Hilton Truong
1615505
Assignment 1 CMPUT 379 W22

Objectives

The objective of this assignment is to get familiar with Unix system calls using system times, process variables, process resource limits and process management functions will be used throughout the assignment. The knowledge from class will be applied in the creation of the program. With this assignment, a shell program is to be written running specific commands that will be described in the following section.

Design Overview

All implemented functions have basic error checking to see if it's a valid command or not. The program will also run up to a maximum of 32 tasks.

A time limit of 10 minutes is set for the CPU time and user and CPU times will be printed upon closure of the program.

1. cdir – this function takes a path as a parameter and the program will change the directory to the specified directory if it exists. If it does not exist, it will display an error that the directory change has failed. It can accept $HOME as a parameter.
2. pdir – this function does not take any parameters and just outputs the current directory that the user is in.
3. lstasks – this function does not take any parameters and outputs a list of all tasks that are currently running and not terminated.
4. run – this function takes 1 minimum parameter up to a maximum of 5. It is called by: run [program name] [arg1] [arg2] [arg3] [arg4]. It can take up to 4 extra arguments. A fork is created to run the specified program and may create multiple processes. The program can only take up to a maximum of 32 head tasks. If there is an error, it will output the error in the terminal.
5. stop – this function takes one parameter, the task number and sends a signal to stop the head process.
6. continue – this function takes one parameter, the task number and sends a signal to continue the head process.
7. terminate – this function takes one parameter, the task number and sends a signal to kill the head process. If the process is already terminated, it will output an error message.
8. check – this function takes one parameter, the target pid. It will run ps and check the task with the given pid and see if it is terminated or not. If it is, it will print the task information out. If it is not and it has children it will print the information out and the information of it's children.
9. exit – this function does not take any parameters and terminates all processes that are still running before exiting the program.
10. quit – this function just prints a warning message that it does not terminate the processes that are still running before it exits the program.

A makefile is provided to compile the program from the .cpp file and create a tar file containing the submission.

Project Status

The project is complete and can handle all the requirements specified in the assignment pdf. The makefile will compile and provide the executable for the function. All the above commands work, and most commands have some basic error checking, however common sense and basic logic will need to be used as the program does not handle advanced commands such as arrow up and ctrl commands.

The hardest part of the implementation was figuring out the check function. There was a lot of thought in how to get it to process the information that popen was giving and sorting through all the information. Then the logic and how the program would flow to print out the information after processing the information was a bit of a challenge.

Testing and Results

The testing of this program was done with the guidelines specified on eClass in the example output 1-3 pdfs. Each function was run and tested and made sure that they worked and provided the output that they were supposed to. The style of most of the outputs were matched to what was specified in the pdfs. All the functions work as they are intended to if they are not abused.

Acknowledgments

Code provided on the CMPUT-379 eClass – starter.cc was used for a lot of the imports and the general structure of the program

For iterating over a vector, inspiration was taken from:
https://stackoverflow.com/questions/12702561/iterate-through-a-c-vector-using-a-for-loop

For taking the inputs under main, inspiration was taken from:
https://stackoverflow.com/questions/236129/how-do-i-iterate-over-the-words-of-a-string#236803

The use of stringstreams was learned from: https://www.cplusplus.com/reference/sstream/stringstream/

The textbook Reference in Advanced Programming in the Unix Environment [APUE 3/E] was used to read up on the functions in the table such as chdir(), fork(), execlp() and etc.

The course notes from Professor Elmallah were also used to implement some of functions