Assignment 1: Building a Command Line Interpreter with C Programming

User Documentation

Ray Krishardi Layadi - 26445549

Introduction

Shell prompts the user for commands and interprets the given command if the given command is one of the commands supported by the shell (clear, cd, ls, echo, create, copy, search, run, halt, help, and quit). The "help" command can be used to display the user manual which includes information on the shell's built-in commands and their corresponding syntax. Finally, to exit the shell, the user can use the "quit" command.
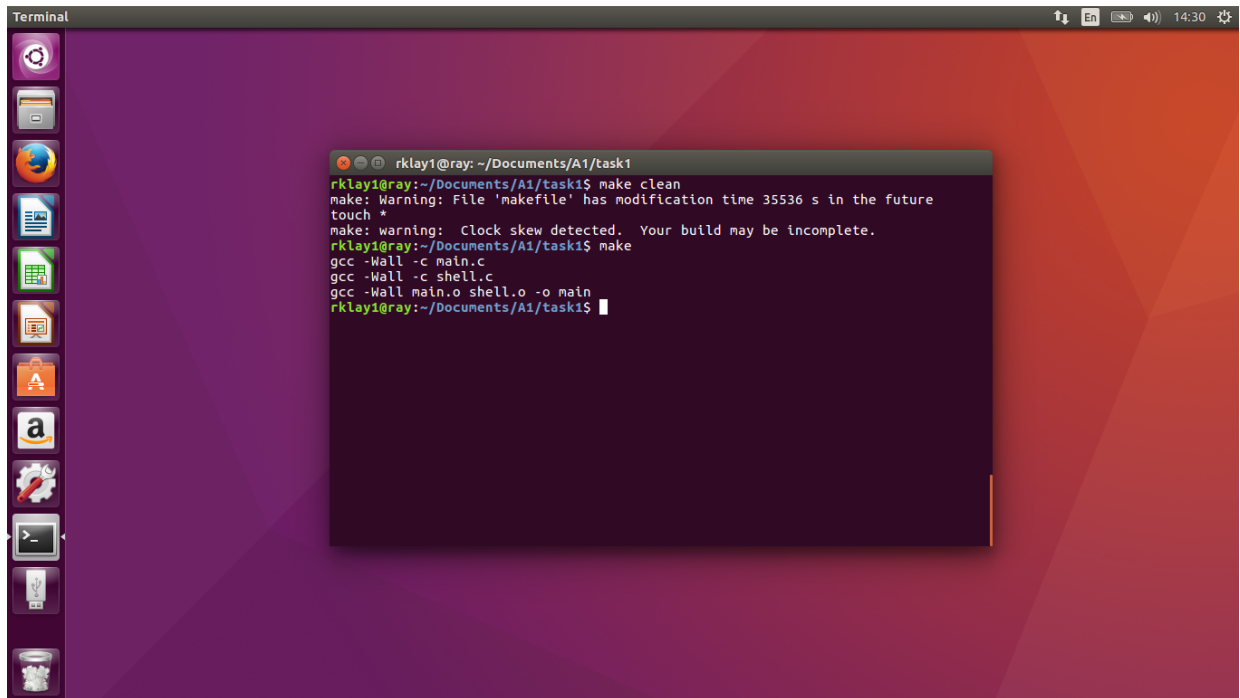
How to use the shell (CLI) program

1. Build the shell (CLI) program with "make" command

macOS Sierra (version 10.12.5)

```
[haha:task1 ray$ make clean
 touch *
[haha:task1 ray$ make
 gcc -Wall -c main.c
 gcc -Wall -c shell.c
 gcc -Wall main.o shell.o -o main
 haha:task1 ray$ 
```

Ubuntu 16.04 LTS

Description

To build the shell (CLI) program, the following commands are used:

- make clean
- make

Note: The "make clean" command will execute the "touch *" command which is used to fix the "make: warning: Clock skew detected. Your build may be incomplete." warning in Ubuntu 16.04 LTS.

2. Run the shell (CLI) program

```
[haha:task1 ray$ ./main
 $ ▌
```

Description

To run the shell (CLI) program, the following command is used:

- ./main

3. Execute commands supported by the shell

Supported commands:

- Clear
    - Clear the output screen.
- cd <directory>
    - Change the default directory to <directory>.
    - If <directory> does not exist, an appropriate error message will be displayed.
    - If the argument <directory> is not present, the current directory is changed to the directory given by the HOME environment variable in Unix.
    - **Additional functionalities:**
        - Support <directory> as path and name

```
[haha:task1 ray$ ./main
 $ ls
main   main.c   main.o   makefile   manual.txt   shell.c   shell.h   shell.o
 $ cd /Users/ray/Desktop/Monash/FIT2100/A1
 $ ls
task1   task2   user_documentation.docx
 $ cd task1
 $ ls
main   main.c   main.o   makefile   manual.txt   shell.c   shell.h   shell.o
 $ cd aaa
cd: aaa: No such directory
```

3

- ls <directory>
    - List the contents of the given directory named <directory>.
    - **Additional functionalities:**
        - If <directory> does not exist, an appropriate error message will be displayed.
        - If the argument <directory> is not present, ls will list the contents of the current directory
        - Support <directory> as path and name

```
[haha:task1 ray$ ./main
$ ls
main  main.c  main.o  makefile  manual.txt  shell.c  shell.h  shell.o
$ ls /Users/ray/Desktop/Monash/FIT2100/A1
task1  task2  user_documentation.docx
$ ls aaa
ls: aaa: No such directory
```

- echo <statement>
    - Display <statement> on the output screen with a newline character.
        - Statement is a sequence of characters that begins and ends with a double quote (e.g. "statement").
- help
    - Display the user manual using the command less in Unix.
    - **Additional functionalities:**
        - User manual is accessible from any directory

```
[haha:task1 ray$ ./main
$ ls
main  main.c  main.o  makefile  manual.txt  shell.c  shell.h  shell.o
$ help
$ cd
$ ls
Applications  CLionProjects  Desktop  Documents  Downloads  github  Ide
$ help
```

- quit
    - Terminate the CLI program and exit.

- create <file>
  - Create a new file with the name <file>.
  - **Additional functionalities:**
    - If there is an existing file with the same name <file> then the existing file will be overwritten by the newly created file.
    - Print the appropriate error message if there is no argument given by the user
- copy <old> <new>
  - Copy the contents of an existing file named <old> to a new file named <new>.
  - If <old> does not exist, an appropriate error message will be displayed.
  - **Additional functionalities:**
    - If there is an existing file with the same name <new> then the existing file will be overwritten by the newly created file.
    - Print the appropriate error message if the user does not provide the required arguments
- search <pattern> <file>
  - Display the number of occurrences of a sequence of characters <pattern> in the file named <file>.
  - If <file> does not exist, an appropriate error message will be displayed.
  - **Additional functionalities:**
    - Print the appropriate error message if the user does not provide the required arguments

- run <program>**[&]**
  - o Execute an executable program named <program> in **foreground**.
  - o **Additional functionalities:**
    - ▪ If the **'&' symbol** is included at the end of <program> (e.g. program&), the executable program named <program> will run in **background**.
    - ▪ Print the appropriate error message if the user does not provide the required arguments

```
[haha:task2 ray$ ./main
$ run input
Enter first name: aaa
Enter last name: bbb
Initial: ab
$ run infinite&
Background: "infinite" (pid->1895)
HELLO!
$ HELLO!
HELLO!

$ halt infinite
Killed: "infinite" (pid->1895)
```

- halt <program>
  - o Stop executing the program named <program>.
  - o **Additional functionalities:**
    - ▪ Print the appropriate error message if the user does not provide the required arguments