

ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN



ĐỒ ÁN HỆ ĐIỀU HÀNH PROJECT 1 – SIMPLE SHELL



♦ SINH VIÊN THỰC HIỆN ♦

Nguyễn Hữu Gia Trí	-	1712254
Huỳnh Thái Anh	-	1712272
Lê Hoài Bảo	-	1712286

MỤC LỤC

TỔ CHỨC CHƯƠNG TRÌNH.....	4
CÁCH HÀM TRONG CHƯƠNG TRÌNH.....	5
CHẠY VÀ KIỂM THỬ CHƯƠNG TRÌNH.....	6
MỨC ĐỘ HOÀN THIÊN YÊU CẦU?.....	12

THÔNG TIN THÀNH VIÊN VÀ PHÂN CÔNG

1. Thông tin thành viên

Tên	MSSV	Email
Nguyễn Hữu Gia Trí	1712254	1712254@student.hcmus.edu.vn
Huỳnh Thái Anh	1712272	1712272@student.hcmus.edu.vn
Lê Hoài Bảo	1712286	1712286@student.hcmus.edu.vn

2. Phân công

Tên	Công việc	Mức độ hoàn thành
Nguyễn Hữu Gia Trí	<ul style="list-style-type: none">Tổ chức chương trìnhViết chương trình chínhHỗ trợ viết báo cáo	100%
Huỳnh Thái Anh	<ul style="list-style-type: none">Tìm hiểu các lệnh trong linux và các hàm fork(),exec(),wait(),dup2(),pipe().Hỗ trợ viết chương trìnhViết báo cáo	100%
Lê Hoài Bảo	<ul style="list-style-type: none">Tìm hiểu các lệnh trong linux và các hàm fork(),exec(),wait(),dup2(),pipe().Hỗ trợ viết chương trìnhViết báo cáo	100%

TỔ CHỨC CHƯƠNG TRÌNH

- **Yêu cầu của đề án :**

Xây dựng chương trình để thực hiện các lệnh như một giao diện shell và dùng các hàm hỗ trợ : **fork()** **exec()**, **wait()**, **dup2()**, **pipe()** .

- Tạo tiến trình con và thực hiện lệnh trong tiến trình con
- Cung cấp tính năng lịch sử (nhớ và thực hiện lệnh trước đó)
- Hỗ trợ redirecting input và output (“<” và “>”)
- Tiến trình cha và con giao tiếp thông qua **pipe()**
- Nếu thêm ‘&’ ở cuối lệnh sẽ cho phép tiến trình con chạy dưới background hoặc đồng thời với tiến trình cha.

- **Ý nghĩa các hàm :**

- **fork()**: Tạo một tiến trình con . Nếu không có yêu cầu thực hiện nào khác , tiến trình con này sẽ thực hiện cùng yêu cầu với tiến trình cha.
- **execvp(char *command, char *params[])** : thực hiện lệnh command với các tham số params.
- **dup2(fd, STDOUT_FILENO)**: nhân đôi file description , sau đó chuyển file description được nhân thành đầu ra tiêu chuẩn . Có nghĩa là : tất cả các output của terminal sẽ được ghi ra file .
- **pipe(int fd[2])**: Tạo ra một pipe để liên lạc giữa hai tiến trình . Dữ liệu trả về của fd[0] là dữ liệu đầu vào của fd[1].

- **Cấu trúc chương trình :**

1. Định dạng chuỗi nhập vào của người dùng bao gồm xóa các ký tự trắng dư thừa và các ký tự lỗi.
2. Chương trình bao gồm 4 chế độ thông qua việc kiểm tra chuỗi nhập vào (**input**) của người dùng :
 - **INPUT_MODE** : Chuỗi nhập vào có ký tự ‘<’ .
Ở chế độ này , lấy dữ liệu từ file làm input cho lệnh ở command line
 - **OUTPUT_MODE**: Chuỗi nhập vào có ký tự ‘>’
Ở chế độ này , dữ liệu đầu ra của lệnh ở command line sẽ được ghi vào file
 - **PIPE_MODE**: Chuỗi nhập vào có ký tự ‘|’
Ở chế độ này, output của lệnh đầu sẽ là input của lệnh sau đó .
 - **NORMAL_MODE** : Chuỗi nhập vào không bao gồm các trường hợp trên .
Ở chế độ này , chương trình chạy theo các lệnh của linux
3. Chương trình cung cấp tính năng cho phép người dùng thực hiện lại lệnh đã nhập trước đó bằng cách nhập “!!” .

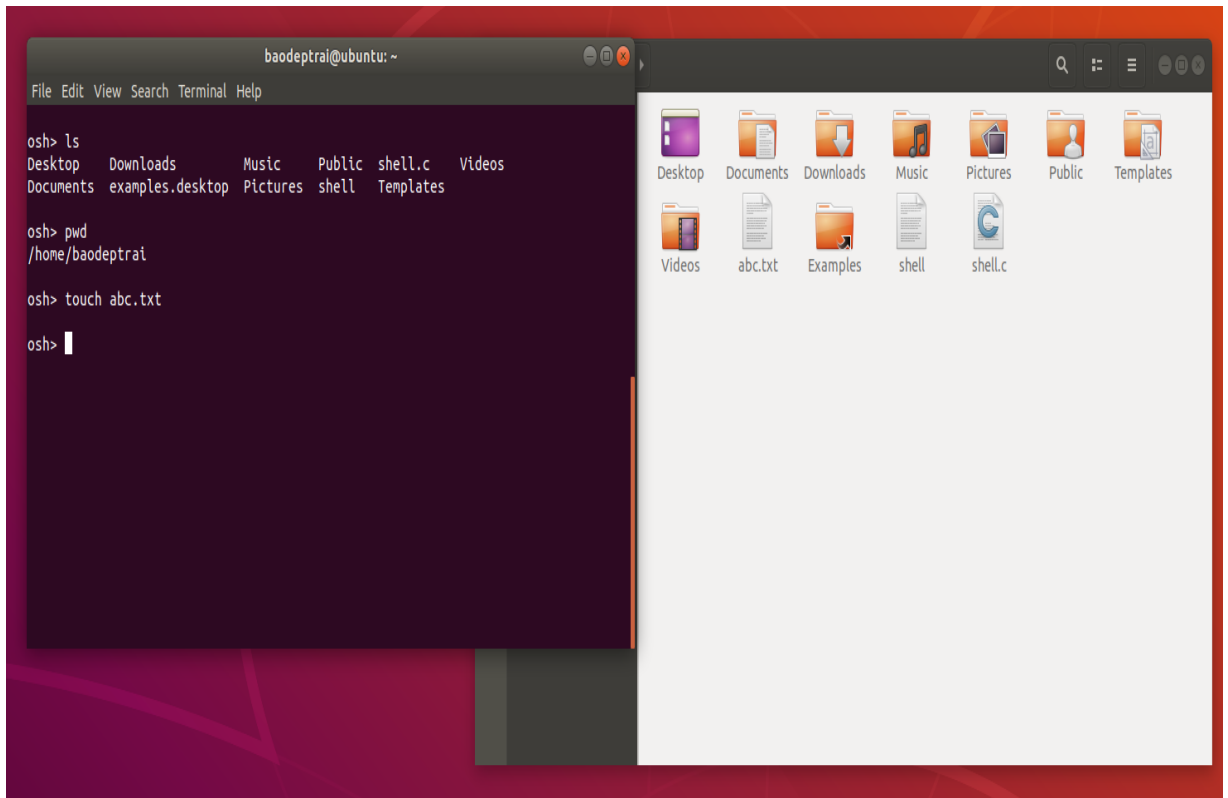
CÁC HÀM TRONG CHƯƠNG TRÌNH

1. `string formatString(string input)`
Xử lý chuỗi nhập vào (string `input`) của người dùng, xóa các khoảng trắng dư thừa và các ký tự gây nhiễu.
2. `vector<string> split(string str, char delim)`
Tách chuỗi (string `str`) thành các tokens theo ký tự tùy chọn (char `delim`). Trả về vector chứa các tokens.
3. `vector<string> splitSpecialCommand(string input, char delim)`
Dùng cho `INPUT_MODE`, `OUTPUT_MODE`, `PIPE_MODE` với char `delim` lần lượt là '<' '>' '|'.
Tách chuỗi (string `input`) thành các tokens trước (char `delim`) và sau (char `delim`). Trả về vector chứa các tokens.
4. `char **vectorToCharArray(vector<string> input)`
Chuyển kiểu dữ liệu vector thành kiểu dữ liệu char**
5. `void handle(string &input)`
Hàm xử lý chính của chương trình.
6. `void excute(bool isRunning)`
Hàm chạy chương trình.

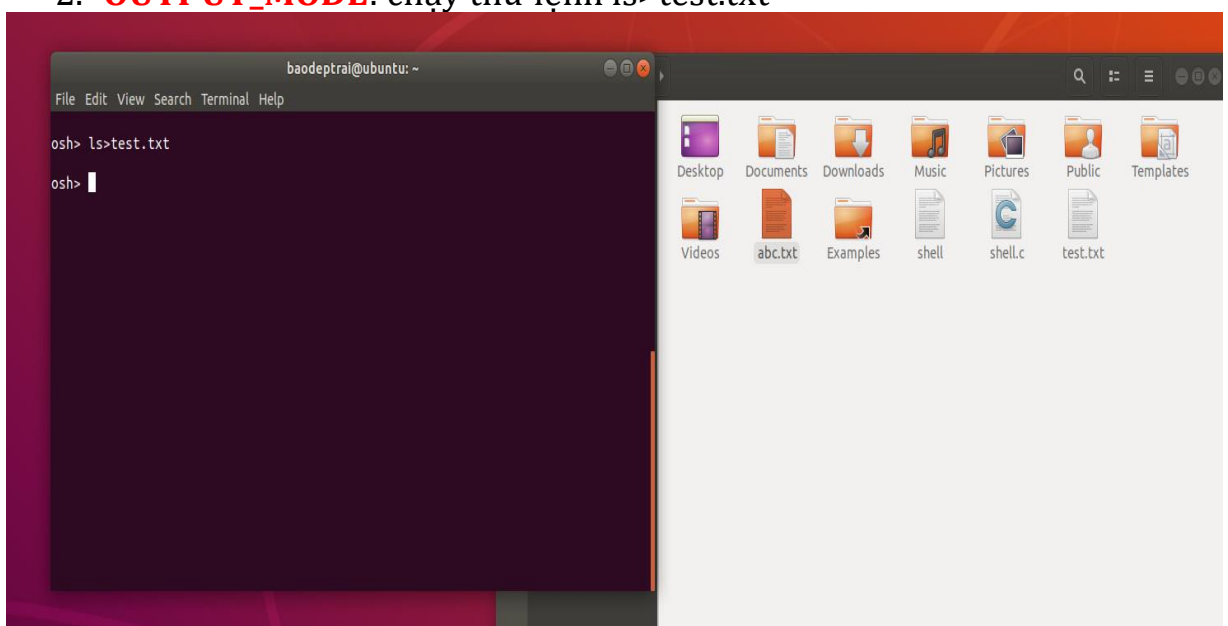
CHẠY VÀ KIỂM THỬ CHƯƠNG TRÌNH

1. **NORMAL_MODE** :

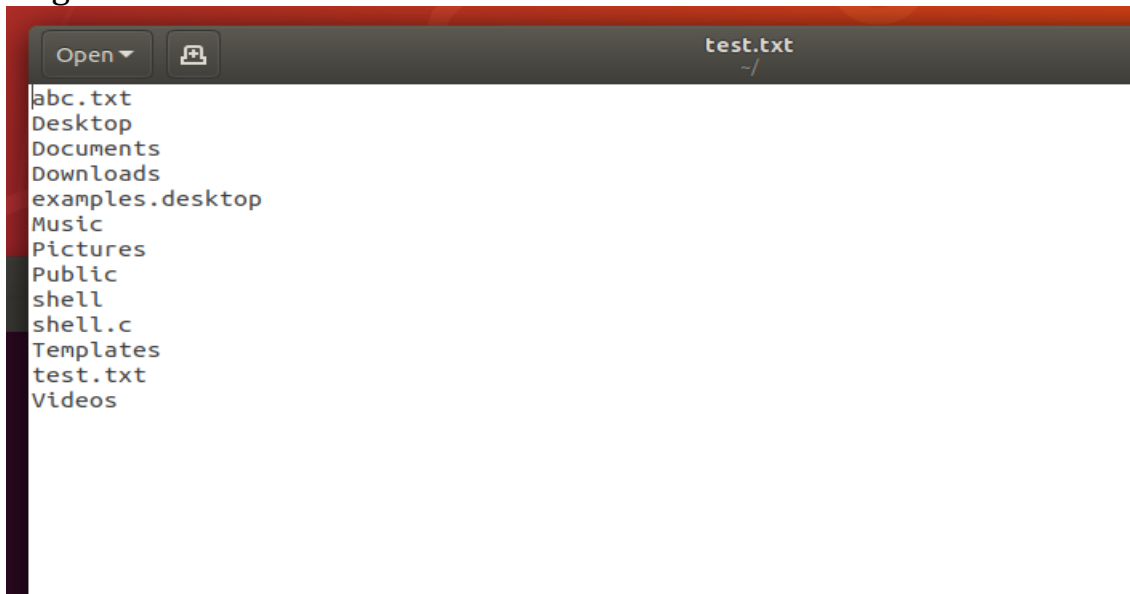
Thực hiện lần lượt gõ các lệnh ls, pwd, touch kết quả ra được như hình dưới.



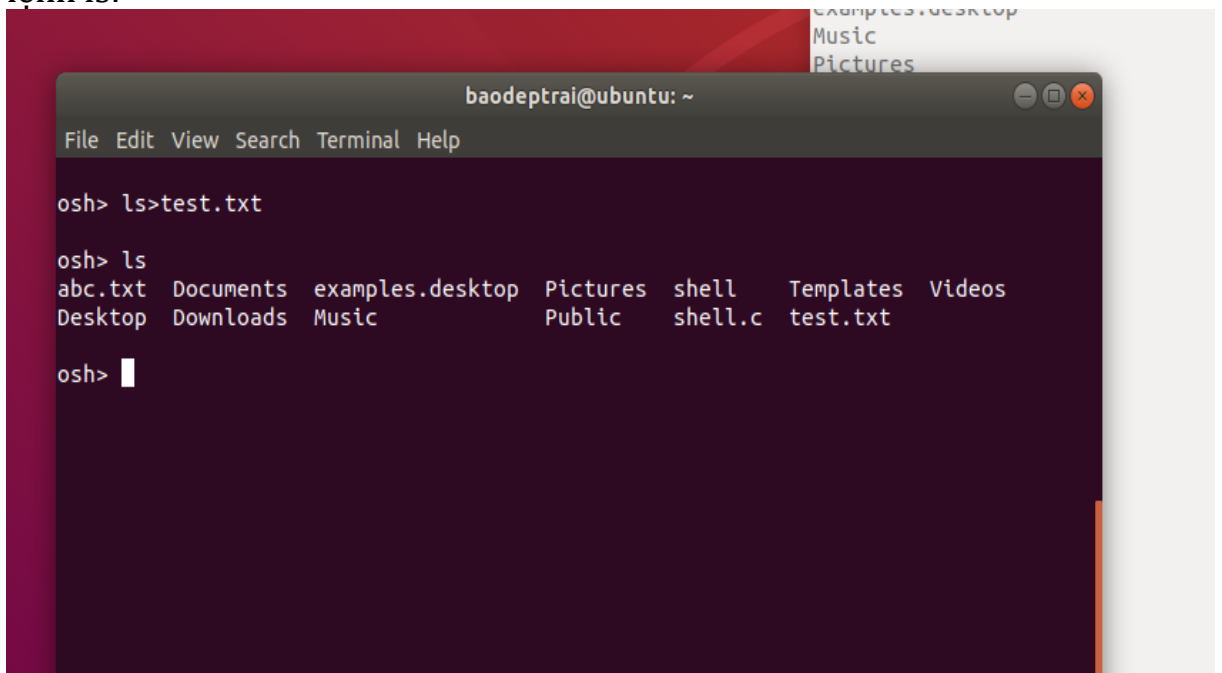
2. **OUTPUT_MODE**: chạy thử lệnh ls>test.txt



Một file test.txt đã được tạo ra. Tiếp tục kiểm tra xem bên trong test.txt có gì:



Kết quả thu được giống với kết quả đưa ra màn hình command của lệnh ls:

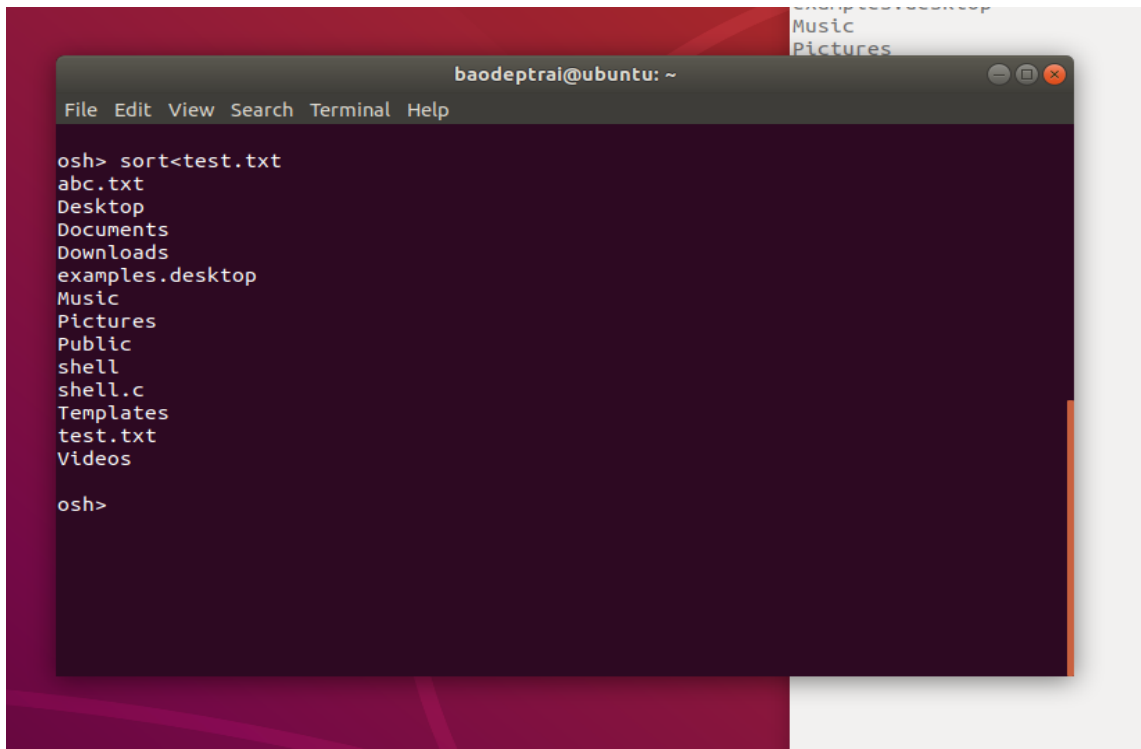


3. INPUT_MODE

Ví dụ : Chạy lệnh sort<test.txt

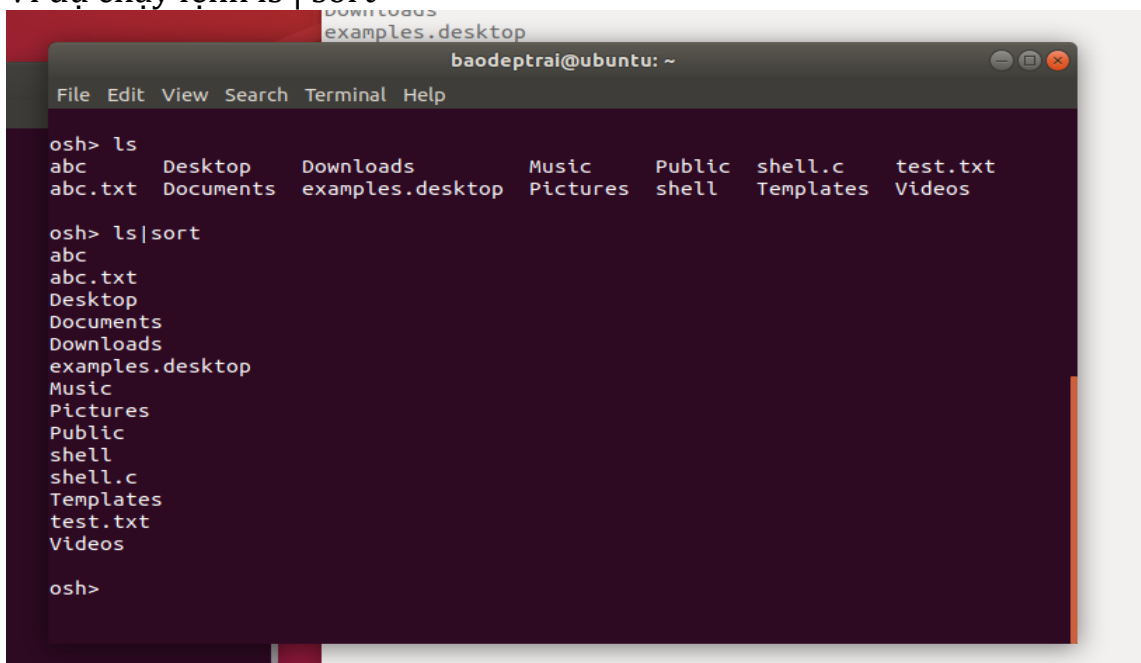
Dữ liệu đầu vào của lệnh trên là file test.txt

Lệnh sort sẽ sắp xếp các dãy kí tự có trong file test.txt sau đó in ra màn hình kết quả:



```
baodeptrai@ubuntu: ~  
File Edit View Search Terminal Help  
osh> sort<test.txt  
abc.txt  
Desktop  
Documents  
Downloads  
examples.desktop  
Music  
Pictures  
Public  
shell  
shell.c  
Templates  
test.txt  
Videos  
osh>
```

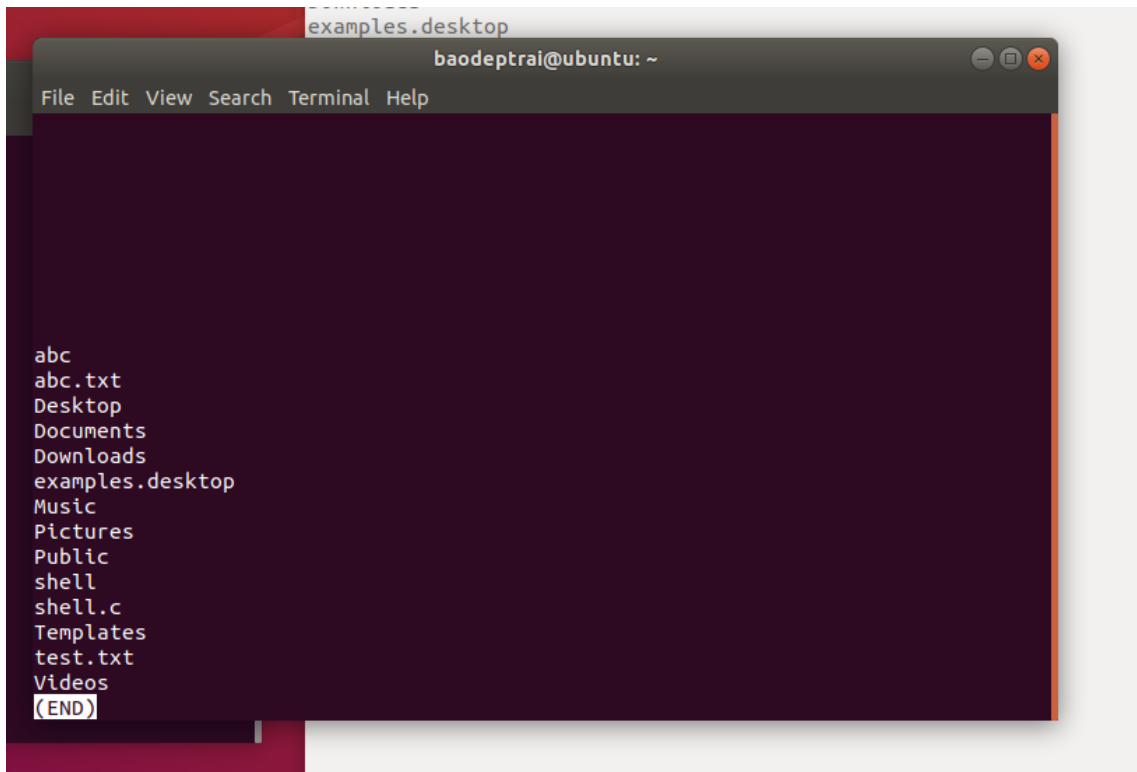
4. PIPE_MODE: Ví dụ chạy lệnh ls | sort



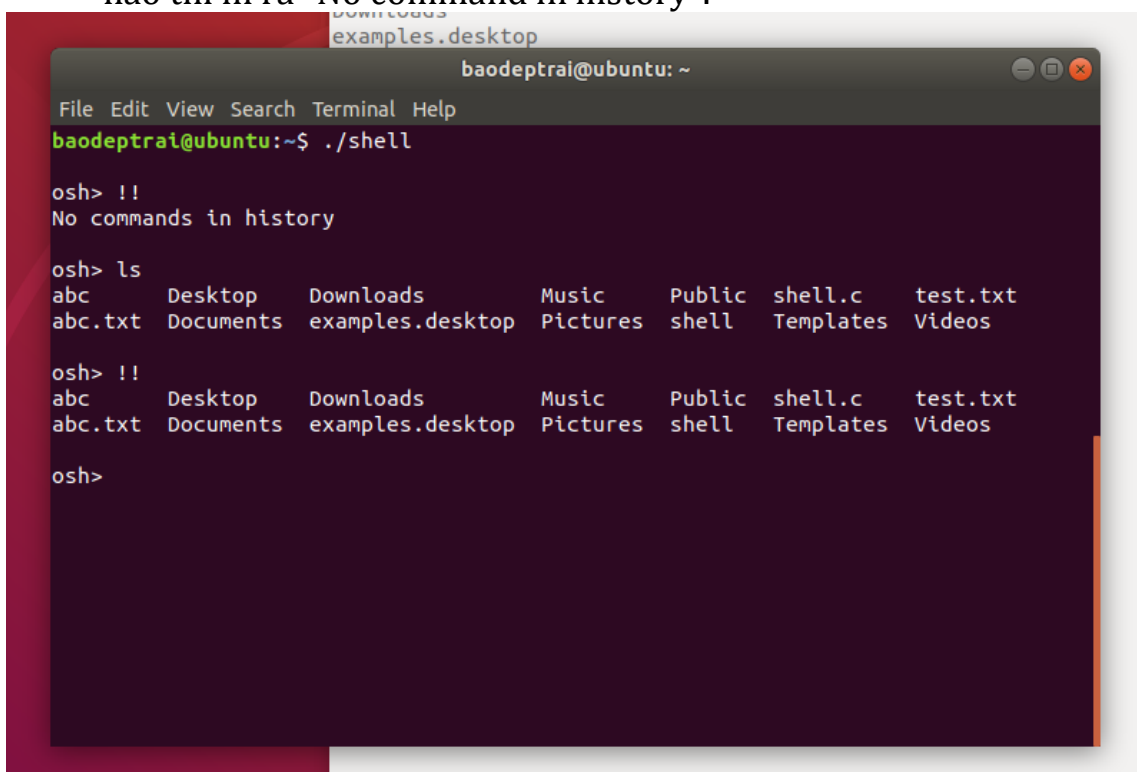
```
baodeptrai@ubuntu: ~  
File Edit View Search Terminal Help  
osh> ls  
abc      Desktop  Downloads  Music    Public  shell.c  test.txt  
abc.txt  Documents examples.desktop Pictures  shell   Templates Videos  
osh> ls|sort  
abc  
abc.txt  
Desktop  
Documents  
Downloads  
examples.desktop  
Music  
Pictures  
Public  
shell  
shell.c  
Templates  
test.txt  
Videos  
osh>
```

Lệnh ls liệt kê các file hiện hành tại thư mục hiện tại sau đó sắp xếp chúng bằng lệnh sort.

Tương tự với ls | less

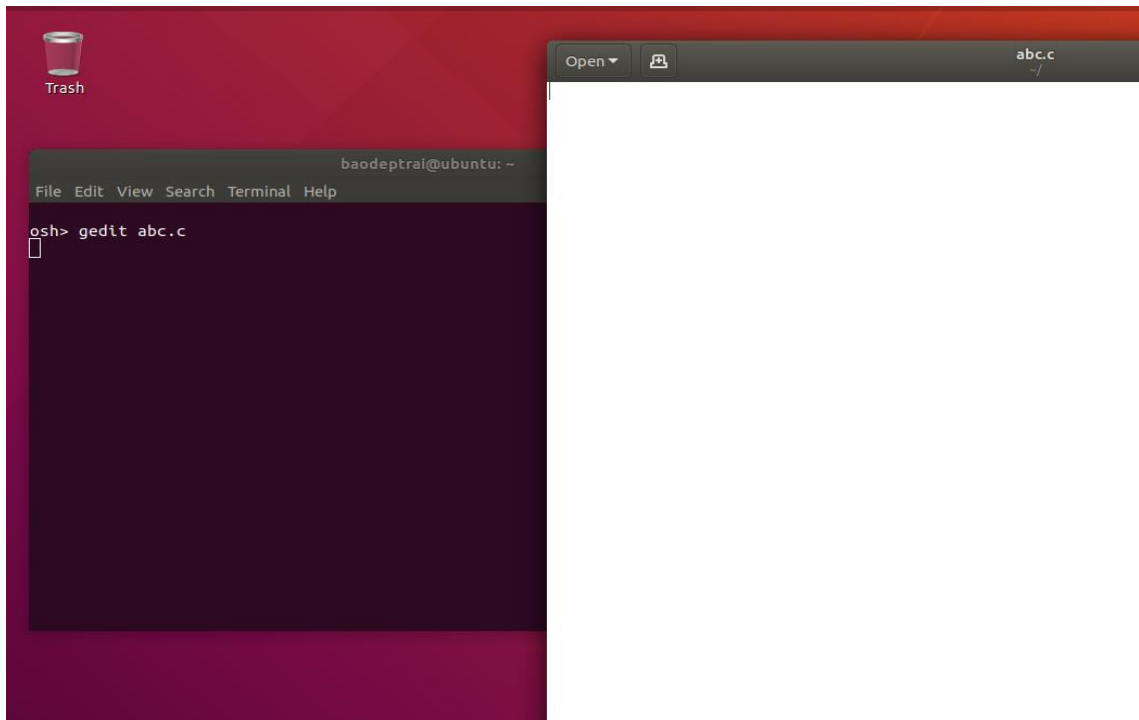


5. **!!** thực hiện lệnh trước đó đã nhập. Nếu trước đó không có lệnh nào thì in ra “No command in history”.

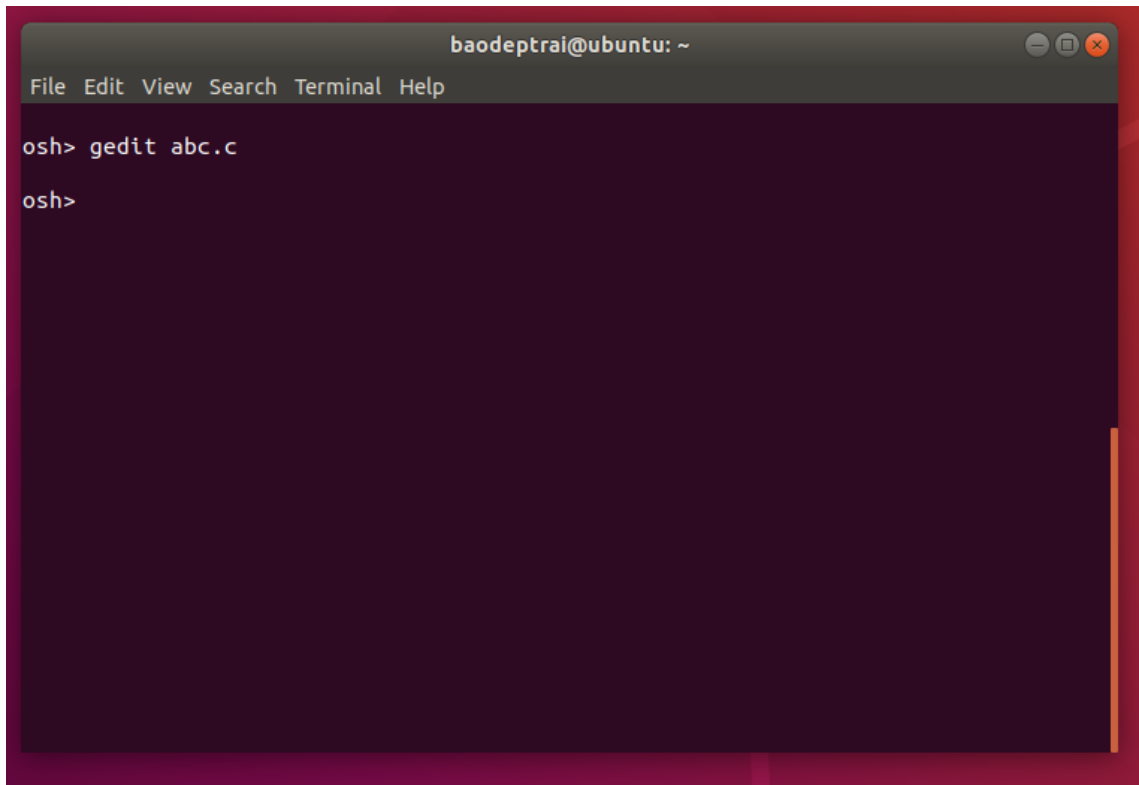


6. Thực hiện lệnh có dấu **&**:

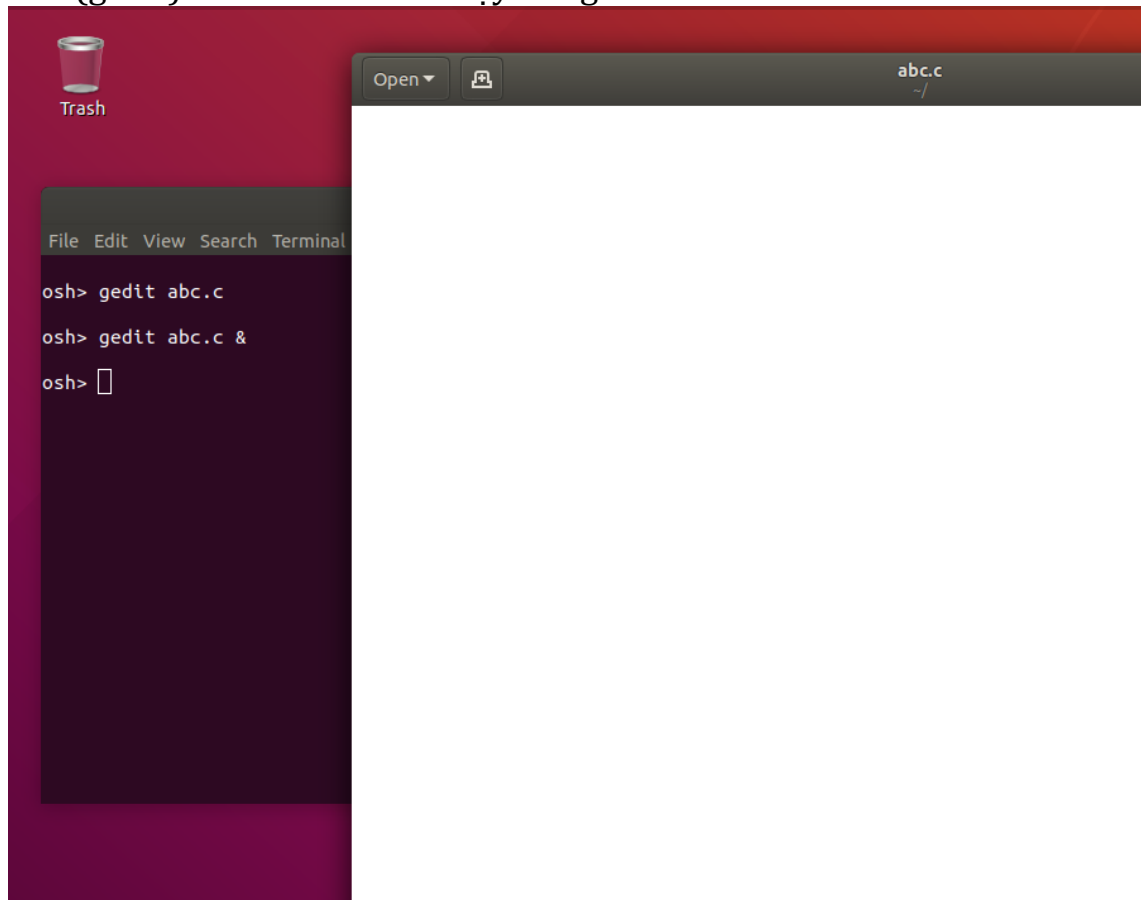
Khi chạy không có dấu &
Tiến trình cha chờ tiến trình con (gedit) hoàn thành mới tiếp tục chạy.



Khi tắt cửa sổ abc.c:



Khi chạy có dấu **&**: Tiến trình cha thực hiện không chờ đợi tiến trình con (gedit) hoàn thành mà chạy đồng thời với tiến trình con.



MỨC ĐỘ HOÀN THIỆN YÊU CẦU ?

(100%)