Name: James Le
Student ID: 1639541
CCID: quanganh

CMPUT 379 ASSIGNMENT 1

Project Report

Objectives:

The objective of the assignment is to get used to the basic Unix system calls such as fork(), getenv(), etc., and concepts such as process, environment variables, resource limits, etc. In addition, it helps us to appreciate all the works behind the scenes of a shell.

Design Overview:

- cdir pathname:
  - The program changes its current working directory to the specified directory pathname. **Any directory alias must be started with "$"**. For example, suppose the alias $HOME is home/ubuntu/, then $HOME/folder1 is home/ubuntu/folder1, but HOME/folder1 only means the relative pathname HOME/folder1 from the current directory. For other directories that do not use an alias, we can type cdir pathname for any pathname we want.
  - Non-existence pathname will return an error.
- pdir:
  - Printing the current directory. Any argument after "pdir" will be ignored.
- lstasks:
  - This command prints all accepted tasks that the user has not explicitly terminated (even if it has been completed). If we want to delete the task from lstasks, we have to either terminate or exit.
  - Each of the entries in "lstasks" is stored in struct task_info, and we have an array of 32 of these structs.
- run pgm arg1 arg2 arg3 arg4:
  - Fork a process. Here, I use the **"double fork" technique** to avoid zombie processes, which means that the parent process will fork the child process, and the child process will fork the grandchild process, and we use execlp() in the grandchild process. In addition, the child process will be immediately terminated[1] after it forks a grandchild process for process init to pick the grandchild up and delete its data in the process table after the grandchild process finishes.
  - Since we are using the double fork technique, **when we use "check" on msh379, it will not show any tasks in "lstasks"** since the original parent of those processes in "lstasks" has been terminated, and their new parent is init. **Therefore, the tree of descendants rooted at process msh379 will not include those grandchild processes.**
  - If the current directory "." is not already in the search PATH, then we should add "./" before the program name. For example, "run ./myclock" instead of "run myclock". In the process of testing, if the run command does not work, then try to add "./" before the program name.

1. In fact, we delay about 0.1 seconds before terminating the child process since we want to know whether or not the grandchild can successfully open the file. If it is, we do not do anything, but if it is not, we broadcast a signal to its grandparent process to display the error and not add the task information to lstasks.

    - Non-existence file or not having permission to execute the file will return an error.
    - Only works with absolute or relative pathname with **no directory alias**.
    - **A maximum of 4 arguments can be specified**. Specifying more arguments does not return an error. Instead, **the program will ignore all arguments after the first 4**.
    - The program accepts at most 32 tasks. Terminating a task does not leave room for other tasks – in fact, 32 tasks here mean the total number of tasks that have been successfully launched using the "run" command. When we already have 32 tasks, the "run" command will not do anything. However, we can still use other commands.
- stop taskNo:
    - Send SIGSTOP signal using kill(taskNo.pid,SIGSTOP) where taskNo.pid is the process ID of that particular task.
    - Return error if the taskNo is not found. Any arguments after taskNo will be ignored.
- continue taskNo:
    - Send SIGCONT signal using kill(taskNo.pid,SIGCONT) where taskNo.pid is the process ID of that particular task.
    - Return error if the taskNo is not found. Any arguments after taskNo will be ignored.
- terminate taskNo:
    - Send SIGKILL signal using kill(taskNo.pid,SIGKILL) where taskNo.pid is the process ID of that particular task.
    - Return error if the taskNo is not found. Any arguments after taskNo will be ignored.
- check target_pid:
    - Read the process table from "ps -xao user,pid,ppid,state,start,cmd --sort=start" and print all the processes that is the descendant of the process target_pid (include target_pid itself).
    - If the process target_pid is defunct, then we do not print all of its descendants; instead, we print out that this process is defunct.
    - As mentioned above, since we are using the double fork technique in the "run" command, **when we use "check" on msh379, it will not show any tasks in "lstasks"** since the original parent of those processes in "lstasks" has been terminated, and their new parent is init. **Therefore, the tree of descendants rooted at process msh379 will not include those grandchild processes.**
    - I create a linked list structure to store the data of each row in "ps -xao user,pid,ppid,state,start,cmd --sort=start" so that I can construct the descendant trees easier in the "check" command.
    - I use a recursive function to print out all the elements in the descendant tree roots at process target_pid.
    - Any arguments after target_pid will be ignored.
- exit:
    - First, it terminates all process in "lstasks" that has not been finished yet. For example, tasks 0,2,4 are running, and tasks 1,3 have already been completed, then it terminates tasks 0,2,4. After that, it deletes all the entries in "lstasks".
    - Any arguments after taskNo will be ignored.

- quit:
  - End the main loop and print out the timing information. Note that this command does not terminate any tasks in "lstasks".
  - Any arguments after taskNo will be ignored.

Files Layout:

- There are 4 files to concern: The first one is msh379.c, the second one is linked_list.c, and the third one is Makefile, and the last one is this file - quanganh_CMPUT379_A1.pdf.
- msh379.c contains the main loop, which controls the main flow of the program.
- linked_list.c contains an implementation of a singly linked list with each node stores the data of a row in the process table "ps" command.
- Makefile defines a series of tasks to be executed – such as "make", "tar", and "clean".
- quanganh_CMPUT379_A1.pdf is this file – project report.

Project Status:

All features have been implemented.

Testing and Results:

I have tried to do similar steps as the three example outputs posted on eClass, and it all works. Although, one small detail: I have not added the current directory "." to search PATH, so I have to add "./" before the program name in "run" command in some user-defined scripts. For xClock and xEyes, I do not have to do that.

Acknowledgments:

I use the code in "Advanced Programming in the UNIX Environment" by W. Richard Stevens and Stephen A. Rago and read Linux documentation on Internet from the "Linux man-pages project" at The Linux man-pages project (kernel.org).

The function pr_times() used for printing the time format is taken from "Advanced Programming in the UNIX Environment" page 281, 282.