

SimpleShell_Project 2_Report

Group members:

- 1751064 Nguyen Hoang Gia
- 1751063 Pham Bao Duy

GitHub Repository link:

<https://github.com/nhgia/SimpleShell>

I. README.md

Prerequisite

- Ubuntu / Linux. (Recommended Ubuntu 18.04)
- Download or clone the project.
- Install essential components by running:

```
sudo apt-get install build-essential
```

- Change directory to the project folder.

Install: Compile the C file into output and runnable

- Compile through Makefile, run

```
make
```

- Run the shell

```
./simple-shell
```

SimpleShell commands and features

1. Single command

- List sub-folder & sub-item

```
ls
```

- Current directory

```
pwd
```

- Ping website

```
ping -i 5 -w 10 -W 10 google.com
```

- ...

2. Check history of run commands

- Using double exclamation mark "!!" for checking history

```
!!
```

3. Concurrency commands

- Add an ampersand mark after the command with whitespace required.

```
ping -i 5 -w 10 -W 10 google.com &
```

4. I/O redirection

- Redirect the output of a command to a file and input from a file to a command.

```
ls > myFile.txt  
cat < myFile.txt
```

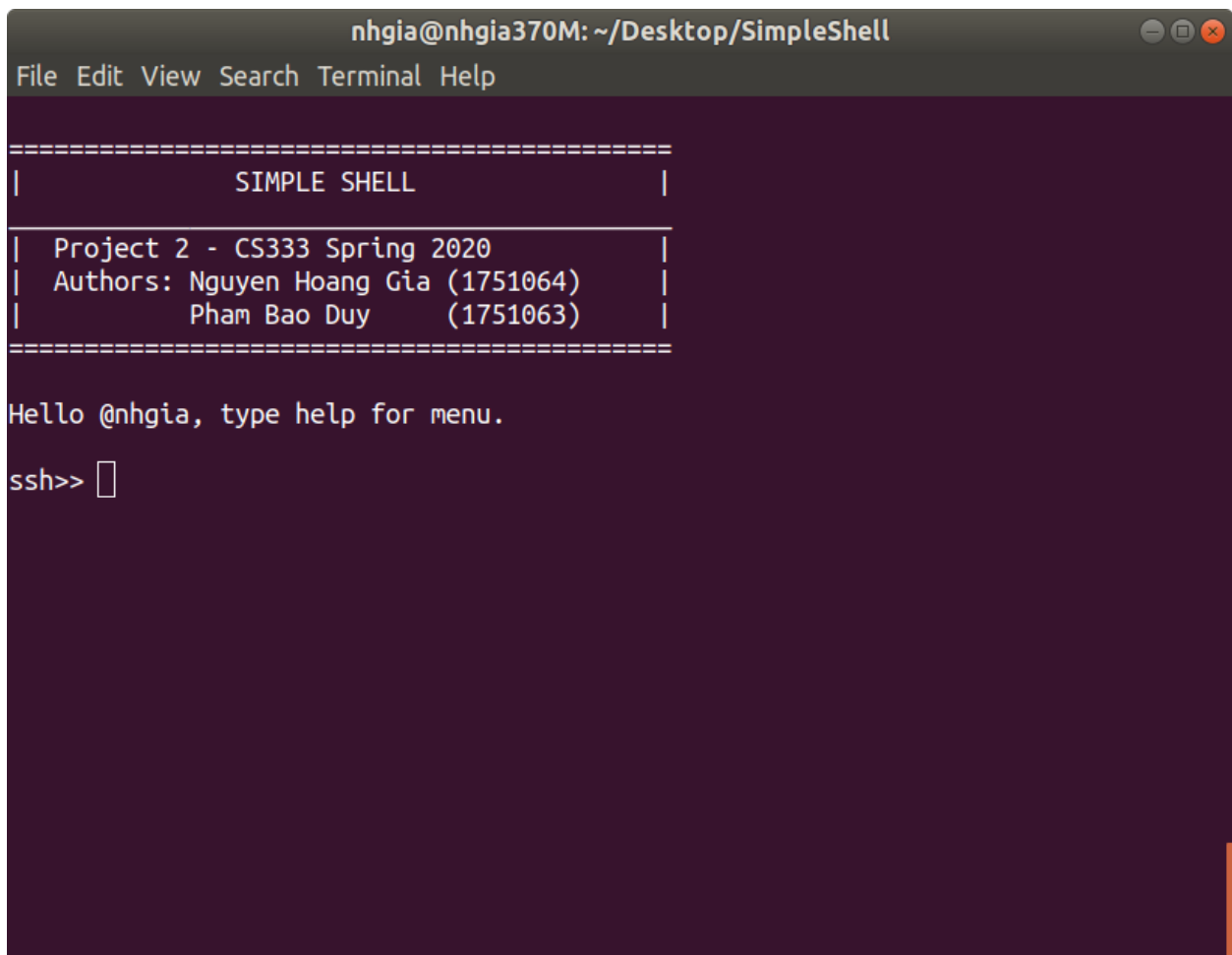
5. Communication via a Pipe

- Output of one command to serve as input to another using a pipe.
- For example: list sub-folder and sub-item in current directory, just display which has character "a" in the name.

```
ls | grep a
```

II. Result report

1. Main (after running ./simple-shell)



```
nhgia@nhgia370M: ~/Desktop/SimpleShell  
File Edit View Search Terminal Help  
=====
```

SIMPLE SHELL
Project 2 - CS333 Spring 2020
Authors: Nguyen Hoang Gia (1751064)
Pham Bao Duy (1751063)

```
=====
```

Hello @nhgia, type help for menu.

ssh>>

2. Single command with fork() child process

```
nhgia@nhgia370M: ~/Desktop/SimpleShell
File Edit View Search Terminal Help

=====
|               SIMPLE SHELL               |
=====
| Project 2 - CS333 Spring 2020             |
| Authors: Nguyen Hoang Gia (1751064)       |
|           Pham Bao Duy (1751063)         |
=====

Hello @nhgia, type help for menu.

ssh>> ls
Makefile  README.md  simple-shell  simple-shell.c
ssh>> 
```

3. History by “!!”

```
nhgia@nhgia370M: ~/Desktop/SimpleShell
File Edit View Search Terminal Help

ssh>> ls
Makefile  README.md  simple-shell  simple-shell.c
ssh>> !!
Makefile  README.md  simple-shell  simple-shell.c
ssh>> 
```

4. Concurrent commands

```
nhgia@nhgia370M: ~/Desktop/SimpleShell
File Edit View Search Terminal Help
ssh>> ping -i 5 -w 10 -W 10 google.com &
ssh>> PING google.com(2404:6800:4003:c03::66 (2404:6800:4003:c03::66)) 56 data b
ytes
64 bytes from 2404:6800:4003:c03::66 (2404:6800:4003:c03::66): icmp_seq=1 ttl=43
time=51.2 ms
ls -l
total 44
-rw-rw-r-- 1 nhgia nhgia 196 Thg 6 6 21:56 Makefile
-rw-r--r-- 1 nhgia nhgia 1434 Thg 6 6 22:23 README.md
-rwxr-xr-x 1 nhgia nhgia 22272 Thg 6 7 02:01 simple-shell
-rw-r--r-- 1 nhgia nhgia 8547 Thg 6 7 02:01 simple-shell.c
ssh>> 64 bytes from 2404:6800:4003:c03::66 (2404:6800:4003:c03::66): icmp_seq=2
ttl=43 time=51.0 ms
pwd
/home/nhgia/Desktop/SimpleShell
ssh>>
--- google.com ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 5005ms
rtt min/avg/max/mdev = 51.021/51.125/51.230/0.249 ms
ls
...
Makefile README.md simple-shell simple-shell.c
ssh>> 
```

5. Output redirection

```
nhgia@nhgia370M: ~/Desktop/SimpleShell
File Edit View Search Terminal Help
ssh>> ls -l > myFile.txt
ssh>> 
```

```
myFile.txt
~/Desktop/SimpleShell
total 44
-rw-rw-r-- 1 nhgia nhgia 196 Thg 6 6 21:56 Makefile
-rwxr-xr-x 1 nhgia nhgia 0 Thg 6 7 02:17 myFile.txt
-rw-r--r-- 1 nhgia nhgia 1434 Thg 6 6 22:23 README.md
-rwxr-xr-x 1 nhgia nhgia 22272 Thg 6 7 02:01 simple-shell
-rw-r--r-- 1 nhgia nhgia 8547 Thg 6 7 02:01 simple-shell.c
```

6. Input redirection

```
nhgia@nhgia370M: ~/Desktop/SimpleShell
File Edit View Search Terminal Help
ssh>> ls -l > myFile.txt
ssh>> cat < myFile.txt
*** IOREDIRECT: File not found.
ssh>> cat < myFile.txt
total 44
-rw-rw-r-- 1 nhgia nhgia 196 Thg 6 6 21:56 Makefile
-rwxr-xr-x 1 nhgia nhgia 0 Thg 6 7 02:17 myFile.txt
-rw-r--r-- 1 nhgia nhgia 1434 Thg 6 6 22:23 README.md
-rwxr-xr-x 1 nhgia nhgia 22272 Thg 6 7 02:01 simple-shell
-rw-r--r-- 1 nhgia nhgia 8547 Thg 6 7 02:01 simple-shell.c
ssh>> 
```

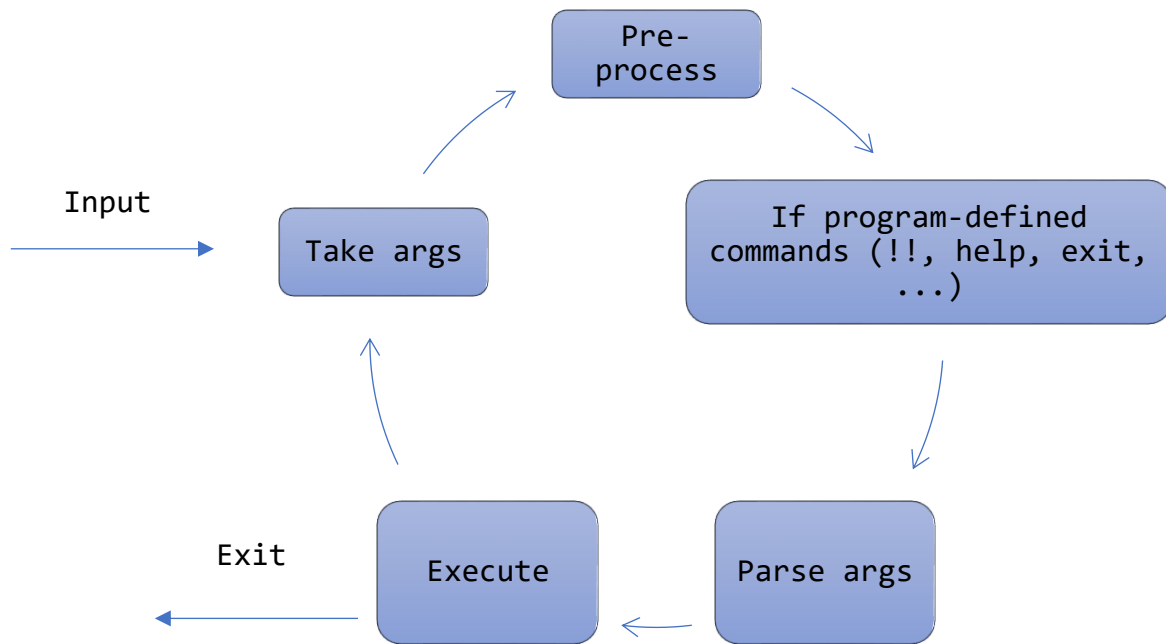
7. Communication via a Pipe

```
nhgia@nhgia370M: ~/Desktop/SimpleShell
File Edit View Search Terminal Help
ssh>> ls -l | grep m
-rwxr-xr-x 1 nhgia nhgia 297 Thg 6 7 02:17 myFile.txt
-rw-r--r-- 1 nhgia nhgia 1434 Thg 6 6 22:23 README.md
-rwxr-xr-x 1 nhgia nhgia 22272 Thg 6 7 02:01 simple-shell
-rw-r--r-- 1 nhgia nhgia 8547 Thg 6 7 02:01 simple-shell.c
ssh>> 
```

III. Code review

Overall, the program at run time works like normal C program but take input argument from user entered via Terminal.

The main work-flow of simple-shell can be illustrate by this:



- Pre-process: Flush the input-stream, get new one from `fgetc()`, add `'\0'` for end of user input.
- Next, check whether the input (first argument or `args[0]`) is special case or not. This includes show history ("`!!`" command), help menu, exit program, clear screen.
- Function `parse()` takes an input line and returns a zero-terminated array of char pointers, each of which points to a zero-terminated character string. This function loops until a binary zero is found, which means the end of the input line "command" is reached. If the current character of "command" is not a binary zero, `parse()` skips all white spaces and replaces them with binary zeros so that a string is effectively terminated. Once `parse()` finds a non-white space, the address of that location is saved to the current position of "args" and the index is advanced. Then, `parse()` skips all non-whitespace characters. This process repeats until the end of string "command" is reached and at that moment "args" is terminated with a zero.
- The `execute()` function: After get everything from parse, now the argument passed into `execute()` is an array of character, which element is a part of input. For example: `ls -l | grep a` will be `args[0] = "ls"`, `args[1] = "-l"`, `args[2] = "|"`, `args[3] = "grep"`, `args[4] = "a"`. Next, it continues to check whether it is pipe (contains "|") or IO redirect (contains "<" or ">"),...
- After got what kind of command from input, we fork child process to execute input command through `fork()` function. Thanks for the `pid_t` data type and also depending on which kind of command, we separate child process logic:

(after fork 1st child process)

pid_t	fork() > 0 At parent process	fork() == 0 At child process		fork() < 0 Error
Single command with child process	Wait for child process to be killed.	Execute command.		Exit()
Concurrent	Return to parent process immediately, no need to wait.			
IO Redirect	Wait for child process to be finished.	<ul style="list-style-type: none">• If output: open file, redirection using dup2() function (send output to opened file).• If input: open file, redirection using dup2() function (send file to input).		
Pipe	Execute 1 st command.	#1 child	#2 child	
		Fork 2 nd child process. Write to child #2.	Read from child #1. Execute 2 nd command.	
		Wait for both child process to be killed.		

- END -