CMPUT 379
OPERATING SYSTEM CONCEPTS

Assignment #1
Process Management Programs

Due: Thursday, October 4, 2018 09:00 PM

By Hee Soo Park (1389532)

Objectives

a1jobs : To run executable processes as child processes of a1jobs, and able to list all process that have not been explicitly terminated by the user, suspend a running process, resume a suspended process, and terminate a listed processes. Within a user set limit of CPU time, and records user and CPU time for the current process.

a1mon : To monitor process that has a user given pid (eg., a1jobs) running on the same workstation. The a1mon program monitors the target_pid with interval of time that user inputs, while monitoring, a1mon displays numerous information about every child process of the target_pid, and updates and displays simultaneously when some action is done from the target_pid.
And when target_pid is terminated, a1mon cleans up the leftover running process that target_pid process have left behind.

Acknowledgements

char *trim(char *str); function code have been referenced and used from "https://stackoverflow.com/questions/122616/how-do-i-trim-leading-trailing-whitespace-in-a-standard-way"

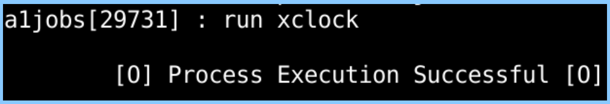
I have referenced lecture given by Prof. Ehab Elmallah, his office hours.
Lab hours with TA's, lecture slides

Referecd fork()/pipe from
"https://www.geeksforgeeks.org/c-program-demonstrate-fork-and-pipe/"

Design Overview

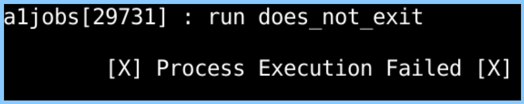
a1jobs

- Success run of a1jobs



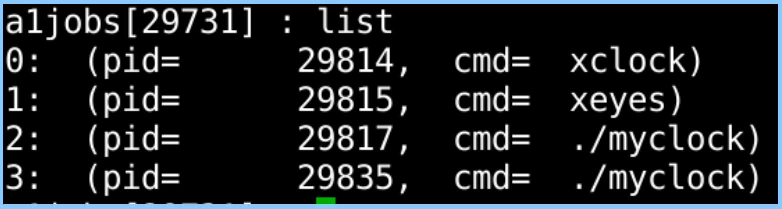
```
a1jobs[29731] : run xclock

        [0] Process Execution Successful [0]
```

- Failed run of a1jobs



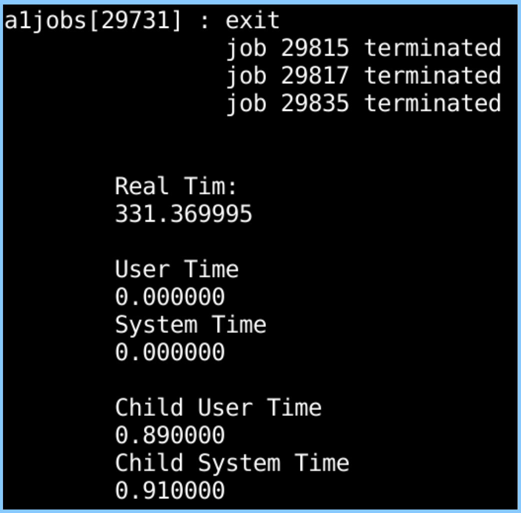
```
a1jobs[29731] : run does_not_exit

        [X] Process Execution Failed [X]
```

- List of all admitted jobs that have not been explicitly terminated by the user.



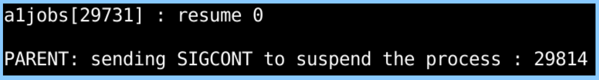
```
a1jobs[29731] : list
0:  (pid=     29814,  cmd=  xclock)
1:  (pid=     29815,  cmd=  xeyes)
2:  (pid=     29817,  cmd=  ./myclock)
3:  (pid=     29835,  cmd=  ./myclock)
```

- Exit of a1jobs, terminating process that have not been explicitly terminated by the user

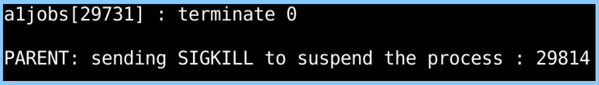


```
a1jobs[29731] : exit
                job 29815 terminated
                job 29817 terminated
                job 29835 terminated


        Real Tim:
        331.369995

        User Time
        0.000000
        System Time
        0.000000

        Child User Time
        0.890000
        Child System Time
        0.910000
```

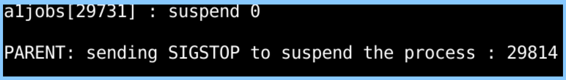
- Suspend, Resume, Terminate return in a1jobs



```
a1jobs[29731] : resume 0

PARENT: sending SIGCONT to suspend the process : 29814
```



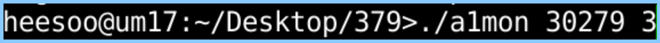
```
a1jobs[29731] : terminate 0

PARENT: sending SIGKILL to suspend the process : 29814
```



```
a1jobs[29731] : suspend 0

PARENT: sending SIGSTOP to suspend the process : 29814
```

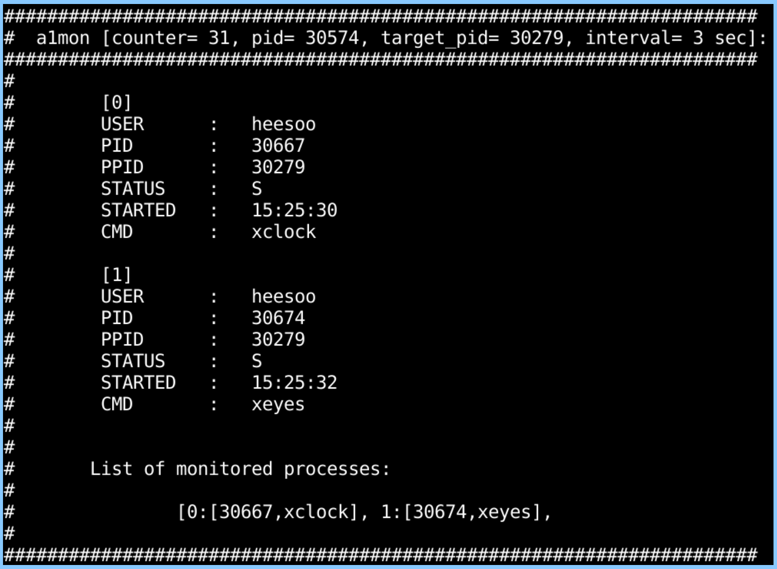
a1mon

- Running a1mon from terminal with correct number of arguments



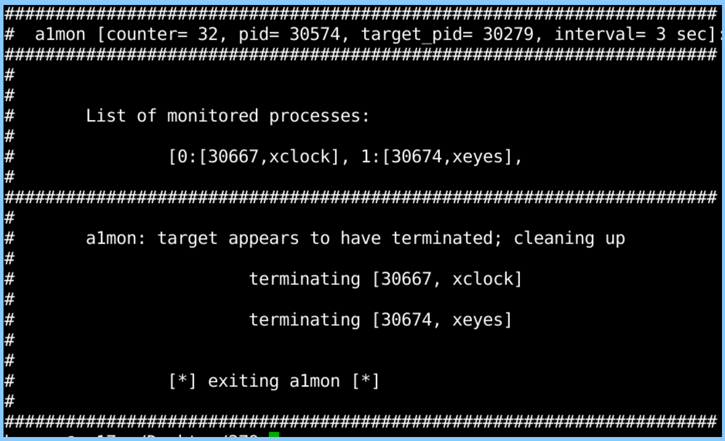
```
heesoo@um17:~/Desktop/379>./a1mon 30279 3
```

- Iteration of a1mon, monitoring all the process that have been successfully executed by target_id and displaying its info.



```
######################################################################
#  a1mon [counter= 31, pid= 30574, target_pid= 30279, interval= 3 sec]:
######################################################################
#
#        [0]
#        USER      :   heesoo
#        PID       :   30667
#        PPID      :   30279
#        STATUS    :   S
#        STARTED   :   15:25:30
#        CMD       :   xclock
#
#        [1]
#        USER      :   heesoo
#        PID       :   30674
#        PPID      :   30279
#        STATUS    :   S
#        STARTED   :   15:25:32
#        CMD       :   xeyes
#
#
#       List of monitored processes:
#
#               [0:[30667,xclock], 1:[30674,xeyes],
#
######################################################################
```

- When target_id is terminated, a1mon also terminates after cleaning all the left over process from target_id



```
##################################################################
#  a1mon [counter= 32, pid= 30574, target_pid= 30279, interval= 3 sec]:
##################################################################
#
#
#       List of monitored processes:
#
#               [0:[30667,xclock], 1:[30674,xeyes],
#
##################################################################
#
#       a1mon: target appears to have terminated; cleaning up
#
#                       terminating [30667, xclock]
#
#                       terminating [30674, xeyes]
#
#
#               [*] exiting a1mon [*]
#
##################################################################
```

Project Status

a1jobs have been finished with all the required functions, they are stable, consistent and correct. I have implemented all the required function as modular as possible. Run, list, suspend, resume, terminate, exit and quit work flawlessly. But please note that, user needs to wait until next input line is printed in terminal before typing next instruction. As sleep() have been implemented to make program work concurrently

a1mon have been finished with all the required functions, they are stable consistent and correct. I have also implemented all the required functions as modular as possible. Iteration counter, display terminal GUI, and clean-up on target_pid terminate all work correctly. Including child-of-child is also added to monitored processes.

I am satisfied with both my programs and their outputs and GUI.

I had hard time figuring out the return value of execlp to decide if running the process have been successful or not. Also execlp returned from child and grabbing it in parent was also a challenge, but use of pipe and O_NONBLOCK was very helpful to resolve this issue. GUI is extra but I felt displaying irrelevant process was hard to search and find the child pid of target_id. So my a1mon only contains the relevant processes.

Testing and Results

I have tested a1jobs by

running function by, running a process and check if they run successfully or fail to run.
List function have been tested by checking if successfully ran process have been append to list and display properly.
Suspend have been tested by suspending a running process and check if the status of the process have been changed from S+ to T and tested resume function by checking the status of the process from T to S+, from ps -aux.
Terminate function have been tested by running a process successfully and terminating it, and check if the process have been removed from the list and also check if the process still exists in ps -aux.
Exit function have been tested by exiting a1jobs, and check in ps -aux if the process that have not be explicitly terminated by user have been removed and been terminated by exit function.

I have tested a1mon by
making sure if a1mon refreshed and updates by user input interval,
if any action is taken in target_pid (eg., run, suspend, resume, terminate …) a1mon updates the updated data and displays correctly,
if target_pid is terminated check if a1jobs clean-up the process that have not been explicitly terminated by user in target_pid, by checking if those process have been removed in ps-aux