Ayub Ahmed
1530472

OVERVIEW:

The objective of this project is to gain insight on handling processes in a linux system including how to fork processes, execute programs within forked processes and monitoring said processes for status changes.

DESIGN OVERVIEW:

a1jobs: Tracks a list of jobs (programs executed in child processes) with 32 being the maximum admitted number of jobs. It performs a range of commands with regards to the jobs including printing a list of jobs and sending signals to the processes.

- Send_sig:
  - Sends signals to the user-specified job number to either terminate, resume, or suspend.
  - If the job number specified from the user is invalid (greater than 31) or the argument for the job number is not all digit characters, it returns to main without doing anything.
  - Returns void
- Limit_cpu:
  - Sets a limit on the cpu resources the program can use
- Run_child_prog:
  - Forks a child process
  - Executes a program in the child process using the execlp() function with the arguments specified in the user input
  - If the program does not exist or execlp returns with some error it kills the child process and returns to main.
  - Otherwise, it adds the job to the array of structs containing the admitted jobs
    - Adds child pid and command from the user input to struct object
  - Returns 1 if successful, otherwise returns 0
- Print_list:
  - Iterates through the admitted jobs contained within the jobs struct array and prints those which have not been terminated by the user
- suspend_job:
  - Sends the signal SIGSTOP to the job specified by the user
- Resume_job
  - Sends the signal SIGCONT to the job specified by the user
- Terminate_job
  - Sends the signal SIGKILL to the job specified by the user
  - Sets the command in the struct array of jobs to NULL pointer
- Exit_job
  - Iterates through struct jobs array current_jobs and exits any job which has not been exited already
- Remove_newline:
  - Removes the new line from the end of the user input and returns it
- Str_to_i
  - Takes the string representation of a number and converts it to an integer

- Is_num
    - Checks if a string contains only digit characters
    - Returns null pointer if it contains non-digit character, otherwise return the string
- Free_var
    - Free dynamically allocated variables created during execution
- Main:
    - Iterates through a while loop getting user input
    - Calls the function that executes the action specified by the user
    - Prints elapsed times before termination of a1jobs program (real, sys, user, child user, child system)

a1mon: Keeps track of a tree of processes rooted at the target process and preforms a cleanup of descendant processes if the target process has terminated.

- Str_to_i
    - Takes the string representation of a number and converts it to an integer
- Is_num
    - Checks if a string contains only digit characters
- Limit_cpu:
    - Sets a limit on the cpu resources the program can use
- Free_var
    - Free dynamically allocated variables created during execution
- Terminate_tree:
    - Iterates through the struct array containing all descendant processes and terminates them
- Main:
    - Opens a pipe to ps
    - Prints header and lines from the ps file
    - Tokenizes input
    - Goes through the ps file line by line adding any descendant process to the array of monitored processes
    - Checks if the target process is still active
        - If it is then program sleeps for the user specified interval (if specified) iterates a counter then loops back
        - Otherwise calls terminate_tree to terminate descendant processes

<u>PROJECT STATUS</u>

The status of the project is complete. One of the difficult parts to implement was the feature that checks whether the child process has executed the program successfully or not. The most difficult part was most likely tokenizing strings and dealing with strings in c. At first I tried to write my own function to parse the strings but discovered the function strtok() which made the process somewhat easier. The last part that I had found difficult was keeping track of the dynamically allocated struct arrays and reallocating more size when they became full.

<u>TESTING AND RESULTS</u>

a1jobs:

- Entered inputs not containing a recognized command
  - Program prints an error statement and loops back to allow the user to enter another input
- Entering non-numbers for job number argument
  - Prints error and loops back to take another user input
- Print statements to print tokens to make sure tokenizing worked correctly
  - Many small errors with splitting the string that were fixed by adding print statements
  - String was split into correct tokens and stored correctly into an array
- Running program names that do not exist
  - At first it added the job to the struct, however through print statements I was able to discover it was because the parent executed and checked the status of the child before the child could exit with an error status
  - Fixed by sleeping the parent program for 1 second
- Many syntax errors that were fixed using the error outputs of compilation
- Ran myclock supplied to us, xeyes, and xclock both of which are built in linux programs and they were executed successfully

a1mon:

- Tested the execution if the user entered unrecognized input
  - Prints an error and exits the a1mon program
- Used print statements to see the tokens of the line extracted from the ps pipe
  - Small errors with the delimiter specified and with storing the tokens
  - Fixed using the print statements
- Ran alongside a1jobs which was running xeyes, xclock, and myclock and checked the output
  - Was adding duplicates of every descendant process to the array of monitored processes
    - Implemented a check to see whether the process is already in the monitored processes and only add it if it isn't already being monitored
- Segmentation fault with reallocating space to the struct array of monitored proc
  - Forgot to change the max size variable so the amount of memory wasn't being increased just added a line to increase the variable before using it
- Many syntax errors that were fixed using the error outputs of compilation

ACKNOWLEDGEMENTS

- www.ibm.com
- Advanced programming in the Unix Environment 3rd edition
- man7.org/linux/man-pages
- Lecture slides