Project 1: A Simple Unix Systems Toolkit

Due: 2/25/2021

Project Statement: Develop a simple Unix systems toolkit for process and filesystem management

Project Objectives: Practicing Unix system calls, understanding Unix process management, synchronization, and inter-process communication (IPC), and filesystem management.

Project Description

In this project, you will develop a simple Unix systems toolkit for process and filesystem management, whose behavior is similar to both Unix shells (such as bash) and Unix systems forensic toolkits (such as Sleuth Kit). When your toolkit starts, it should print a dollar sign ($) and a space to the standard output terminal, and then wait for input from the user. You can assume each input line from the user will have no more than 80 characters. At the minimum, your toolkit should meet the following requirements. Name your toolkit as mytoolkit.

- Your toolkit should support a number of commands that you need to implement: "myexit", "mycd", "mypwd", "mytree", "mytime", "mymtimes", and "mytimeout".
  - myexit: terminate the toolkit
  - mycd: change the current working directory of the toolkit. Syntax: mycd dir. dir is the directory where the user wants to change to. If dir starts with a slash (/"), dir is an absolute path. Otherwise, dir is a path related to the current directory.
  - mypwd: print the absolute path of the current working directory
  - mytree: print the directories (and files) in a tree-like format. Syntax: mytree [dir], where dir is an optional command parameter. If dir is not given, use the current working directory as the beginning dir. The output of mytree should be similar to that of the Unix command tree. That is, you should align the directories and files based on their hierarchical positions in the filesystem. You do not need to print the horizontal and vertical lines as in the output of the Unix tree command; however, you need to align the printout properly.
  - mytime: run a command and time its running time. Syntax: mytime cmd [arguments], where cmd is the command to run, and the arguments are the optional command line arguments to the command cmd. The output of mytime should be similar to the Unix command time. At the minimum, mytime should report three values: 1) user CPU time; 2) system CPU time; and 3) elapsed wall-clock time for running the command cmd.
  - mymtimes: report the hourly number of files last modified in the last 24 hours. Syntax: mymtimes [dir], where dir is the optional beginning directory. If dir is not given, use the current working directory. For the beginning directory, this command will examine the last modification time of all the regular files (under the given directory and all the subdirectories, if any). It will group the files according to their last modification time into hourly time intervals in the last 24 hours, and report the number of files in each hourly time intervals in the last 24 hours. An example run of mymtimes is provided at the end of the project description.
  - mytimeout: The command mytimeout will run a command with a time limit. It is similar to the Unix command timeout. Syntax: mytimeout snds cmd [cmd-args]. The command mytimeout will start the command cmd specified on the command line and allow it to run up to snds seconds. Note that the optional cmd-args are the command arguments for cmd. After snds seconds, mytimeout will kill the process of cmd if it has not terminated, by sending a TERM signal. mytimeout then terminates itself (normal exit). You can assume that all the commands cmd given to mytimeout are existing commands in Unix, and you can call exec to run them inside mytimeout.
- You can implement these commands as either internal commands or external commands in the terminology of Unix shells. That is, they can be implemented either as a function in your mytoolkit program, or an independent, stand-alone program, and toolkit will fork a new child process to run that program. However, you need to consider the consequence of both choices to decide if you can implement a

command as an internal or external command. If some of them are implemented as external command, you also need to make sure that your toolkit can locate a command (program).
- Your toolkit should also terminate if end of file character (CTRL-D) is encountered.
- Your toolkit should support all existing external Unix commands in the system (i.e., executable programs that can be found in one of the search path contained in environment variable PATH). That is, a user of your toolkit should be able to run any Unix external commands in your toolkit program.
- Your toolkit should support pipes, e.g, command1 | command2 | command3 | command4. If it is needed, you can assume a maximum number of pipes in any command line. However, it should be greater than 1 (that is, you should support at least two pipes like this: cmd1 | cmd2 | cmd3). Note that commands may have parameters.
- Your toolkit should support I/O redirections, e.g. command1 < file1, or command1 > file1, or command1 < file1 > file2.
- You can assume that pipe and I/O redirection will not appear together on any command line of the toolkit.
- Your toolkit does not need to support any other special characters (in particular, special character expansion such as *).
- You cannot use the "system()" library routine (the function "system") in this project.

Notes on Project Requirements

- You cannot create any (long-term) zombie processes. For example, you cannot keep a terminated child process in the process table until your toolkit terminates. Points will be deducted if you have any (long-term) zombie processes running. That is, your toolkit must handle properly when a child process terminates.

- You should provide a proper makefile so that the TA can compile all your program(s).

- You should provide a README file to explain your choice of implementing the commands, in particular, they are internal or external commands. For external commands, please also explain any special requirements you may have in order to run your commands. In addition, you should also include any limitations or problems that your toolkit may have.

Grading Policy

A program with compiling errors will get 0 point. A program containing "system()" function will get 0 point. Total points: 100.

1. proper README file (5)
2. proper makefile file (5)
3. commands "myexit", "mycd", "mypwd", (10)
4. running single command (with and without parameters) on a command line (10)
5. running multiple piped commands ("|") (10)
6. allowing I/O redirection in a command ("<" and ">") (10)
7. command "mytree" (10)
8. command "mytime" (10)
9. command "mymtimes(10)
10. command "mytimeout" (10)
11. miscellaneous components/requirements of the project (e.g., proper handling of processes to avoid long-term zombies; terminating toolkit when end-of-file is encountered) (10)

Project Submission

Project must be submitted online on Canvas. Tar your readme file, makefile, all source code files into a single tar file. Make sure that you tar all the files successfully (you can check the content of the tar file by running "tar -tvf tar_file"). You are responsible for empty tar file or wrong tar file submitted, and late penalty will apply if you need to re-submit after the deadline.

Hints

Make sure your toolkit can terminate properly when end-of-file is encountered. We may test your toolkit by passing (i.e., redirect) to it a file containing a sequence of command lines.

An example run of our implementation of mymtimes:

```
duan@linprog2 (~...assignments/proj1) % mymtimes.x
~duan/courses/cop5570
Wed Jan 27 11:20:11 2021: 0
Wed Jan 27 12:20:11 2021: 0
Wed Jan 27 13:20:11 2021: 0
Wed Jan 27 14:20:11 2021: 0
Wed Jan 27 15:20:11 2021: 0
Wed Jan 27 16:20:11 2021: 0
Wed Jan 27 17:20:11 2021: 0
Wed Jan 27 18:20:11 2021: 0
Wed Jan 27 19:20:11 2021: 0
Wed Jan 27 20:20:11 2021: 0
Wed Jan 27 21:20:11 2021: 0
Wed Jan 27 22:20:11 2021: 0
Wed Jan 27 23:20:11 2021: 0
Thu Jan 28 00:20:11 2021: 0
Thu Jan 28 01:20:11 2021: 0
Thu Jan 28 02:20:11 2021: 0
Thu Jan 28 03:20:11 2021: 0
Thu Jan 28 04:20:11 2021: 0
Thu Jan 28 05:20:11 2021: 0
Thu Jan 28 06:20:11 2021: 0
Thu Jan 28 07:20:11 2021: 0
Thu Jan 28 08:20:11 2021: 0
Thu Jan 28 09:20:11 2021: 1
Thu Jan 28 10:20:11 2021: 3
```