Project TA - Peter CHUNG (cspeter@cse.ust.hk): Handling questions before the project deadline
Grader TA - Peter CHUNG (cspeter@cse.ust.hk): Collecting late submissions, grading, and handling appeals

COMP3511 Operating System Fall 2021

Project Assignment 1 (PA1) Simplified Linux Shell

File Redirection + 2-level Pipe Handling

23 Sep 2021 (Thursday) 15 Oct 2021 (Friday) 23:59

Introduction

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

Overview

You need to implement a command line program that supports a simple command with multiple input arguments, a file input/output redirection command, and a 2-level pipe command. For example, suppose there are 4 files in the working directory:

```
redir_and_pipe redir_and_pipe.c in.txt out.txt
```

The sample usage of the program is:

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

`$>` represents the shell prompt.
`./` means the current working directory.
< means input redirection, > means output redirection.
For example:

in.txt	**out.txt** (after running the command)
`ls`	`redir_and_pipe redir_and_pipe.c in.txt out.txt`

Restrictions

In this assignment, you **CANNOT** use `system` function defined in the C Standard library. 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 the related Linux system calls such as `pipe` and `dup/dup2`.

When connecting pipes, POSIX file operations such as `read, open, write, close` should be used, but not using `fread, fopen, fwrite, fclose` from C standard library. For details, please refer to the PA1 related lab.

Development Environment

CS Lab 2 is the development environment. Please use one of the following machines (`csl2wk`**`XX`**`.cse.ust.hk`), where **`XX`**`=01…40`. The grader will use the same platform to grade all submissions. Please note that different `gcc` compilers and operating system 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.

Remote login is supported on all CS Lab 2 machines, so you are not required to be physically present in CS Lab 2. For details about remote login to CS Lab 2 machines, please refer to the first few labs.

Starting Point

`redir_and_pipe_skeleton.c` is the starting point.
Please rename the file as `redir_and_pipe.c`

You can add helper functions, constants, and variables.

You don't need to include extra header files as necessary header files are included.

Read carefully the documentation. 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. C99 compiler option is added to allow flexible coding styles (e.g., variables can be declared anywhere within a function). Here are the commands to compile and run `redir_and_pipe.c`

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

$> gcc -std=c99 -o redir_and_pipe redir_and_pipe.c

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

File Input Redirection

Instead of typing the command on the console, the input can be redirected from a text file. The file input redirection feature can be completed by using the `dup/dup2` system call (they are discussed in the lab). The key idea is to close the default stdin (0) and replace the stdin with the file descriptor of an input file.

We can use the following command to count the number of lines of the file (`redir_and_pipe.c`). Here is a sample input file redirection usage:

```
$> wc -l < redir_and_pipe.c
```

File Output Redirection

Like the file input redirection, the output can also be redirected to a text file. The file output redirection feature can be completed by using the `dup/dup2` system call. The key idea is to close the stdout (1) and replace the stdout with the file descriptor of an output file.

We can use the following command to redirect the output of the ls command to an output text file (`out_demo_ls.txt`). Here is a sample output redirection usage:

```
$> ls -lh > out_demo_ls.txt
```

2-Level Pipe

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,

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.

2-level pipe can be extended to multiple-level pipe. However, in this assignment, you only need to implement a 2-level pipe.

Sample Test Cases

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

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

The input test cases are provided. You can assume that each input file stores a single line of command. Each file redirection command will have at most one character (<) and at most one character (>). Each pipe command will have exactly one (|) character. You can also 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 assignment, you only need to handle 3 types of empty space characters: spaces ( ), tabs (\t), and end line character (\n)
`ls -lh`	It displays the same output as above
`wc -l < redir_and_pipe.c`	It displays the total number of lines of the file (`redir_and_pipe.c`)
`ls -lh > out_demo_ls.txt`	Instead of displaying the result on the console, the output will be redirected to a text file (`out_demo_ls.txt`)
`wc -l < redir_and_pipe.c > out_demo_both.txt`	The total number of lines of the file (`redir_and_pipe.c`) should be redirected to a text file (`out_demo_both.txt`)
`wc -l > out_demo_both2.txt < redir_and_pipe.c`	The result is the same as above, except that the file output is redirection to another text file (`redir_and_pipe.c`)
`ls | sort -r`	It displays the filenames of the current working directory in a reverse alphabetical order. In this assignment, you only need to handle a 2-level pipe command.

Please note that the exact output will be different based on the files and configuration of your machine. In addition, you can assume that the output files do not exist in the machine

(Hint: you can use the command `rm -f out*.txt` to remove all output files)

Sample Linux Executable

The sample Linux executable file (which is runnable in a CS Lab 2 machine) is provided for your reference. After the executable file is downloaded to a CS Lab 2 machine, you need to add an execution permission bit to the file. For example:

```
$> chmod u+x redir_and_pipe
```

Please note that the executable file can only be executed in one of the CS Lab 2 machines. It cannot be run in other Linux / Mac / Windows machines.

Marking Scheme

1. (20%) Explanation of `process_cmd` within the comment block. Please write it near the top of the source code file to speed up the grading process. A template is provided near the top of the skeleton code.
2. (80%) The given 8 test cases. We do not have other hidden test cases.
3. **Important - DON'T** hardcode all possible test cases. Hardcoding can be easily identified by reviewing the source code, and the grader TA will check your source code. Students who hardcoding the output of the test cases (without implementing the codes) cannot get any mark even all test cases are passed.
4. **Important – DON'T** use any `system` function call in your implementation. Zero marks will be given if a `system` function call is used to handle the commands.

Plagiarism: Both parties (i.e., students providing the code and students copying the code) will receive 0 marks. A plagiarism detection software will be used to identify the cheating cases. We will handle the potential cheating cases near the end of the semester.

Submission

Submission platform: Canvas (https://canvas.ust.hk)
File to submit: **`redir_and_pipe.c`**

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

Make sure you submit the correct file. For example, in the past semesters, some students submitted the Linux executable file instead of the source file. Zero marks will be given as the grader TA cannot grade the executable file.

Late Submission

For late submission, please submit it via email to the grader TA.
There is a 10% deduction, and only 1 day late is allowed
(Reference: Chapter 1 of the lecture notes)