



ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN

Nguyễn Huy Tú **18120254**

Nguyễn Duy Vũ **18120264**

BÁO CÁO ĐỒ ÁN 1

SIMPLE SHELL

| **Giáo viên hướng dẫn** |

TS. Trần Trung Dũng

Ths. Lê Giang Thanh

Môn học: Hệ Điều Hành

Thành phố Hồ Chí Minh – 2020

MỤC LỤC

MỤC LỤC	2
BÁO CÁO ĐỒ ÁN	3
1. Báo cáo tiến độ.	3
1.1. Thành viên nhóm.....	3
1.2. Mức độ hoàn thành.....	3
2. Báo cáo kĩ thuật.....	3
2.1. Mô tả bài toán.....	3
2.2. Các hàm.	4
2.2.1. Các hàm phục vụ.	4
2.2.2. Thực thi lệnh trong tiến trình con.....	4
2.2.3. Tạo lệnh history.....	4
2.2.4. Điều hướng input và output.....	4
2.2.5. Giao tiếp thông qua pipe.....	5
2.3. Hàm main.	5
3. Các lệnh cơ bản được hỗ trợ.	6
4. Các test-case.	6
TÀI LIỆU THAM KHẢO	10

BÁO CÁO ĐỒ ÁN

1. Báo cáo tiến độ.

1.1. Thành viên nhóm.

Nguyễn Huy Tú - 18120254

Nguyễn Duy Vũ - 18120264

1.2. Mức độ hoàn thành.

Số chức năng làm được

Chức năng	Mức độ hoàn thành
Thực thi lệnh trong tiến trình con	100%
Tạo lệnh history	100%
Điều hướng input, output	100%
Giao tiếp thông qua pipe	100%
Tổng:	100%

2. Báo cáo kĩ thuật.

2.1. Mô tả bài toán.

Được viết bằng ngôn ngữ C/C++, mô phỏng một giao diện shell đơn giản của hệ điều hành Linux, nhận các lệnh của người dùng và thực thi mỗi lệnh trong tiến trình riêng biệt. Đồ án này tập trung thực thi chức năng điều hướng input và output, thực thi pipe - một dạng giao tiếp giữa các tiến trình của một cặp lệnh. Sử dụng lệnh hàm gọi hệ thống của UNIX là `fork()`, `exec()`, `wait()`, `dup2()` và `pipe()`.

Mã nguồn được chứa trong file `simple-shell.c`

2.2. Các hàm.

2.2.1. Các hàm phục vụ.

Hàm *fork()*:

- Tạo ra một tiến trình con mới là bản sao của tiến trình ban đầu là tiến trình cha. Hai tiến trình sẽ sở hữu vùng nhớ riêng tách ra độc lập.
- Hàm sẽ trả về 0 tại tiến trình con, một con số định danh (cho tiến trình con được tạo) tại tiến trình cha.

2.2.2. Thực thi lệnh trong tiến trình con.

Hỗ trợ các lệnh ls, cat, v.v... được nêu ở mục 3.

Hàm *child(char* args[], char* redirec_args[])*

- Sau khi đã gọi hàm *fork()* để tạo tiến trình con, phân tích câu lệnh của người dùng. Gọi hàm *execvp(argv[0], argv)* để thực thi lệnh tại tiến trình con. Nếu lệnh của người dùng có '&' yêu cầu thì tiến trình cha đợi tiến trình con thực hiện xong mới thực thi.
- Không có giá trị trả về.

2.2.3. Tạo lệnh history.

Hàm *history_feature(char *history[], int &count_history, char* line)*

- Dùng mảng tĩnh để lưu một số cố định lệnh đã được người dùng nhập (lịch sử):
 - Người dùng có thể xem và thực hiện lệnh gần đây nhất bằng cách nhập lệnh "!!"
 - Nếu lịch sử trống (chưa có câu lệnh gần đây nhất được lưu) thì chương trình sẽ xuất thông báo "No commands in history".
 - Trường hợp người dùng bỏ trống không nhập lệnh mà chỉ nhấn enter thì khi gõ "!!", chương trình sẽ hiển thị lệnh mới nhập và chạy lệnh đó.
- Giá trị trả về: true hoặc false để chương trình thực hiện lệnh.

2.2.4. Điều hướng input và output.

Hàm *child()*

- Sử dụng hàm *split_redirection()* để biết được người dùng muốn điều hướng input (<) hay điều hướng output (>). Với tham số tiếp theo ký tự điều hướng là đường dẫn điều hướng. Tiếp theo, sao chép ký tự và đường dẫn điều hướng vào mảng chứa các tham số. Sau đó, thực thi các câu lệnh và xoá khỏi mảng các tham số.
- Không trả về giá trị.

2.2.5. Giao tiếp thông qua pipe.

Không hỗ trợ câu lệnh chuyển hướng Input, Output hay câu lệnh giao tiếp qua Pipe.

Hàm *split_pipe()*

Sử dụng '|' là ký tự liên kết 2 câu lệnh. Dùng hàm *parse()* phân tích câu lệnh nhập của người dùng, chương trình sẽ lưu token trước ký tự '|' vào mảng tham số cho tiến trình con thứ nhất, token sau ký tự '|' vào mảng tham số cho tiến trình con thứ hai.

Hàm *exec_with_pipe()*

Nhận vào các mảng tham số đã được tạo bởi hàm *split_pipe()*. Hàm tạo 1 pipe giao tiếp giữa 2 tiến trình. Sau khi khởi tạo pipe, gọi *fork()* để tạo 2 tiến trình con và dùng *dup2()* để sao chép file description output của tiến trình con đến file description input của tiến trình 2 và ngược lại. Cuối cùng, thực thi lệnh dựa vào 2 mảng tham số của 2 tiến trình.

2.3. Hàm main.

Chương trình sẽ thực hiện một vòng lặp vô tận. Sau đó, nhận lệnh nhập vào từ người dùng vào biến line. Kiểm tra xem line có nằm trong những trường hợp:

- Nếu có ký tự '&' ở cuối thì đặt wait = 1 (để thực hiện tiến trình con rồi tiến trình cha).
- Nếu là "exit" thì thoát khỏi vòng lặp.

- Nếu là !! thì line = history.

Kiểm tra TYPE của inputLine như pipe (|), redirect (<,>) hay câu lệnh bình thường.

Tiếp theo, tách token cho inputLine. Mỗi token được phân cách với nhau bởi ký tự phân cách hoặc khoảng trắng. Token sẽ được lưu vào mảng char* argv[] chứa các tham số. Nếu cuối câu lệnh có ký tự '&' thì câu lệnh đó sẽ được thực thi trong tiến trình con, khi tiến trình con kết thúc thì tiến trình cha mới tiếp tục.

Cuối cùng là thực hiện lệnh tùy theo TYPE, kiểm tra &, forking ra tiến trình con và gọi lệnh exec_with_pipe().

3. Các lệnh cơ bản được hỗ trợ.

Câu lệnh "ls" liệt kê thư mục hiện hành và câu lệnh "ls -l" hiện thị đầy đủ các thông tin (quyền truy cập, chủ, kích thước...).

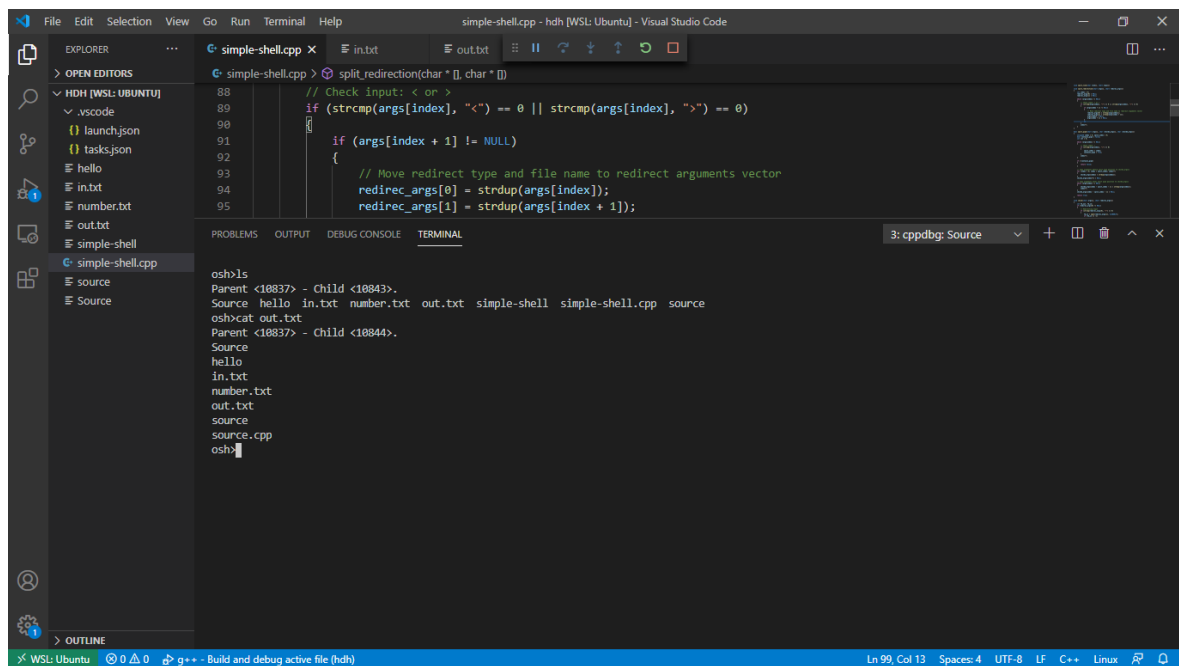
Câu lệnh "exit" dùng để thoát chương trình. Ngoài ra chương trình còn xử lý được các trường hợp nhập lệnh sai.

Câu lệnh "cat" để xem nội dung 1 tập tin ngắn và lệnh "echo" hiện thị nội dung văn bản.

Câu lệnh "grep" để tìm kiếm chuỗi trong file.

4. Các test-case.

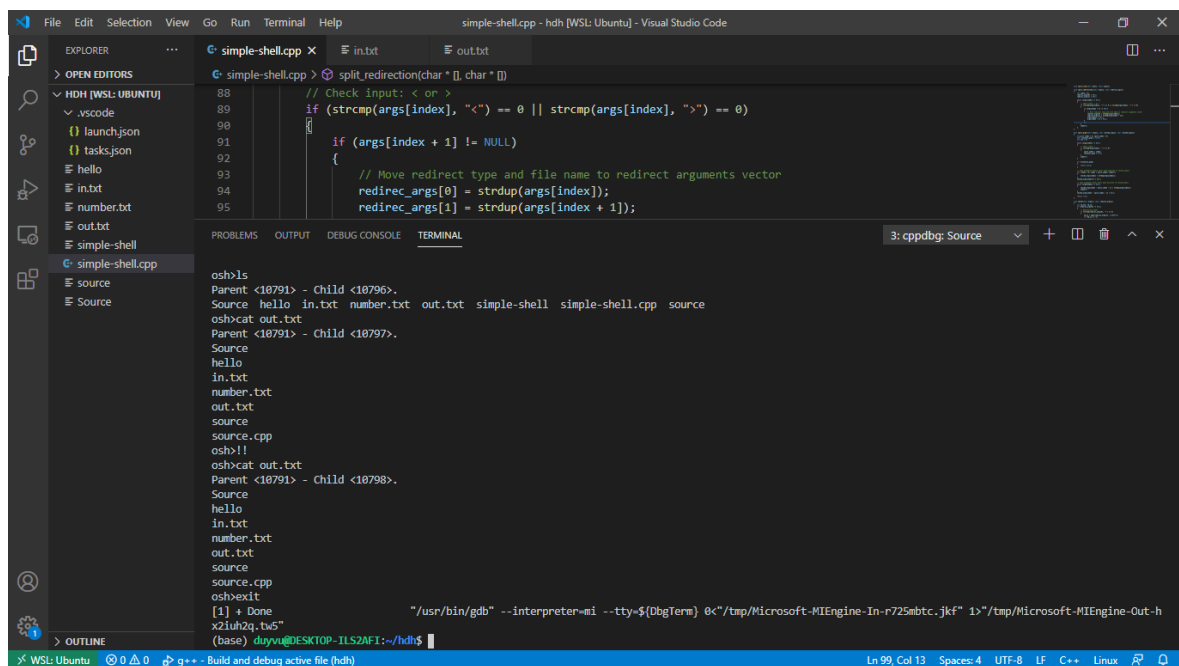
- Childprocess:



```
simple-shell.cpp
88 // Check input: < or >
89 if (strcmp(args[index], "<") == 0 || strcmp(args[index], ">") == 0)
90 {
91     if (args[index + 1] != NULL)
92     {
93         // Move redirect type and file name to redirect arguments vector
94         redirect_args[0] = strdup(args[index]);
95         redirect_args[1] = strdup(args[index + 1]);
96     }
97 }
```

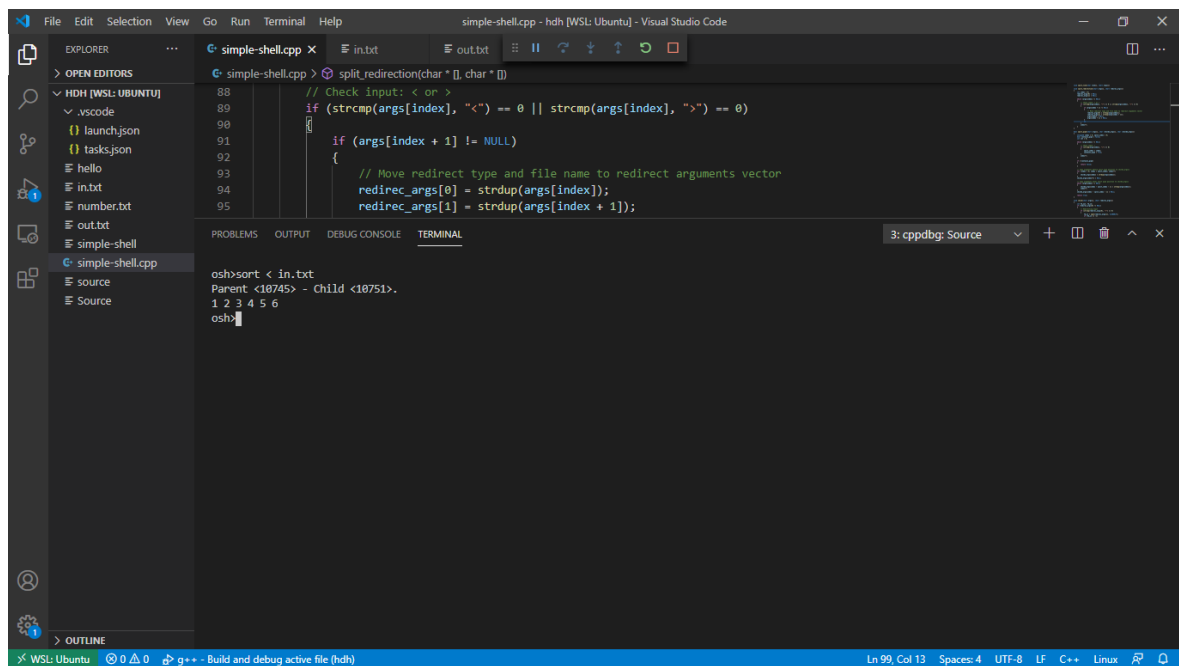
```
ossh>ls
Parent <18837> - Child <18843>.
Source hello in.txt number.txt out.txt simple-shell simple-shell.cpp source
ossh>cat out.txt
Parent <18837> - Child <18844>.
Source
hello
in.txt
number.txt
out.txt
source
source.cpp
ossh>
```

- History:



```
ossh>ls
Parent <18791> - Child <18796>.
Source hello in.txt number.txt out.txt simple-shell simple-shell.cpp source
ossh>cat out.txt
Parent <18791> - Child <18797>.
Source
hello
in.txt
number.txt
out.txt
source
source.cpp
ossh>!!
ossh>cat out.txt
Parent <18791> - Child <18798>.
Source
hello
in.txt
number.txt
out.txt
source
source.cpp
ossh>exit
[1] + Done
x2iuh2q.tw5"
(base) duyvu@DESKTOP-ILS2AF1:~/hdh$
```

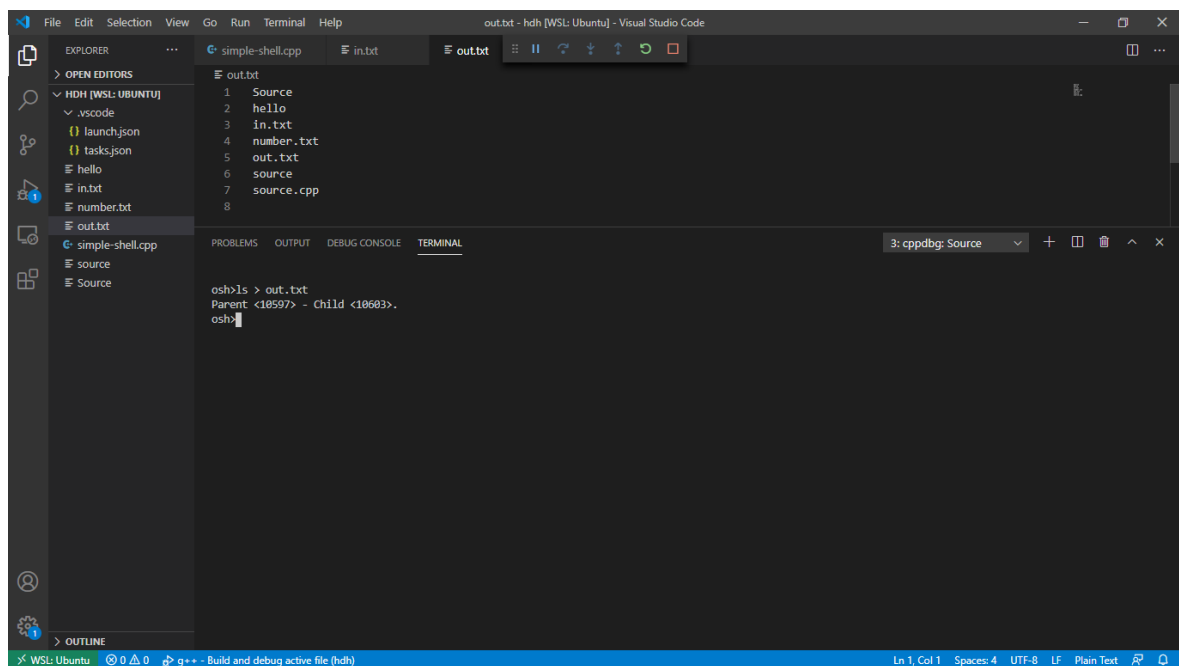
- Input:



```
simple-shell.cpp
88 // Check input: < or >
89 if (strcmp(args[index], "<") == 0 || strcmp(args[index], ">") == 0)
90 {
91     if (args[index + 1] != NULL)
92     {
93         // Move redirect type and file name to redirect arguments vector
94         redirec_args[0] = strdup(args[index]);
95         redirec_args[1] = strdup(args[index + 1]);
96     }
97 }
```

```
osh>sort < in.txt
Parent <18745> - Child <18751>.
1 2 3 4 5 6
osh>
```

- Output:



```
out.txt
1 Source
2 hello
3 in.txt
4 number.txt
5 out.txt
6 source
7 source.cpp
8
```

```
osh>ls > out.txt
Parent <18597> - Child <18683>.
osh>
```

- File:


```
88 // split_redirection(char * [], char * [])
89 if (strcmp(args[index], "<") == 0 || strcmp(args[index], ">") == 0)
90 {
91     if (args[index + 1] != NULL)
92     {
93         // Move redirect type and file name to redirect arguments vector
94         redirec_args[0] = strdup(args[index]);
95         redirec_args[1] = strdup(args[index + 1]);
```

total 156

-rwxr-xr-x	1	duyvu	duyvu	32224	Nov	8	22:36	Source
-rwxr-xr-x	1	duyvu	duyvu	30656	Nov	8	22:43	hello
-rw-r--r--	1	duyvu	duyvu	11	Nov	8	22:53	in.txt
-rw-r--r--	1	duyvu	duyvu	45	Nov	8	22:35	number.txt
-rw-r--r--	1	duyvu	duyvu	57	Nov	8	22:48	out.txt
-rw-r--r--	1	duyvu	duyvu	438	Nov	8	22:58	sda
-rwxr-xr-x	1	duyvu	duyvu	30672	Nov	8	22:59	simple-shell
-rw-r--r--	1	duyvu	duyvu	8897	Nov	8	22:55	simple-shell.cpp
-rwxr-xr-x	1	duyvu	duyvu	30664	Nov	8	22:47	source

(END)

TÀI LIỆU THAM KHẢO

Linux Programmer's Manual

<https://brennan.io/2015/01/16/write-a-shell-in-c/?fbclid=IwAR1Wt9nPDste4n3VIPe1Q3rjEHJ3WPlbmJIfAdOhgrZ1NRXr8D3iAppMet0>

<https://www.geeksforgeeks.org/making-linux-shell-c/>

<https://brennan.io/2015/01/16/write-a-shell-in-c/>

<https://github.com/pranav93y/myshell>