**ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN**

**KHOA CÔNG NGHỆ THÔNG TIN**

**Nguyễn Huy Tú**

**Nguyễn Duy Vũ**

**18120254**

**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](#_Toc55771600)

[**BÁO CÁO ĐỒ ÁN** 3](#_Toc55771601)

[**1.** **Báo cáo tiến độ.** 3](#_Toc55771602)

[**1.1.** **Thành viên nhóm.** 3](#_Toc55771603)

[**1.2.** **Mức độ hoàn thành.** 3](#_Toc55771604)

[**2.** **Báo cáo kĩ thuật.** 3](#_Toc55771605)

[**2.1.** **Mô tả bài toán.** 3](#_Toc55771606)

[**2.2.** **Các hàm.** 4](#_Toc55771607)

[**2.2.1.** **Các hàm phục vụ.** 4](#_Toc55771608)

[**2.2.2.** **Thực thi lệnh trong tiến trình con.** 4](#_Toc55771609)

[**2.2.3.** **Tạo lệnh history.** 4](#_Toc55771610)

[**2.2.4.** **Điều hướng input và output.** 4](#_Toc55771611)

[**2.2.5.** **Giao tiếp thông qua pipe.** 5](#_Toc55771612)

[**2.3.** **Hàm main.** 5](#_Toc55771613)

[**3.** **Các lệnh cơ bản được hỗ trợ.** 6](#_Toc55771614)

[**4.** **Các test-case.** 6](#_Toc55771615)

[**TÀI LIỆU THAM KHẢO** 10](#_Toc55771616)

# **BÁO CÁO ĐỒ ÁN**

1. **Báo cáo tiến độ.**
   1. **Thành viên nhóm.**

Nguyễn Huy Tú - 18120254

Nguyễn Duy Vũ - 18120264

* 1. **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% |

1. **Báo cáo kĩ thuật.**
   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

* 1. **Các hàm.**
     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.
  + 1. **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.
  + 1. **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.
  + 1. **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.
  + 1. **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 kí 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 tình con và dùng dup2() để sao chép file description output của tiến trình của 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.

* 1. **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 kí 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\* agrv[] chứa các tham số. Nếu cuối câu lệnh có ký tự ‘&’ thì câu lệnh đó sẽ được thực thi trong tiến trình con, khi tiếng 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().

1. **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…).

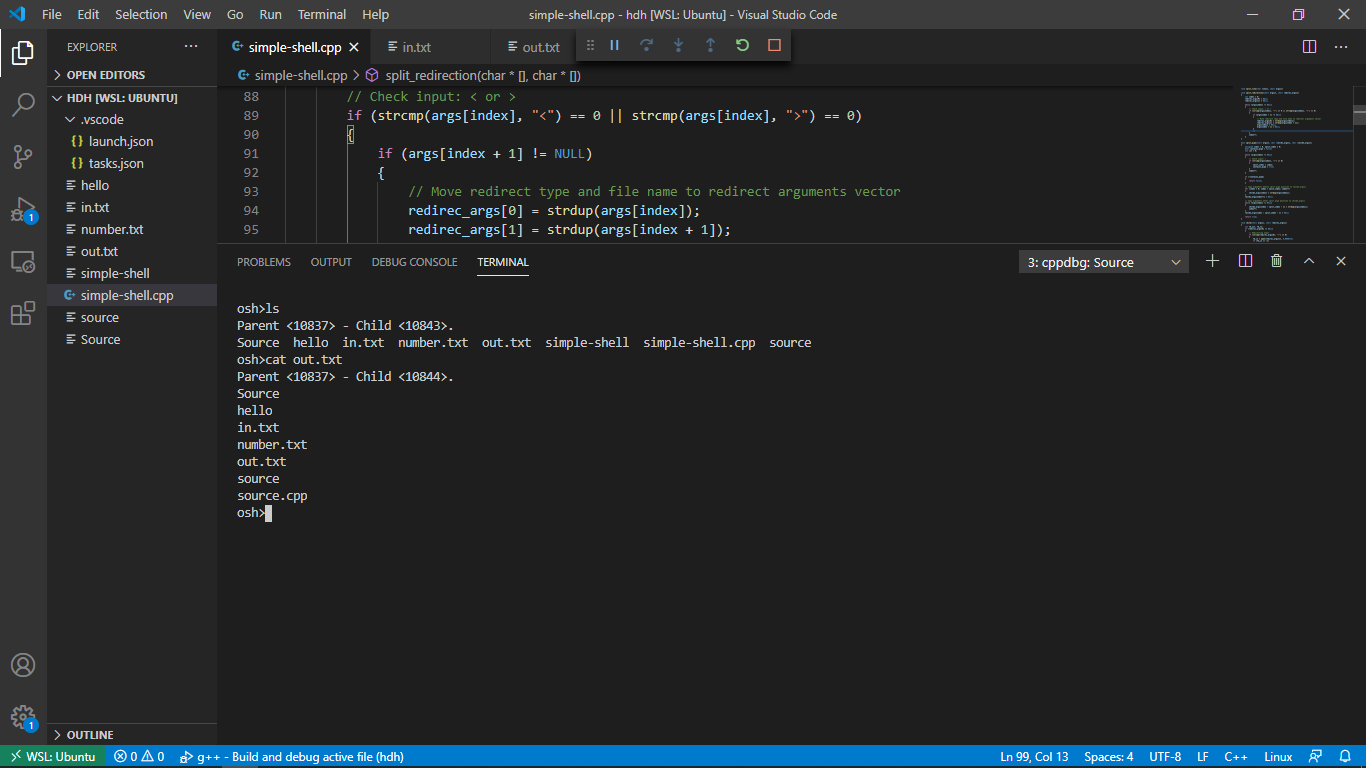
Câu lệnh “exit” dùng để thoát chương trình. Ngoài ra chương trình còn xử lí được các trường hợp nhập lệnh sai.

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

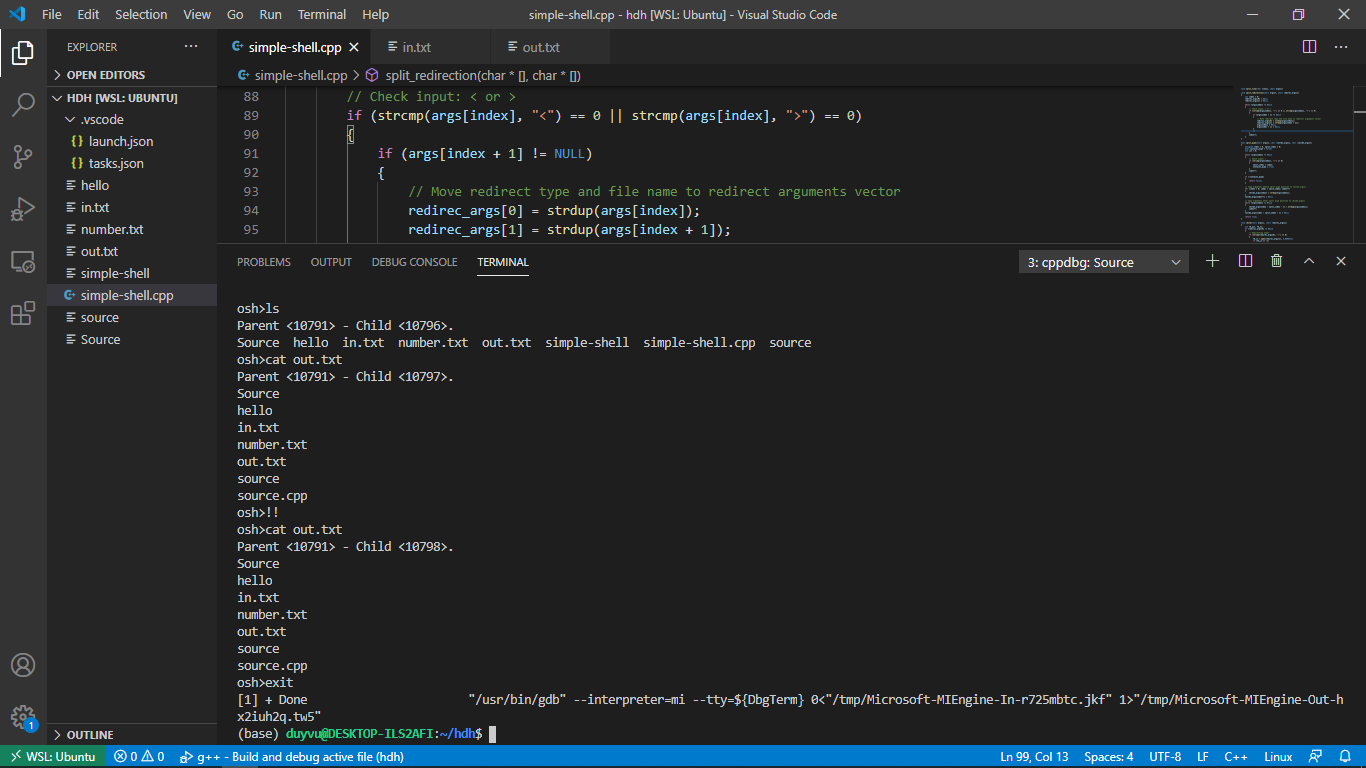
Câu lệnh “grep” để tìm kiếm chuỗi trong file.

1. **Các test-case.**

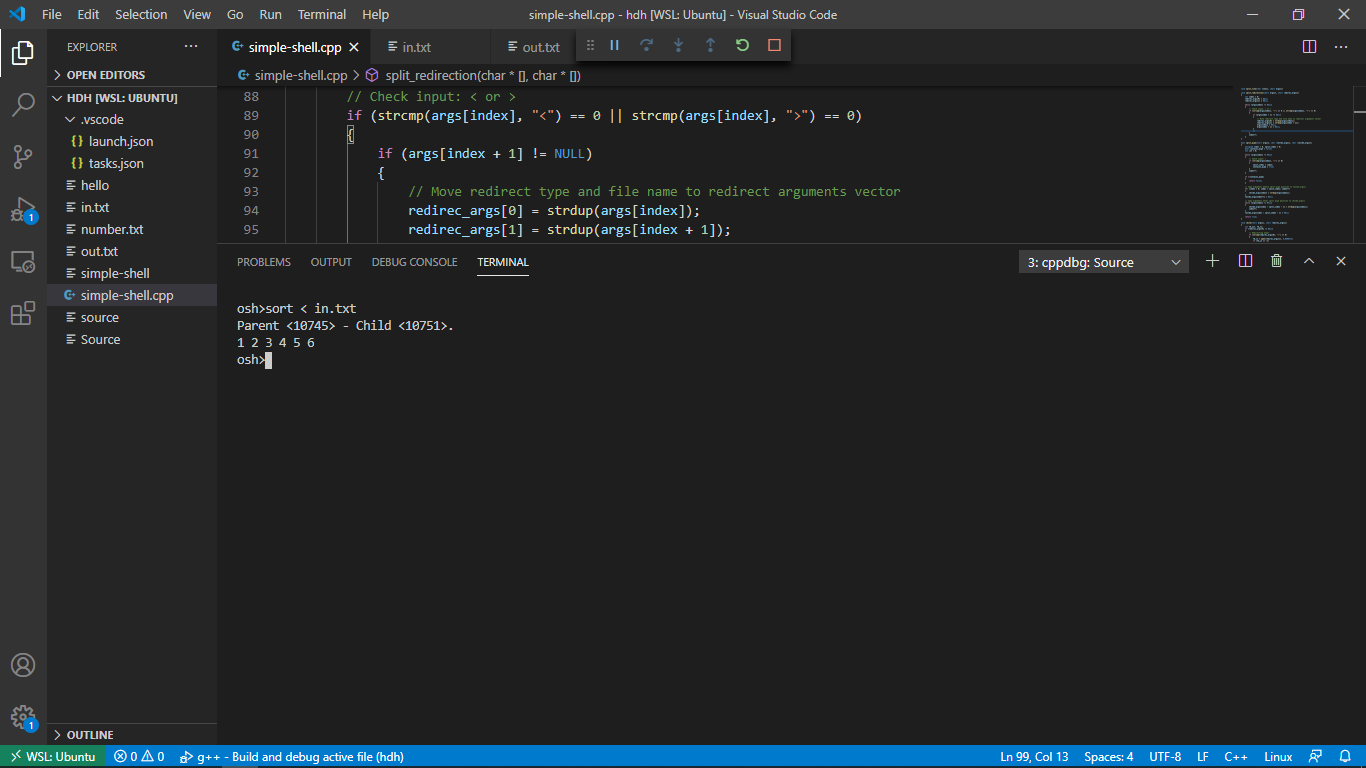
* Childprocess:



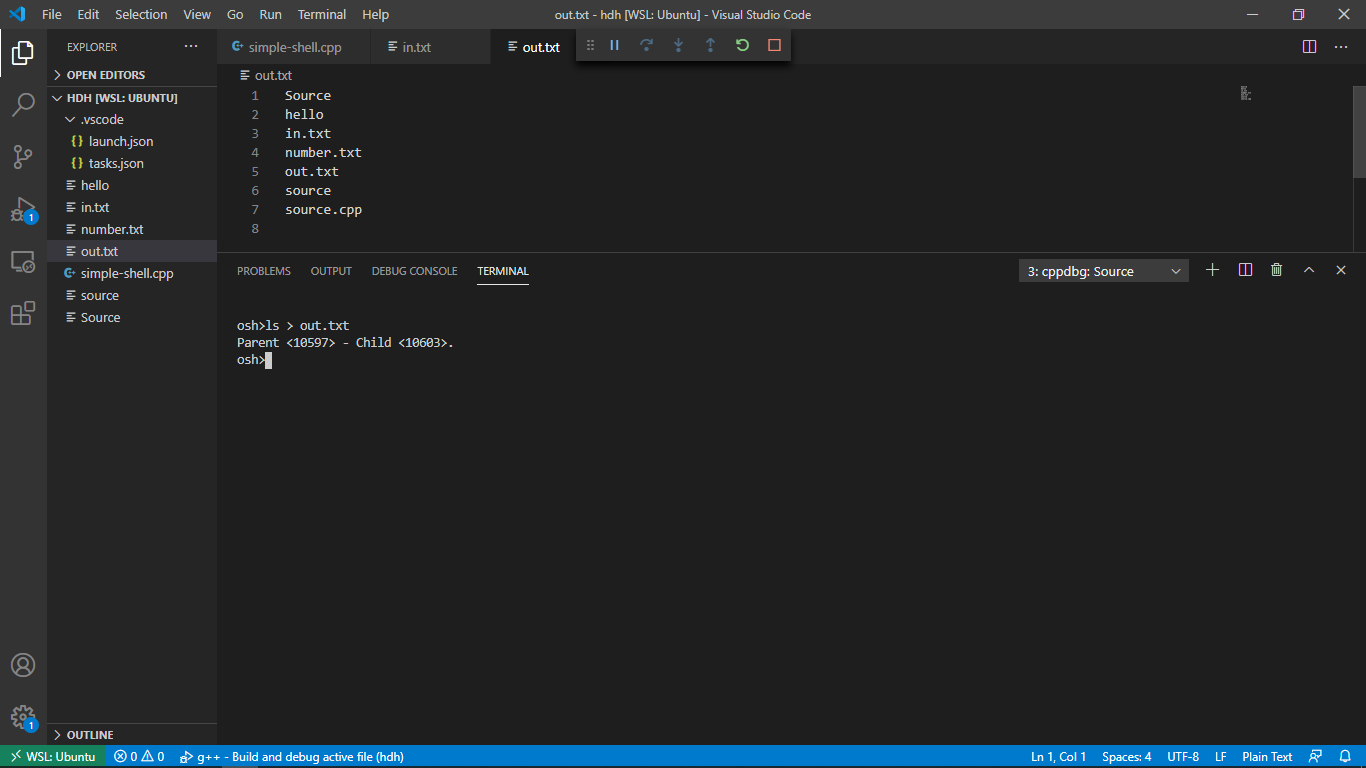
* History:



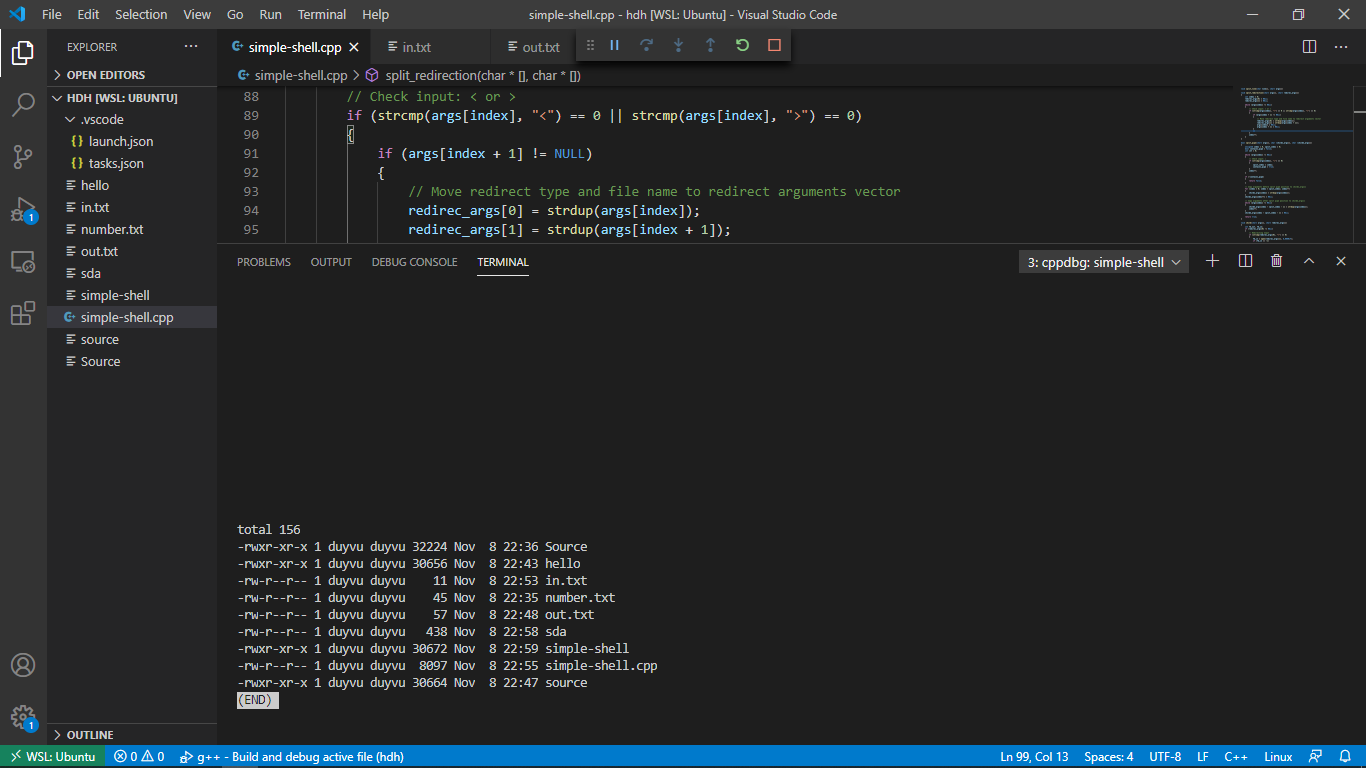
* Input:



* Output:



* Pile:



# **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