CMPUT379 – Assign 1
Project Report

Brady Pomerleau

Objectives:

To gain understanding of processes and process management techniques in operating systems generally, and to gain experience creating, monitoring, controlling, and terminating processes in a Unix environment specifically.

Design Overview:

- a1jobs
    - provide interface for creating, monitoring, and controlling jobs using set of commands:
        - list: list all of the processes created by a1jobs
        - run command [args]: fork, and exec command [args] (up to 4 arguments)
        - suspend index: suspend process corresponding to index value
        - resume index: resume process corresponding to index value
        - terminate index: terminate process corresponding to index value
        - exit: terminate all children of a1jobs and exit program
        - quit: quit program without terminating child processes
    - limit on total number of created processes for each execution of a1jobs is 32
    - at close of program, print the time since a1jobs started, CPU and system times of a1jobs, and CPU and system times of a1jobs+children execution times
        - since a1jobs does not clean up zombies, children times will always be 0
- a1mon
    - monitor a process and its descendants; upon termination of monitored process, descendent processes are terminated
    - program initiated with “a1mon pid interval”
    - pid is process id of process to be monitored
    - interval is the refresh interval for checking status of process tree
    - a1mon receives information about processes by opening a pipe to a ps process

    - once a1mon determines target process is terminated (either by its state=zombie or lack of existence in ps output), a1mon terminates all descendent processes of pid
    - all descendants are tracked, even if they are not direct descendants and are orphaned before the termination of the target process. They will be cleaned up at the end of the routine.

Project Status:

All functionality is implemented as described.

Notes:

a1jobs doesn't clean up zombies.

For a bash script to be run successfully, the path name must be included or . must be included in the PATH.

In a1mon, if orphans of a descendant of the target process are generated, a1mon does not clean terminate them.

TREE struct has an automatically expanding buffer so it will grow arbitrarily large. However this feature is not working, and I don't have time to fix it, so the MAX_TREE_BUF_SIZE macro must be set sufficiently high for the purposes of the program. This is problematic because the code keeps track of pids even afer the process has ended, so the array grows with each interval for some processes. If I had more time, I would make two sets, the list of processes discovered at iteration, and the old list of monitored processes, and then I would take the "AND" of the two sets. This would remove processes from the tree that no longer exist, and would keep the tree from growing ad infinitum.

Testing and Results:

The following tests were conducted on university lab computers:

Test	Description	Result
a1jobs: run command	Run a system call, confirm success with ps	success
a1jobs: run command	Run a bash script, confirm success with ps	Success (if /. added to PATH, or run ./file executed)
a1jobs: job limit	Run more than 32 jobs	success
a1jobs: list command	Basic test: run some jobs and see list output	success
a1jobs: list command	Confirm "list" doesn't list explicitly terminated jobs	success
a1jobs: suspend command	Suspend a job, verify status with ps	success
a1jobs: resume command	Resume a suspended job, verify status with ps	success
a1jobs: terminate	Terminate a job, verify status with ps	success

command		
a1jobs: exit command	Verify jobs are terminated with ps	success
a1jobs: quit command	Basic test: run command, program should terminate	success
a1mon: descendants all accounted for	Generate a descendant tree using a1jobs (with exec functionality disabled, can create multi-generational trees)	success
a1mon: kill target pid	Confirm all children are killed with ps	success
a1mon: kill child with children	Kill a child that has children. Confirm that orphaned children are still tracked	success

Acknowledgements:

To complete this assignment I used the following resources:

- Course textbooks:
  - Operating Systems Concepts ver.9
  - Advanced Programming in the Unix Environment ver.3
- various C programming errors troubleshooting accomplished using Google search and forums (primarily stackoverflow.com)