\0';
25     }
26 }
27
28 // Hàm tách chuỗi: Sử dụng strdup để cấp phát động an toàn
29 void Tokenizer(char *tokens[], char *source, char *delim, int *num_of_words)
30 {
31     char *p = strtok(source, delim);
32     int count = 0;
33     while (p != NULL)
34     {
35         if (count < MAX_LINE/2) {
36             tokens[count] = strdup(p); // Cấp phát bộ nhớ mới cho từng token
37             count++;
38         }
39         p = strtok(NULL, delim);
40     }
41     *num_of_words = count;
42     tokens[count] = NULL;
43 }
44
45 // Giải phóng bộ nhớ sau mỗi vòng lặp
46 void FreeArgs(char *tokens[], int count) {
47     for (int i = 0; i < count - 1; i++) {
48         if (tokens[i] != NULL) {
49             free(tokens[i]);
50             tokens[i] = NULL;
51         }
52     }
53 }
54
55 void control_sig(int sign)
56 {
57     printf("\nit007sh~");
58     fflush(stdout);
59 }
60
61 void do_redirect_output(char *filename)
62 {
63     int fd = open(filename, O_CREAT | O_WRONLY | O_TRUNC, 0666);
64     if (fd < 0) {
65         perror("Lỗi mở file output");
66         exit(1);
67     }
68     dup2(fd, STDOUT_FILENO);
69     close(fd);
70 }
71
72 void do_redirect_input(char *filename)
73 {
74     int fd = open(filename, O_RDONLY);
75     if (fd < 0) {
76         perror("Lỗi mở file input");
77         exit(1);
78     }
79     dup2(fd, STDIN_FILENO);
80     close(fd);
81 }
82
83 void Execute_pile(char *parsed[], char *parsedpipe[])
84 {
85     pid_t PID1, PID2;
86     int fd[2];
87
88     if (pipe(fd) < 0) { perror("Pipe failed"); return; }
89
90     PID1 = fork();
91     if (PID1 < 0) return;
92
93     if (PID1 == 0) { // P1
94         PID2 = fork();
95         if (PID2 < 0) exit(0);
96
97         if (PID2 == 0) { // P2 (Writer)
98             close(fd[0]);
99             dup2(fd[1], STDOUT_FILENO);
100            close(fd[1]);
101            if (execvp(parsed[0], parsed) < 0) {
102                perror("Exec P1 failed");
103                exit(0);
104            }
105        } else { // P1 (Reader)
106
107        }
108    }
109 }
```

Hình 1 Nội dung code (1)



```
C ba6.c
Documents > I007 > LAB06 > C ba6.c
55 void control_sig(int sign)
56 {
57     printf("\nit007sh~");
58     fflush(stdout);
59 }
60
61 void do_redirect_output(char *filename)
62 {
63     int fd = open(filename, O_CREAT | O_WRONLY | O_TRUNC, 0666);
64     if (fd < 0) {
65         perror("Lỗi mở file output");
66         exit(1);
67     }
68     dup2(fd, STDOUT_FILENO);
69     close(fd);
70 }
71
72 void do_redirect_input(char *filename)
73 {
74     int fd = open(filename, O_RDONLY);
75     if (fd < 0) {
76         perror("Lỗi mở file input");
77         exit(1);
78     }
79     dup2(fd, STDIN_FILENO);
80     close(fd);
81 }
82
83 void Execute_pile(char *parsed[], char *parsedpipe[])
84 {
85     pid_t PID1, PID2;
86     int fd[2];
87
88     if (pipe(fd) < 0) { perror("Pipe failed"); return; }
89
90     PID1 = fork();
91     if (PID1 < 0) return;
92
93     if (PID1 == 0) { // P1
94         PID2 = fork();
95         if (PID2 < 0) exit(0);
96
97         if (PID2 == 0) { // P2 (Writer)
98             close(fd[0]);
99             dup2(fd[1], STDOUT_FILENO);
100            close(fd[1]);
101            if (execvp(parsed[0], parsed) < 0) {
102                perror("Exec P1 failed");
103                exit(0);
104            }
105        } else { // P1 (Reader)
106
107        }
108    }
109 }
```

Hình 2 Nội dung code (2)

Hình 3 Nội dung code (3)

Hình 4 Nội dung code (4)

```

C_ba6.c
Documents > IT007 > LAB06 > C_ba6.c
179 {
180     if (strcmp(command, "HF") == 0) {
181         }
182     } else {
183         if (count_HF < MAX_LINE) {
184             strcpy(history_command[count_HF], command);
185             count_HF++;
186         }
187     }
188     char command_copy[MAX_LINE];
189     strcpy(command_copy, command);
190     iPileExe = ExecuteString(command_copy, First_Command, Second_Command);
191
192     if (iPileExe == 0) {
193         Tokenerizer(args, command, " ", &number_of_args);
194
195         if (args[0] == NULL) {
196             FreeArgs(args, number_of_args);
197             continue;
198         }
199
200         if (strcmp(args[0], "exit") == 0) {
201             FreeArgs(args, number_of_args);
202             break;
203         }
204         if (strcmp(args[0], "cd") == 0) {
205             char dir[MAX_LINE];
206             if (args[1] == NULL || strcmp(args[1], "-") == 0) strcpy(dir, getenv("HOME"));
207             else strcpy(dir, args[1]);
208
209             if (chdir(dir) != 0) perror("Lỗi cd");
210             FreeArgs(args, number_of_args);
211             continue;
212         }
213
214         if (strcmp(args[0], "&") == 0) {
215             for(int i=0; i<number_of_args; i++) exec_args[i] = args[i];
216             exec_args[number_of_args] = NULL;
217             int exec_argc = number_of_args;
218
219             int run_background = 0;
220             if (exec_argc > 0 && strcmp(exec_args[exec_argc - 1], "&") == 0) {
221                 run_background = 1;
222                 exec_args[exec_argc - 1] = NULL;
223                 exec_argc--;
224             }
225
226             char *file_to_redirect = NULL;
227             int redirect_type = 0;
228
229             if (exec_argc > 1) {
230                 if (strcmp(exec_args[exec_argc - 2], ">") == 0) {
231                     redirect_type = 1;
232                     file_to_redirect = exec_args[exec_argc - 1];
233                     exec_args[exec_argc - 2] = NULL;
234                 }
235                 else if (strcmp(exec_args[exec_argc - 2], "<") == 0) {
236                     redirect_type = 2;
237                     file_to_redirect = exec_args[exec_argc - 1];
238                     exec_args[exec_argc - 2] = NULL;
239                 }
240             }
241
242             pid = fork();
243             if (pid < 0) {
244                 perror("Fork failed");
245                 FreeArgs(args, number_of_args);
246                 return -1;
247             }
248
249             if (pid == 0) { // CHILD
250                 if (redirect_type == 1) do_redirect_output(file_to_redirect);
251                 else if (redirect_type == 2) do_redirect_input(file_to_redirect);
252
253                 if (execvp(exec_args[0], exec_args) < 0) {
254                     printf("Lệnh không hợp lệ\n");
255                 }
256                 exit(1);
257             }
258             else { // PARENT
259                 if (!run_background) wait(NULL);
260             }
261             FreeArgs(args, number_of_args);
262         }
263     }
264
265     else {
266         Execute_pile(First_Command, Second_Command);
267         FreePipeArgs(First_Command);
268         FreePipeArgs(Second_Command);
269     }
270
271     return 0;
272 }

```

Hình 5 Nội dung code (5)

```

C_ba6.c
Documents > IT007 > LAB06 > C_ba6.c
179 {
180     if (iPileExe == 0) {
181         if (exec_argc > 0 && strcmp(exec_args[exec_argc - 1], "&") == 0) {
182             }
183
184             char *file_to_redirect = NULL;
185             int redirect_type = 0;
186
187             if (exec_argc > 1) {
188                 if (strcmp(exec_args[exec_argc - 2], ">") == 0) {
189                     redirect_type = 1;
190                     file_to_redirect = exec_args[exec_argc - 1];
191                     exec_args[exec_argc - 2] = NULL;
192                 }
193                 else if (strcmp(exec_args[exec_argc - 2], "<") == 0) {
194                     redirect_type = 2;
195                     file_to_redirect = exec_args[exec_argc - 1];
196                     exec_args[exec_argc - 2] = NULL;
197                 }
198             }
199
200             pid = fork();
201             if (pid < 0) {
202                 perror("Fork failed");
203                 FreeArgs(args, number_of_args);
204                 return -1;
205             }
206
207             if (pid == 0) { // CHILD
208                 if (redirect_type == 1) do_redirect_output(file_to_redirect);
209                 else if (redirect_type == 2) do_redirect_input(file_to_redirect);
210
211                 if (execvp(exec_args[0], exec_args) < 0) {
212                     printf("Lệnh không hợp lệ\n");
213                 }
214                 exit(1);
215             }
216             else { // PARENT
217                 if (!run_background) wait(NULL);
218             }
219             FreeArgs(args, number_of_args);
220         }
221     }
222
223     else {
224         Execute_pile(First_Command, Second_Command);
225         FreePipeArgs(First_Command);
226         FreePipeArgs(Second_Command);
227     }
228
229     return 0;
230 }

```

Hình 6 Nội dung code (6)

- Giải thích code
- Các thư viện và hằng số
 - Các thư viện <stdio.h>, <unistd.h>, <syswait.h>, ... cung cấp các hàm hệ thống để thao tác nhập/xuất, tạo tiến trình, thực thi lệnh và xử lý các tín hiệu
 - #define MAX_LINE 80: Quy định độ dài tối đa của một câu lệnh là 80 ký tự
- Các hàm hỗ trợ xử lý chuỗi và bộ nhớ
 - EnterCommand(char command[]): Dùng để đọc dòng lệnh người dùng nhập từ bàn phím. Thực hiện bằng cách dùng fgets để đọc và xóa các ký tự xuống dòng ở cuối chuỗi để các lệnh hệ thống hiểu được
 - Tokernizer: Dùng để tách chuỗi lệnh dài thành các từ riêng biệt. Thực hiện bằng cách sử dụng strdup(p) để cấp phát bộ nhớ động riêng biệt cho từng từ
 - FreeArgs() và FreePipeArgs(): Dùng để dọn dẹp bộ nhớ sau khi thực thi xong một lệnh vì Tokernizer dùng strdup để cấp phát động, nếu không giải phóng thì máy có thể bị memory leak sau một thời gian chạy
- Xử lý tín hiệu và I/O
 - control_sig(int sign): Dùng để xử lý tín hiệu ngắt, ví dụ khi ấn Ctrl + C thay vì chương trình bị tắt ngầm thì hàm này sẽ chặn tín hiệu đó và chỉ in ra dòng nhắc it007sh> mới
 - do_redirect_output(char *filename): Dùng để xử lý toán tử >. Thực hiện bằng cách mở file và dùng dup2 để thay thế STDOUT bằng file đó. Kết quả lệnh sẽ chui vào file thay vì hiện lên màn hình
 - do_redirect_input(char *filename): Dùng để xử lý toán tử <. Thực hiện bằng cách dùng dup2 để thay thế STDIN bằng nội dung file và lệnh sẽ đọc dữ liệu từ file
- Xử lý Pipe
 - Execute_pile(): Dùng để thực thi hai lệnh nối nhau bởi dấu |. Thực hiện bằng cách tạo một pipe, sau đó fork để tạo tiến trình con, cháu. P2(Writer) để chạy lệnh đầu tiên, chuyển đầu ra vào đầu ghi của ống. P1(Reader) chạy lệnh thứ hai, chuyển đầu vào lấy từ đầu đọc của ống. Parent(shell) dùng để đóng toàn bộ ống và chờ con xong
 - Find_pile_char, parseCommandLineSimple, ExecuteString: Nhóm hàm này phát hiện dấu |, cắt đôi câu lệnh thành 2 phần sau đó tách từ cho từng phần để chuẩn bị đưa vào Execute_pile
- Hàm main
 - Một vòng lặp vô hạn điều khiển luồng chạy của Shell
 - Nếu là lệnh pipe thì gọi ExecuteString() để xem có dấu | không, nếu có thì chạy theo luồng Pipe
 - Nếu không phải là lệnh pipe: Tách chuỗi lệnh (Tokernizer), và thực thi các hàm được người dùng nhập

Câu 1: Thực thi command trong tiến trình con. Ví dụ: khi thực hiện it007sh> cat abc.txt. Kết quả sẽ hiển thị nội dung của file abc.txt, kết thúc dòng lệnh sẽ hiển thị dấu nhắc it007sh> để người dùng nhập lệnh tiếp theo. Lưu ý rằng trong thời thực thi của lệnh cat abc.txt, không cho người dùng nhập command mới.

The screenshot shows a terminal window titled "nguyenkhoanguyen [SSH: local]". The terminal prompt is "it007sh>". In the main pane, the content of the file "abc.txt" is displayed:

```

1 Nguyen Khoa Nguyen KHMT2024.3 24521190
2

```

Hình 7 Nội dung file abc.txt

The screenshot shows a terminal window titled "nguyenkhoanguyen@nguyenkhoanguyen-Inspiron-16-Plus-7630: ~/Documents/IT007/LAB06". The terminal prompt is "it007sh>". The user runs several commands:

```

it007sh>cd Documents/IT007/LAB06
it007sh>ls
abc.txt ba6 ba6.c test1.txt test2.txt test.txt
it007sh>./ba6
it007sh>cat abc.txt
Nguyen Khoa Nguyen KHMT2024.3 24521190
it007sh>echo abc
abc
it007sh>ls
abc.txt ba6 ba6.c test1.txt test2.txt test.txt
it007sh>

```

Hình 8 Kết quả khi thực thi các lệnh cơ bản

- Khi gõ sai lệnh

```
nguyenkhoanguyen@nguyenkhoanguyen-Inspiron-16-Plus-7630:~$ cd Documents/IT007/LAB06/
nguyenkhoanguyen@nguyenkhoanguyen-Inspiron-16-Plus-7630:~/Documents/IT007/LAB06$ ls
abc.txt ba6 ba6.c test1.txt test2.txt test.txt
nguyenkhoanguyen@nguyenkhoanguyen-Inspiron-16-Plus-7630:~/Documents/IT007/LAB06$ ./ba6
it007sh>cat abc.txt
Nguyen Khoa Nguyen KHMT2024.3 24521190
it007sh>echo abc
abc
it007sh>ls
abc.txt ba6 ba6.c test1.txt test2.txt test.txt
it007sh>abc.txt
Lệnh không hợp lệ
it007sh>
```

Hình 9 Kết quả nếu chúng ta nhập sai lệnh

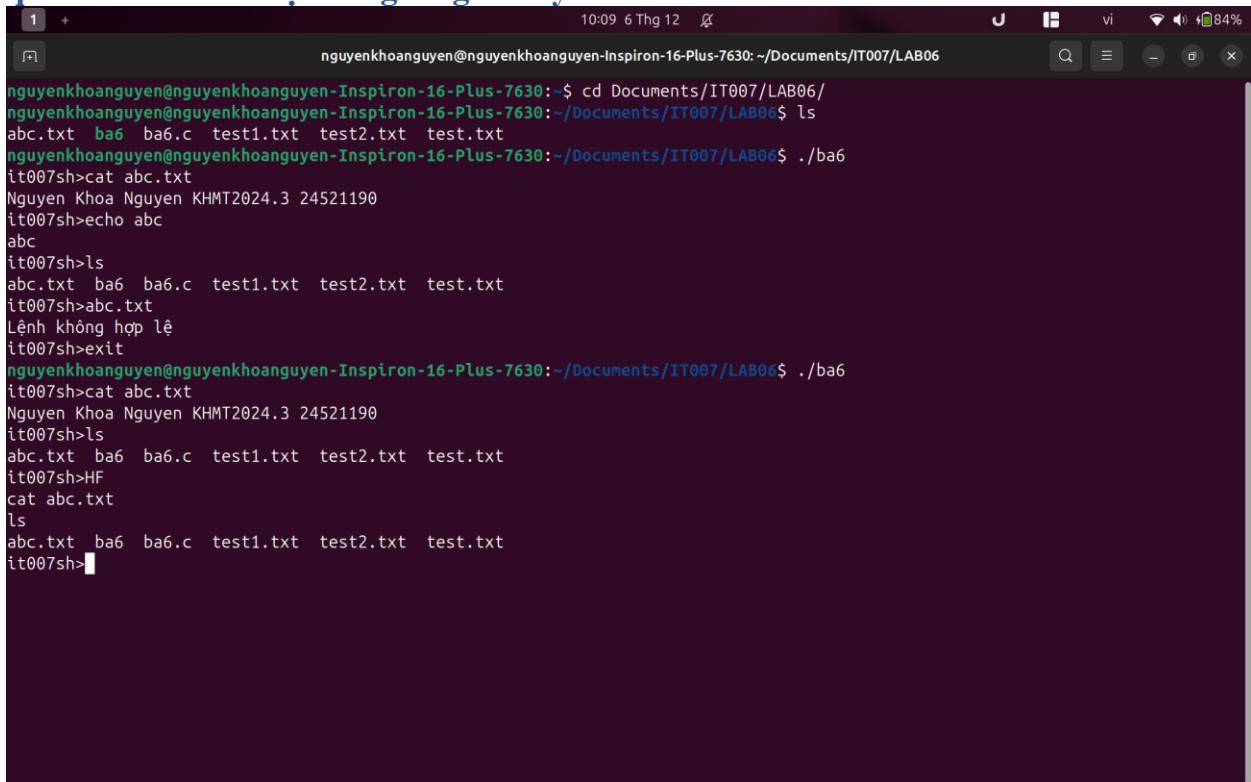
```
nguyenkhoanguyen@nguyenkhoanguyen-Inspiron-16-Plus-7630:~$ cd Documents/IT007/LAB06/
nguyenkhoanguyen@nguyenkhoanguyen-Inspiron-16-Plus-7630:~/Documents/IT007/LAB06$ ls
abc.txt ba6 ba6.c test1.txt test2.txt test.txt
nguyenkhoanguyen@nguyenkhoanguyen-Inspiron-16-Plus-7630:~/Documents/IT007/LAB06$ ./ba6
it007sh>cat abc.txt
Nguyen Khoa Nguyen KHMT2024.3 24521190
it007sh>echo abc
abc
it007sh>ls
abc.txt ba6 ba6.c test1.txt test2.txt test.txt
it007sh>abc.txt
Lệnh không hợp lệ
it007sh>exit
nguyenkhoanguyen@nguyenkhoanguyen-Inspiron-16-Plus-7630:~/Documents/IT007/LAB06$
```

Hình 10 Nhập lệnh exit để thoát chương trình

❖ Giải thích kết quả:

- Hàm EnterCommand đọc chuỗi "cat abc.txt"
- Tokernizer: Tách thành mảng args: ["cat", "abc.txt", NULL]
- Nếu không có | (pipe), không có redirect (>, <), không phải exit hay cd
- Tạo tiến trình con bằng fork
- Thực thi execv: Tiến trình con biến đổi thành chương trình cat, đọc file abc.txt và in nội dung "Nguyen Khoa Nguyen KHMT2024.3 24521190" ra màn hình

Câu 2: Tạo tính năng sử dụng lại câu lệnh gần đây nhất. Cung cấp History Feature, cho phép người dùng thực thi lệnh gần đây nhất bằng cách nhập 'HF' và nhấn enter. Khi sử dụng ký tự 'HF', shell sẽ lần lượt hiện lại các lệnh đã sử dụng trong quá khứ theo thứ tự thời gian gần đây nhất.



The screenshot shows a terminal window on a Linux system. The title bar reads "nguyenkhoanguyen@nguyenkhoanguyen-Inspiron-16-Plus-7630: ~/Documents/IT007/LAB06". The terminal content is as follows:

```
nguyenkhoanguyen@nguyenkhoanguyen-Inspiron-16-Plus-7630:~$ cd Documents/IT007/LAB06/
nguyenkhoanguyen@nguyenkhoanguyen-Inspiron-16-Plus-7630:~/Documents/IT007/LAB06$ ls
abc.txt ba6 ba6.c test1.txt test2.txt test.txt
nguyenkhoanguyen@nguyenkhoanguyen-Inspiron-16-Plus-7630:~/Documents/IT007/LAB06$ ./ba6
it007sh>cat abc.txt
Nguyen Khoa Nguyen KHMT2024.3 24521190
it007sh>echo abc
abc
it007sh>ls
abc.txt ba6 ba6.c test1.txt test2.txt test.txt
it007sh>abc.txt
Lệnh không hợp lệ
it007sh>exit
nguyenkhoanguyen@nguyenkhoanguyen-Inspiron-16-Plus-7630:~/Documents/IT007/LAB06$ ./ba6
it007sh>cat abc.txt
Nguyen Khoa Nguyen KHMT2024.3 24521190
it007sh>ls
abc.txt ba6 ba6.c test1.txt test2.txt test.txt
it007sh>HF
cat abc.txt
ls
abc.txt ba6 ba6.c test1.txt test2.txt test.txt
it007sh>
```

Hình 11 Kết quả khi ta nhập lệnh HF

❖ Giải thích kết quả:

- Trước khi gõ HF, chương trình đã chạy qua một vài lệnh: cat abc.txt và ls. Do đó mảng history_command đang chứa ["cat abc.txt", "ls"] và biến count_HF có giá trị là 2
- Khi nhập lệnh HF, chương trình so sánh chuỗi vừa nhập: strcmp(command, "HF") == 0. Kết quả là đúng nên nhảy vào xử lý lịch sử

- strcpy(command, history_command[count_HF - 1]): Lấy lệnh tại vị trí cuối cùng và copy đè lên biến command, do đó bây giờ command không còn là "HF" nữa mà chính là "ls"

Câu 3: Chuyển hướng vào ra Hỗ trợ các toán tử chuyển hướng '>' và '<', trong đó '>' chuyển hướng đầu ra của lệnh sang một tệp và '<' chuyển hướng đầu vào của lệnh từ một tệp.

- **Ví dụ: nếu người dùng nhập it007sh>ls > out.txt. Đầu ra từ lệnh ls sẽ được chuyển hướng đến tệp out.txt. Tương tự, đầu vào cũng có thể được chuyển hướng.**
- **Ví dụ, nếu người dùng nhập it007sh>sort < in.txt. Tệp in.txt sẽ đóng vai trò là đầu vào cho lệnh sắp xếp.**

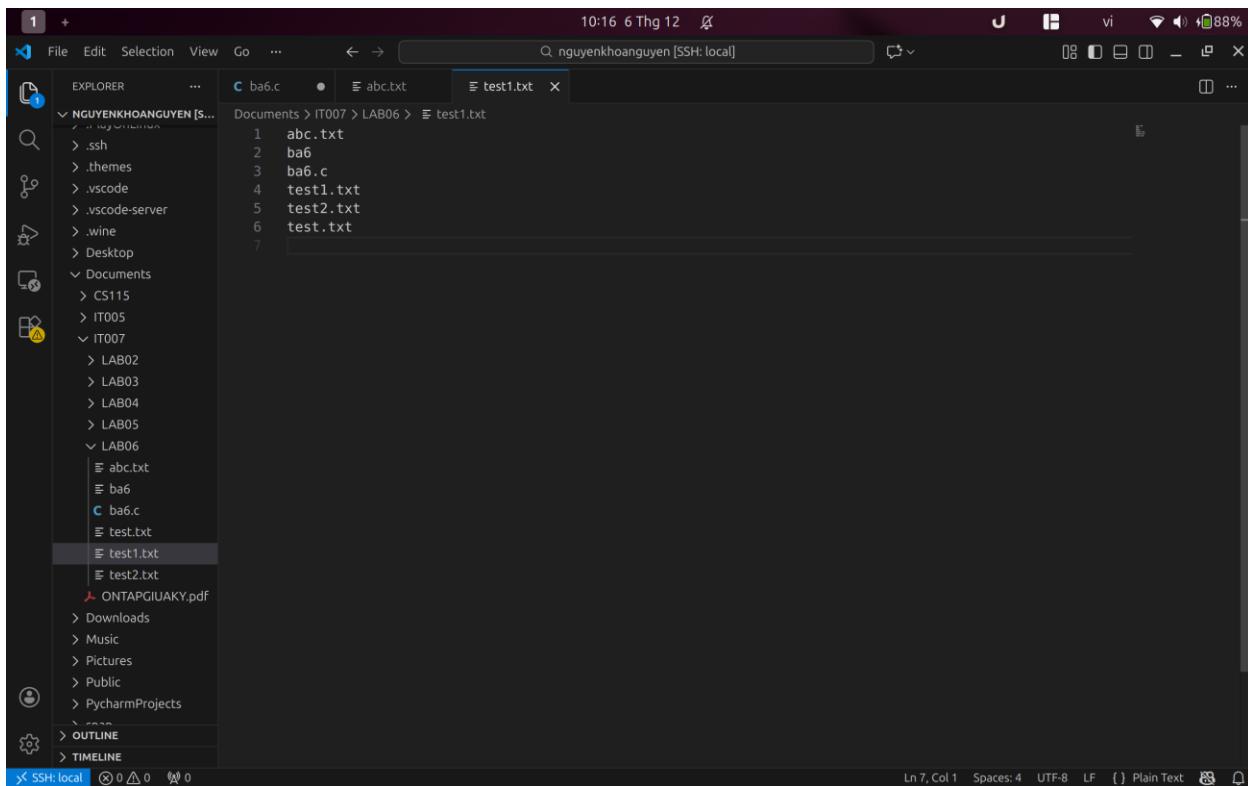
- Testcase 1: Thực thi lệnh ls > test1.txt

```
File Edit Selection View Go ... 10:15 6 Thg 12 2023 nguyenkhoanguyen [SSH: local]
EXPLORER ... C ba6.c abc.txt test1.txt
NGUYENKHOANGUYEN [S... Documents > IT007 > LAB06 > test1.txt
> .dotnet
> .gnupg
> .ipython
> .java
> .jupyter
> .local
> .pki
> .PlayOnLinux
> .ssh
> .themes
> .vscode
> .vscode-server
> .wine
> Desktop
Documents CS115
IT005
IT007 LAB02
LAB03
LAB04
LAB05
LAB06 abc.txt
ba6
C ba6.c
test.txt
test1.txt
OUTLINE
TIMELINE
SSH: local 0 0 0
Ln 1, Col 1 Spaces: 4 UTF-8 LF Plain Text
```

Hình 12 Nội dung ban đầu của file test1.txt

```
nguyenkhoa@nguyenkhoa-Inspiron-16-Plus-7630: ~$ cd Documents/IT007/LAB06/
nguyenkhoa@nguyenkhoa-Inspiron-16-Plus-7630: ~/Documents/IT007/LAB06$ ls
abc.txt ba6 ba6.c test1.txt test2.txt test.txt
nguyenkhoa@nguyenkhoa-Inspiron-16-Plus-7630: ~/Documents/IT007/LAB06$ ./ba6
it007sh>cat abc.txt
Nguyen Khoa Nguyen KHMT2024.3 24521190
it007sh>echo abc
abc
it007sh>ls
abc.txt ba6 ba6.c test1.txt test2.txt test.txt
it007sh>abc.txt
Lệnh không hợp lệ
it007sh>exit
nguyenkhoa@nguyenkhoa-Inspiron-16-Plus-7630: ~/Documents/IT007/LAB06$ ./ba6
it007sh>cat abc.txt
Nguyen Khoa Nguyen KHMT2024.3 24521190
it007sh>ls
abc.txt ba6 ba6.c test1.txt test2.txt test.txt
it007sh>HF
cat abc.txt
ls
abc.txt ba6 ba6.c test1.txt test2.txt test.txt
it007sh>ls > test1.txt
it007sh>
```

Hình 13 Nhập lệnh ls > test1.txt

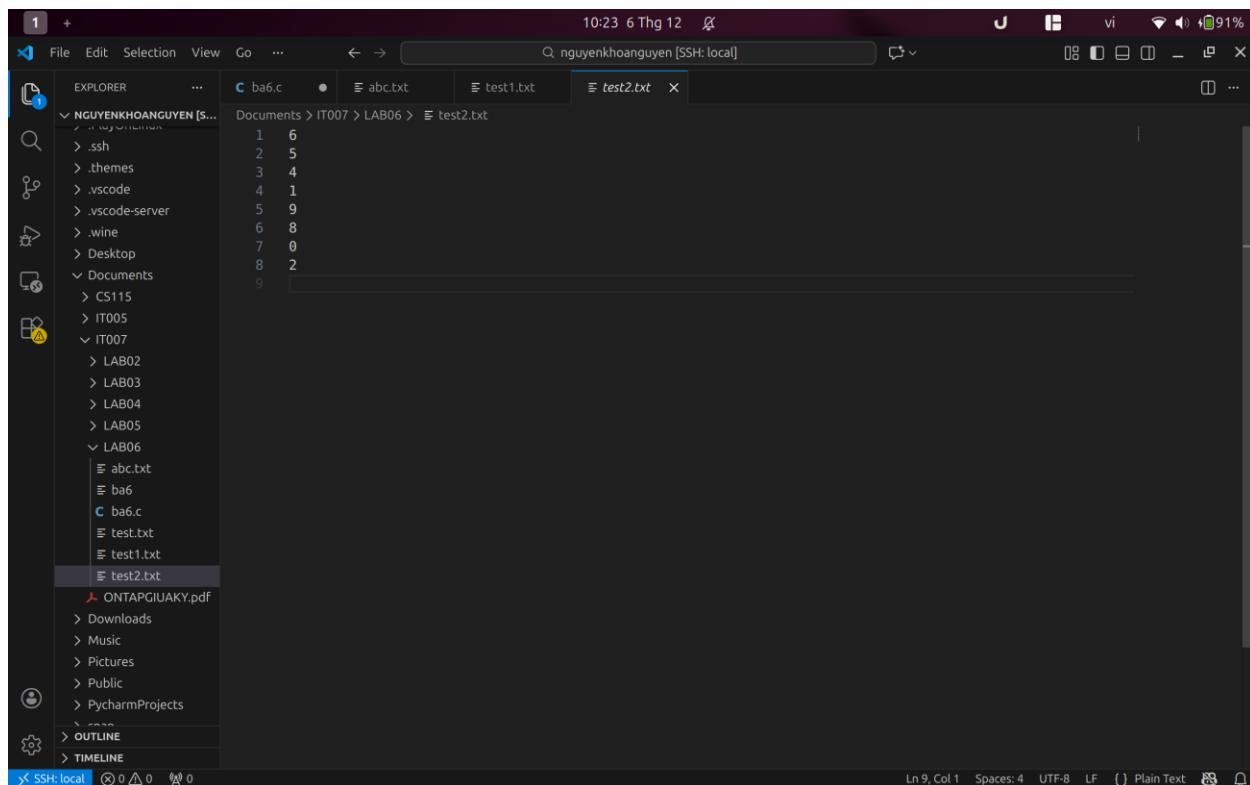


Hình 14 Nội dung của file test1.txt sau khi thực hiện nhập lệnh ls

Giải thích kết quả:

- Khi nhập lệnh ls > test1.txt thì mảng exec_args ban đầu là ["ls", ">", "test1.txt", NULL]
- Chương trình quét thấy kí tự ">" và xác định đây là output redirect và file đích chính là "test1.txt". Chương trình gán NULL vào vị trí dấu ">" và mảng exec_args chỉ còn ["ls", NULL]
- open("test1.txt"): Cờ O_TRUNC sẽ xóa sạch nội dung trong file nếu nó đã tồn tại
- dup2(fd, STDOUT_FILENO): FD 1 bây giờ không trả vào màn hình nữa mà trả thẳng vào file test1.txt, do đó màn hình không hiện gì cả mà chỉ có nội dung trong file

■ Testcase 2: Thực thi lệnh sort < test2.txt



The screenshot shows a terminal window titled "nguyenkhoanguyen [SSH: local]" with the command "sort < test2.txt" run. The terminal displays the following sorted content from the file:

```
1 6
2 5
3 4
4 1
5 9
6 8
7 0
8 2
9
```

Hình 15 Nội dung ban đầu của file test2.txt

```
1  + 10:23 6 Thg 12 2024 91%  
nguyenkhoanguyen@nguyenkhoanguyen-Inspiron-16-Plus-7630:~/Documents/IT007/LAB06  
abc.txt ba6 ba6.c test1.txt test2.txt test.txt  
nguyenkhoanguyen@nguyenkhoanguyen-Inspiron-16-Plus-7630:~/Documents/IT007/LAB06$ ./ba6  
it007sh>cat abc.txt  
Nguyen Khoa Nguyen KHMT2024.3 24521190  
it007sh>echo abc  
abc  
it007sh>ls  
abc.txt ba6 ba6.c test1.txt test2.txt test.txt  
it007sh>abc.txt  
Lệnh không hợp lệ  
it007sh>exit  
nguyenkhoanguyen@nguyenkhoanguyen-Inspiron-16-Plus-7630:~/Documents/IT007/LAB06$ ./ba6  
it007sh>cat abc.txt  
Nguyen Khoa Nguyen KHMT2024.3 24521190  
it007sh>ls  
abc.txt ba6 ba6.c test1.txt test2.txt test.txt  
it007sh>HF  
cat abc.txt  
ls  
abc.txt ba6 ba6.c test1.txt test2.txt test.txt  
it007sh>ls > test1.txt  
it007sh>ls > test1.txt  
it007sh>sort < test2.txt  
0  
1  
2  
4  
5  
6  
8  
9  
it007sh>
```

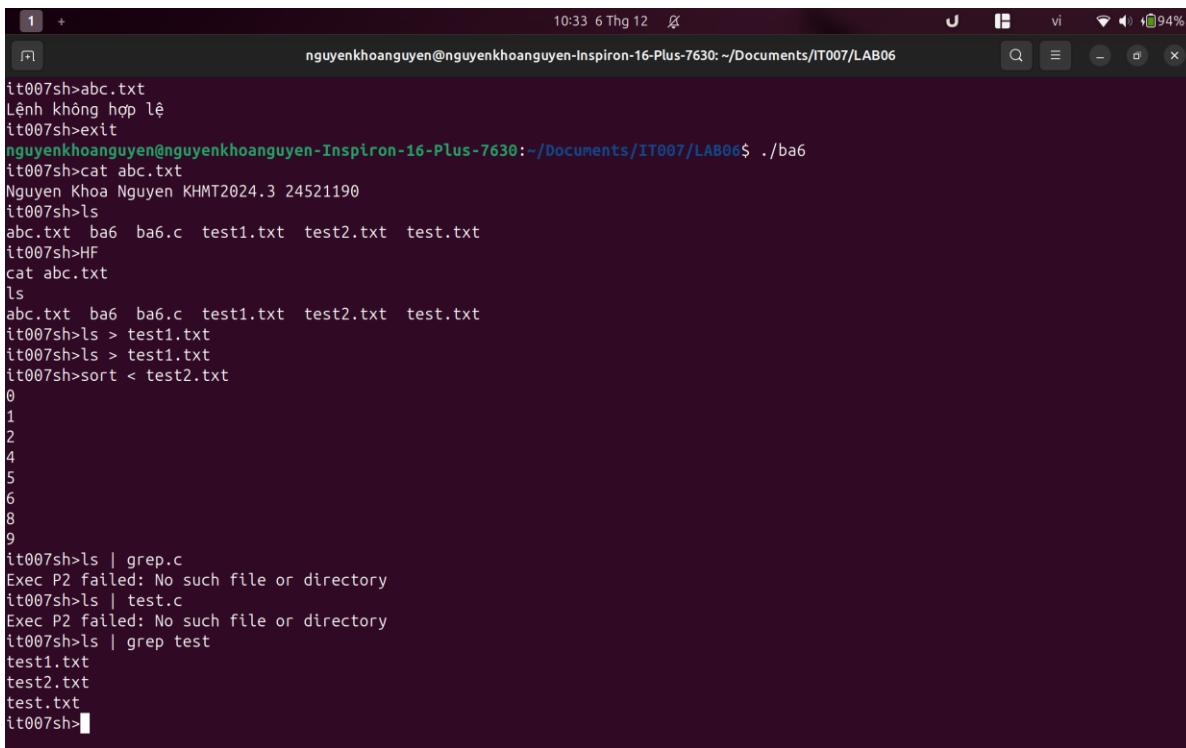
Hình 16 Thực thi lệnh sort < test2.txt

⊕ Giải thích kết quả:

- Vì quét thấy "<" nên đặt redirect_type = 2, lưu tên file cần đọc: file_to_redirect = "test2.txt". Sau đó cắt lệnh và gán NULL vào vị trí dấu "<"
- fd = open("test2.txt", O_RDONLY): Mở file ở chế độ chỉ đọc
- dup2(fd, STDIN_FILENO): Khi bất kỳ lệnh nào đọc từ FD 0 thì nó sẽ lấy dữ liệu từ file test2.txt
- Khi dùng < thì test2.txt -> FD 0 -> sort -> FD 1 -> Màn hình

Câu 4: Giao tiếp sử dụng cơ chế đường ống Cho phép đầu ra của một lệnh đóng vai trò là đầu vào cho lệnh khác bằng cách sử dụng một đường ống. Ví dụ: it007sh>ps aux | grep firefox. Có đầu ra của lệnh ps aux đóng vai trò là đầu vào cho lệnh grep firefox . Gợi ý: sử dụng hàm pipe() và dup2()

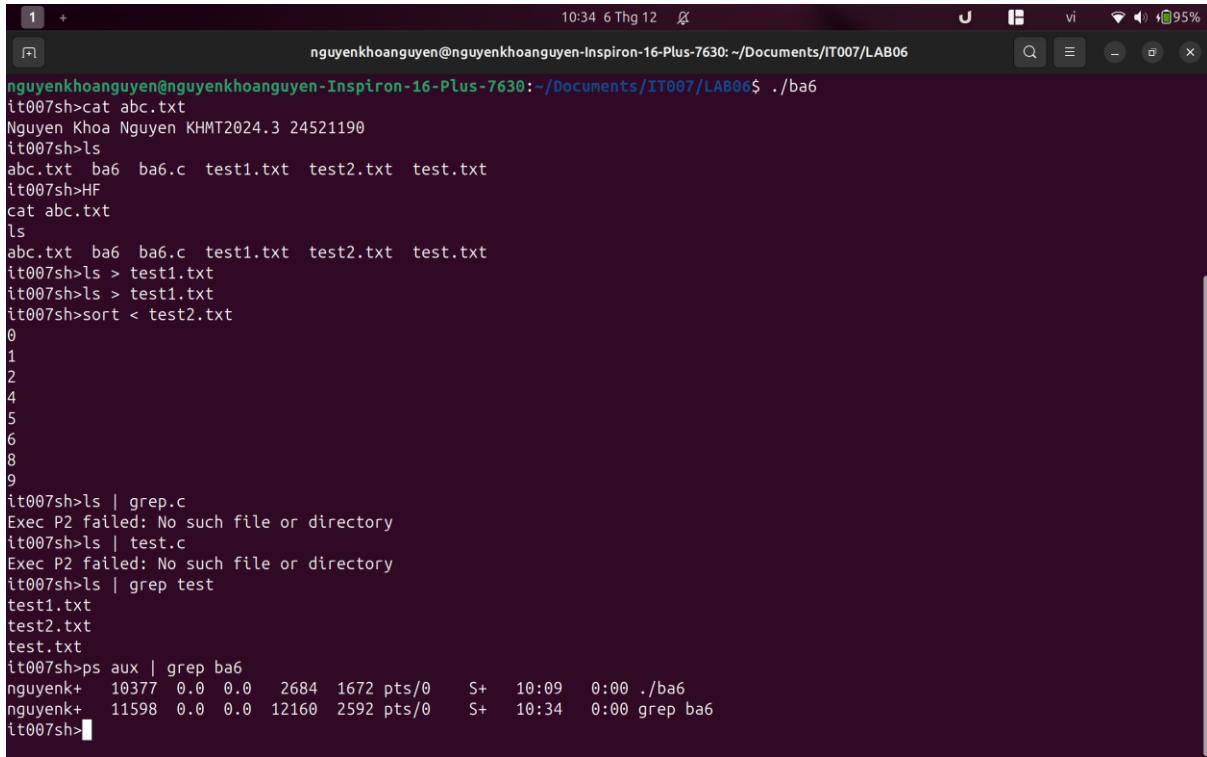
- Testcase 1: Lọc tên file (ls | grep)



```
it007sh>abc.txt
Lệnh không hợp lệ
it007sh>exit
nguyenkhoanguyen@nguyenkhoanguyen-Inspiron-16-Plus-7630:~/Documents/IT007/LAB06$ ./ba6
it007sh>cat abc.txt
Nguyen Khoa Nguyen KHMT2024.3 24521190
it007sh>ls
abc.txt ba6 ba6.c test1.txt test2.txt test.txt
it007sh>HF
cat abc.txt
ls
abc.txt ba6 ba6.c test1.txt test2.txt test.txt
it007sh>ls > test1.txt
it007sh>ls > test1.txt
it007sh>sort < test2.txt
0
1
2
4
5
6
8
9
it007sh>ls | grep.c
Exec P2 failed: No such file or directory
it007sh>ls | test.c
Exec P2 failed: No such file or directory
it007sh>ls | grep test
test1.txt
test2.txt
test.txt
it007sh>
```

Hình 17 Thực thi lệnh ls | grep test

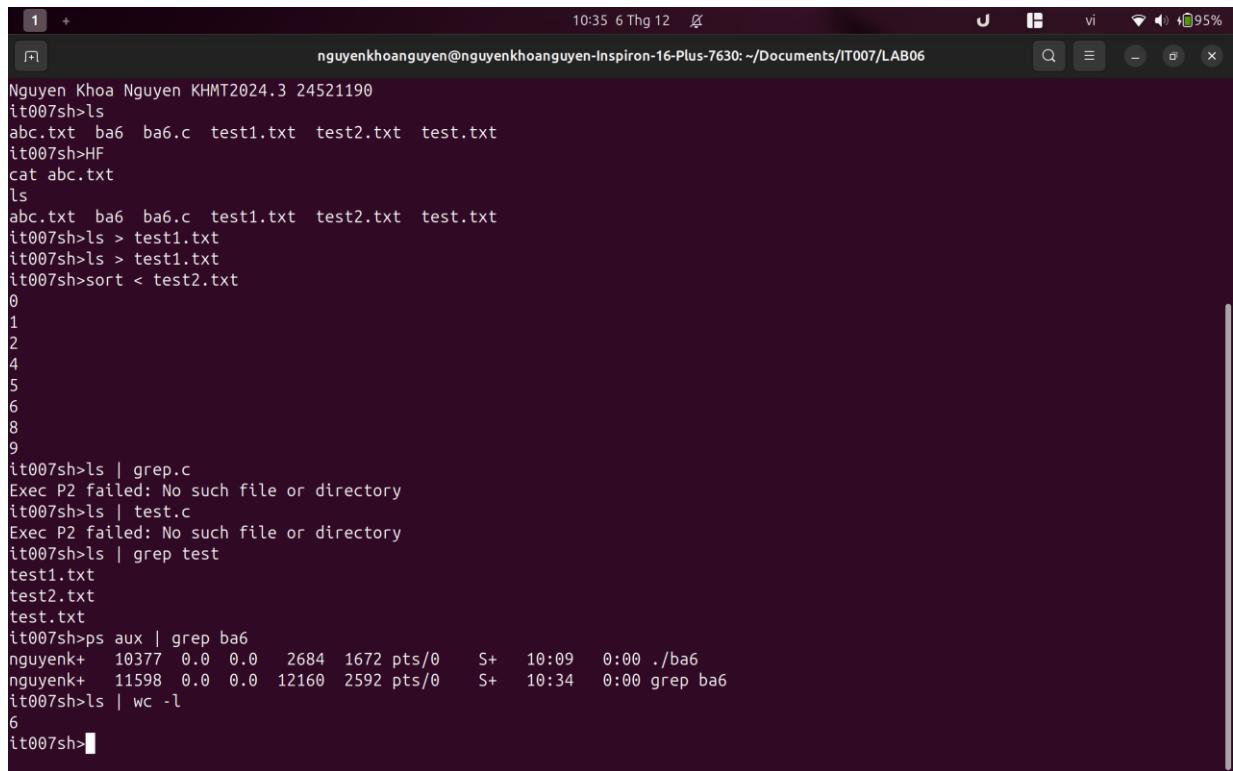
- Testcase 2: Kiểm tra tiến trình (ps | grep): Hiện ra thông tin về tiến trình (PID, User, Time....) đang chạy



```
1 +                               10:34 6 Thg 12 4
nguyenkhoanguyen@nguyenkhoanguyen-Inspiron-16-Plus-7630:~/Documents/IT007/LAB06$ ./ba6
it007sh>cat abc.txt
Nguyen Khoa Nguyen KHMT2024.3 24521190
it007sh>ls
abc.txt ba6 ba6.c test1.txt test2.txt test.txt
it007sh>HF
cat abc.txt
ls
abc.txt ba6 ba6.c test1.txt test2.txt test.txt
it007sh>ls > test1.txt
it007sh>ls > test1.txt
it007sh>sort < test2.txt
0
1
2
4
5
6
8
9
it007sh>ls | grep.c
Exec P2 failed: No such file or directory
it007sh>ls | test.c
Exec P2 failed: No such file or directory
it007sh>ls | grep test
test1.txt
test2.txt
test.txt
it007sh>ps aux | grep ba6
nguyenk+ 10377 0.0 0.0 2684 1672 pts/0 S+ 10:09 0:00 ./ba6
nguyenk+ 11598 0.0 0.0 12160 2592 pts/0 S+ 10:34 0:00 grep ba6
it007sh>
```

Hình 18 Thực thi lệnh ps / grep

- Testcase : Đếm số lượng (ls | wc): Hiện ra tổng số file và thư mục đang có trong folder hiện tại

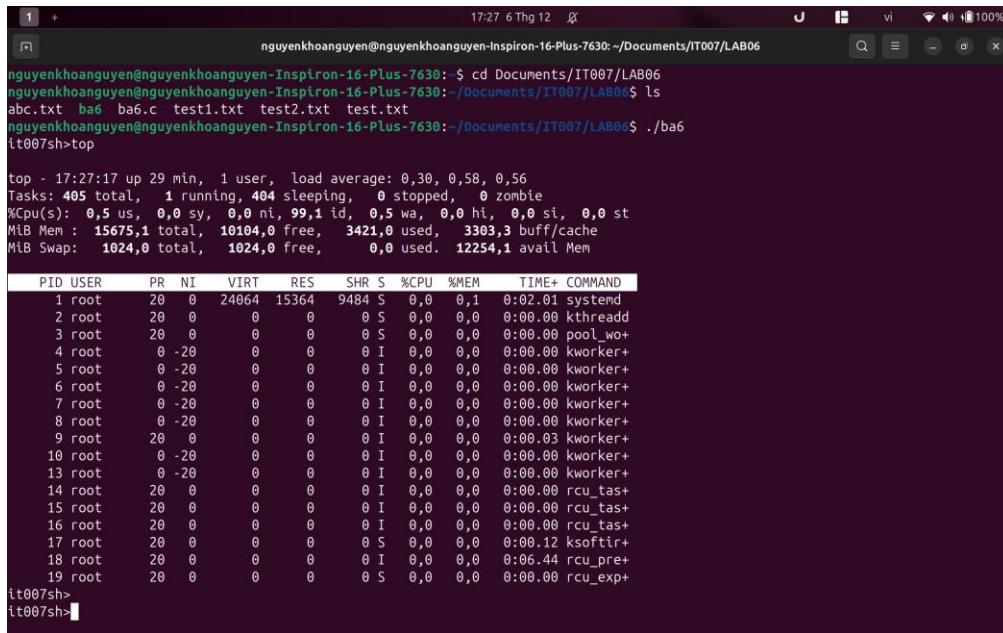


```
1 +                               10:35 6 Thg 12 4
nguyenkhoanguyen@nguyenkhoanguyen-Inspiron-16-Plus-7630:~/Documents/IT007/LAB06$ 
Nguyen Khoa Nguyen KHMT2024.3 24521190
it007sh>ls
abc.txt ba6 ba6.c test1.txt test2.txt test.txt
it007sh>HF
cat abc.txt
ls
abc.txt ba6 ba6.c test1.txt test2.txt test.txt
it007sh>ls > test1.txt
it007sh>ls > test1.txt
it007sh>sort < test2.txt
0
1
2
4
5
6
8
9
it007sh>ls | grep.c
Exec P2 failed: No such file or directory
it007sh>ls | test.c
Exec P2 failed: No such file or directory
it007sh>ls | grep test
test1.txt
test2.txt
test.txt
it007sh>ps aux | grep ba6
nguyenk+ 10377 0.0 0.0 2684 1672 pts/0 S+ 10:09 0:00 ./ba6
nguyenk+ 11598 0.0 0.0 12160 2592 pts/0 S+ 10:34 0:00 grep ba6
it007sh>ls | wc -l
6
it007sh>
```

Hình 19 Thực thi lệnh ls / wc

Câu 5: Kết thúc lệnh đang thực thi. Ví dụ: khi thực hiện lệnh it007sh>top kết quả sẽ liên tục hiển thị các process của hệ thống, khi sử dụng tổ hợp phím Ctrl + C, lệnh thực thi trên sẽ kết thúc và hiển thị dấu nhắc it007sh> mời người dùng nhập lệnh tiếp theo.

- Ngắt lệnh top: Khi ta nhấn Ctrl + C, lệnh top bị ngắt và xuất hiện dòng cho phép người dùng nhập lệnh mới



```

17:27 6 Thg 12 100%
nguyenhoanguyen@nguyenhoanguyen-Inspiron-16-Plus-7630: ~ /Documents/IT007/LAB06
nguyenhoanguyen@nguyenhoanguyen-Inspiron-16-Plus-7630: ~ /Documents/IT007/LAB06 $ ls
abc.txt ba6 ba6.c test1.txt test2.txt test.txt
nguyenhoanguyen@nguyenhoanguyen-Inspiron-16-Plus-7630: ~ /Documents/IT007/LAB06 $ ./ba6
it007sh>top

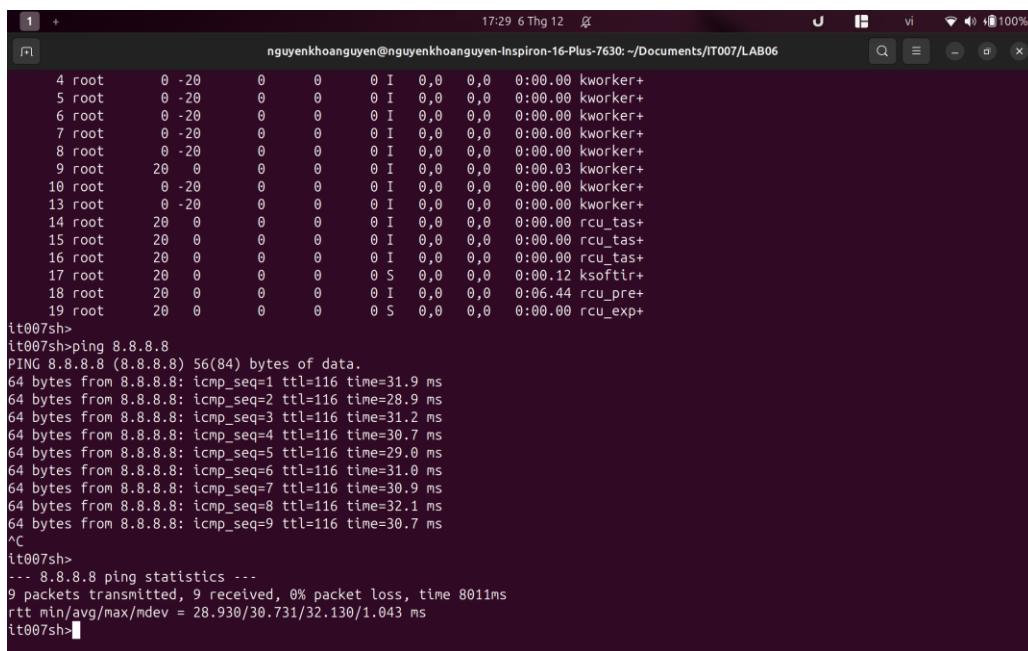
top - 17:27:17 up 29 min, 1 user, load average: 0.30, 0.58, 0.56
Tasks: 405 total, 1 running, 404 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0.5 us, 0.0 sy, 0.0 ni, 99.1 id, 0.5 wa, 0.0 hi, 0.0 si, 0.0 st
Mem: 15675.1 total, 10104.0 free, 3421.0 used, 3303.3 buff/cache
Swap: 1024.0 total, 1024.0 free, 0.0 used. 12254.1 avail Mem

PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND
1 root 20 0 24064 15364 9484 S 0.0 0.1 0:02.01 systemd
2 root 20 0 0 0 0 S 0.0 0.0 0:00.00 kthreadd
3 root 20 0 0 0 0 S 0.0 0.0 0:00.00 pool_wow+
4 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 kworker+
5 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 kworker+
6 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 kworker+
7 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 kworker+
8 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 kworker+
9 root 20 0 0 0 0 I 0.0 0.0 0:00.03 kworker+
10 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 kworker+
13 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 kworker+
14 root 20 0 0 0 0 I 0.0 0.0 0:00.00 rcu_tas+
15 root 20 0 0 0 0 I 0.0 0.0 0:00.00 rcu_tas+
16 root 20 0 0 0 0 I 0.0 0.0 0:00.00 rcu_tas+
17 root 20 0 0 0 0 S 0.0 0.0 0:00.12 ksoftirq+
18 root 20 0 0 0 0 I 0.0 0.0 0:06.44 rcu_pre+
19 root 20 0 0 0 0 S 0.0 0.0 0:00.00 rcu_exp+
it007sh>
it007sh>

```

Hình 20 Kết quả khi ta ngắt lệnh top

- Ngắt lệnh ping: Lệnh ping trên Linux sẽ chạy mãi mãi cho đến khi bị ngắt, do đó ta kiểm tra như sau:



```

17:29 6 Thg 12 100%
nguyenhoanguyen@nguyenhoanguyen-Inspiron-16-Plus-7630: ~ /Documents/IT007/LAB06
4 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 kworker+
5 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 kworker+
6 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 kworker+
7 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 kworker+
8 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 kworker+
9 root 20 0 0 0 0 I 0.0 0.0 0:00.03 kworker+
10 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 kworker+
13 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 kworker+
14 root 20 0 0 0 0 I 0.0 0.0 0:00.00 rcu_tas+
15 root 20 0 0 0 0 I 0.0 0.0 0:00.00 rcu_tas+
16 root 20 0 0 0 0 I 0.0 0.0 0:00.00 rcu_tas+
17 root 20 0 0 0 0 S 0.0 0.0 0:00.12 ksoftirq+
18 root 20 0 0 0 0 I 0.0 0.0 0:06.44 rcu_pre+
19 root 20 0 0 0 0 S 0.0 0.0 0:00.00 rcu_exp+
it007sh>
it007sh>ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=116 time=31.9 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=116 time=28.9 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=116 time=31.2 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=116 time=30.7 ms
64 bytes from 8.8.8.8: icmp_seq=5 ttl=116 time=29.0 ms
64 bytes from 8.8.8.8: icmp_seq=6 ttl=116 time=31.0 ms
64 bytes from 8.8.8.8: icmp_seq=7 ttl=116 time=30.9 ms
64 bytes from 8.8.8.8: icmp_seq=8 ttl=116 time=32.1 ms
64 bytes from 8.8.8.8: icmp_seq=9 ttl=116 time=30.7 ms
^C
it007sh>...
--- 8.8.8.8 ping statistics ---
9 packets transmitted, 9 received, 0% packet loss, time 8011ms
rtt min/avg/max/mdev = 28.930/30.731/32.130/1.043 ms
it007sh>

```

Hình 21 Kết quả khi ta ngắt lệnh ping 8.8.8.8