COMP3511 Operating System (Spring 2020)

Project 1 – A Simplified Linux Shell Program

Release on 6-Mar-2020 (Fri) Due on 27-Mar-2020 (Fri) 23:59

Introduction

The aim of this project is to help students understand **process management** and **inter-process communication** in Linux. Upon completion of the project, students should be able to implement a useful system program using related Linux system calls.

Overview

In this assignment, you need to implement a non-interactive command line interpreter that supports multi-level pipes. The non-interactive command line interpreter is named as `cmd`. Suppose there are 4 files in the current working directory:

```
cmd cmd.c in.txt out.txt
```

Here is a sample usage of the non-interactive command line interpreter.

```
$> ./cmd < in.txt > out.txt
```

`$>` represents the shell prompt.
The contents of `in.txt` and `out.txt` are as follows:

Content of `in.txt`	**Content in `out.txt`**

`ls`	`cmd`
	`cmd.c`
	`in.txt`
	`out.txt`

Restrictions

In this assignment, you **CANNOT** use `system` function defined in the C Standard library

```
// Should not use this function
int system (const char* command);
```

The purpose of the project assignment is to help students understand process management and inter-process communication. It is meaningless to directly use the system function to process the whole command. You should use related Linux system calls such as `pipe` and `dup2`. In addition, POSIX file operations such as `read, open, write, close` should be used, but not using `fread, fopen, fwrite, fclose` from C standard library. Please review the lab materials. The grader TA will check your submitted source codes.

Development Environment

CS Lab 2 is the development environment. Please use one of the following machines (`csl2wk`**`XX`**`.cse.ust.hk`), where **`XX`**=`01…50`. The grader TA will use the same platform to grade all submissions. Please note that different `gcc` compilers and Linux platforms may produce different results.

In other words, *"my program works on my own laptop/desktop computer, but not in one of the CS Lab 2 machines*" is an invalid appeal reason. Please test your program on our development environment (not on your own desktop/laptop) thoughtfully before your final submission, even you are running your own Linux OS. Remote login is supported on all CS Lab 2 machines, so you are not required to be physically present in CS Lab 2.

Starting Point

`cmd_given.c` is a starting point. To start your work, please rename the file as `cmd.c`

Read carefully the documentation in the base code. You are not required to start from scratch as the base file already provides you some useful features (e.g. command line parsing). Necessary programming concepts will also be introduced during the labs.

Please note that C programming language (instead of C++) must be used to complete this assignment. C is not the same as C++. Here is the command to compile and run `cmd.c`

```
$> ls
cmd.c in.txt

$> gcc -o cmd cmd.c

$> ./cmd < in.txt > out.txt
```

Multi-level Pipes

In C programming in Linux, a process creates a pipe by:

```
int ps[2];
pipe(ps);
```

After the pipe function call, `ps[0]` will be assigned to a file pointer to the read end of the pipe, and `ps[1]` will be assigned to the write end of the pipe. In a shell program, a pipe symbol (`|`) is used to connect the output of the first command to the input of the second command. For example,

```
$> ls | sort
```

The `ls` command lists the contents of the current directory. As the output of `ls` is already connected to `sort`, it won't print out the content to the screen. After the output of `ls` has been sorted by `sort`, the sorted list of files appears on the screen.

In this project, you are required to support multiple-level pipes. We assume that there exists at most `16` pipe segments. Please note that each command segment may have zero to many input parameters. For example, the following is a pipe command with 3 segments:

```
$> ls -l | sort | wc -l
```

Sample Test Cases

The grade TA will use the following pattern to grade your submission

```
$> ./cmd < [Input file name] > [output filename]
```

The input test cases (`inX.txt`, where `X=1-7`) are provided. You can assume that each input file stores a single line of command. You can assume that the input format is valid.

Input	**Expected output**
`ls`	`ls` displays the filenames of the current working directory
`ls  -l  -h`	`ls -l -h` displays the filenames of the current working directory using a better format. In this input, several empty space characters (i.e. "`\t\r\n\v\f`" and " ") are used as delimiters
`ls -lh`	It displays the same output as above
`ls | sort`	`ls` displays the filenames of the current working directory, and the filenames are sorted in an ascending order
`ls | sort | sort | sort | sort | sort | sort | sort | sort | sort | sort | sort | sort | sort | sort | sort`	Same as above. This input contains 16 pipe segments. This pattern is useful to test the upper bound of the number of pipe segment
`ls -l | sort | wc -l`	It displays the number of lines printed on the screen after running `ls -l`
`curl https://www.ceo.gov.hk/images/carrie02.jpg --output carrie.jpg | ls carrie.jpg`	The picture of the chief executive of HKSAR will be downloaded by `curl`. The filename "`carrie.jpg`" will be shown if the download is successful

Marking Scheme

1. (30%) Explanation of `process_cmd` within the comment block. You should use point form to clearly explain how you implement this function
2. (70%) The given test cases. Please note that the grader TA will check the source codes and may shuffle the order of test cases. Thus, students who hardcoding the results cannot get any mark.

Plagiarism: Both parties (i.e. students providing the codes and students copying the codes) will receive 0 marks. A plagiarism detection software will be used to identify the cheating cases.

Submission

File to submit: **`cmd.c`**

Please check carefully you submit the correct file. In the past semesters, some students submitted the executable file instead of the source file. Zero marks will be given as the grader TA cannot grade the executable file.

You are not required to submit other files such as the input test cases.

You only need to submit the file via CASS on /before the due day mentioned on the course web page.