CSS430  
Martin Metke  
2017/06/25

Program 1 Report

# Part 1

Part 1 involved creating a file that used system calls in C++ to replicate the functioning of a Linux shell running the command line “ps -A | grep <some search string> | wc -l".

## Algorithm

Initially I considered writing an extensible, while-loop-based program that could be passed any arbitrary string of the form “<command1>[ <arg1>] … [<argN>] [| <command2>[ <arg1>] … [ <argN>][ | …]]” with any number of commands and arguments connected by pipe (“|”) symbols – a generalized solution applied to this specific problem – but I decided in the end to take a naïve approach in the interest of reducing extra testing and the additional burden of writing long REGEXes. If I had taken this route, the main() function would have contained one while loop that stripped commands and pipe symbols from a pre-generated “command line” and decided when and how to set up pipes based on the current process’s position (or PID) and how many arguments remained:  
 Parent: no pipes, just wait()  
 Child1: Pipe for child, no pipe to parent  
 Child2: Pipe to child, pipe from parent  
 …  
 ChildN-1: Pipe to child, pipe from parent  
 ChildN: No pipe to child, pipe from parent.  
This could be accomplished with the c++ regex library and regex groups, but again, a large amount of extra work and testing would be involved to make sure that this generalized solution operated correctly.

In my actual implementation, the **processes** command is passed one and only one argument. This can either be a Linux-style help flag (“-h” or “--help") or a no-spaces search string. Any other number of arguments will result in a failure message and failure return value.

Internally the main() function does some basic argument validation but is essentially a chain of “if/else if/else” statements, as these allowed a fairly simple and logical division of the main function into great-grand-child, grand-child, child, and parent sections.   
 Parent: no pipes, just wait()  
 Child1: Pipe to child, no pipe to parent (closes stdin and dup2’s to FD[RD] pipe end)  
 Child2: Pipe to child, pipe to parent (closes stdin, stdout; uses FD[WR] and FDGC[RD])  
 Child3: No pipe to child, pipe to parent (closes stdout, dup2’s to FDGC[WR])

If the shell command to be emulated had included even one more pipe symbol, I would have gone with the above generalized solution instead, however.

## Results

Output comparison of cmd “processes <string>” vs “ps -A | grep <string> | wc -l" on UWB Linux Lab machines:  
  
sleet01@uw1-320-15:~/git/CSS430/program1$ ./processes tty

2

sleet01@uw1-320-15:~/git/CSS430/program1$ ps -A | grep tty | wc -l

2

sleet01@uw1-320-15:~/git/CSS430/program1$ ./processes Sys

0

sleet01@uw1-320-15:~/git/CSS430/program1$ ps -A | grep Sys | wc -l

0

sleet01@uw1-320-15:~/git/CSS430/program1$ ./processes user

0

sleet01@uw1-320-15:~/git/CSS430/program1$ ps -A | grep user | wc -l

0