Lab 6

Programming to C
CSCI 112, Fall 2020

Objectives

- Practice with using execl, wait, and fork

Description:

You are going to read in all the processes that are currently running (using ps aux). Then you are going to find the last user to do a bash process (is -bash). Then you will do a finger command on that net id. Print out the user's name and the last time they logged in.

Requirements:

- Best practice: create a directory called lab6 to work in
- You must do this all in one program – you can't write a script to do some of this for you.
- Write the output to the screen
- DO NOT USE GLOBALS.
- MUST COMPILE WITH -Wall
- You must submit:
  - 1) Screen shot showing your successful compile and successful run with output
  - 2) source code
- You must use fork, execl, and wait system function calls.
- When reading each line for ps aux, the line can be very long. So malloc memory for line to be 2000 characters. Because of this, you must free the memory after you process each line.
- Find the last netid that is doing a -bash process and is not your netid and is not root.
- Each time you open a file for reading (the ones where you redirected output from ps aux and finger) you must check to see that it will open successfully. Exit if it doesn't. (There is a chance that the parent process will try to open the file for reading before execl is done writing to the file.)

My Output

```
name is ethan.house
On since Wed Nov  4 13:49 (MST) on pts/31 from 153.90.90.156
```

Submission

- Due Date: Monday, 11/16 at 11pm

  Each student will complete and submit this assignment individually. I will check for plagerism. Labs submitted after the due date/time will not be accepted.

Grading

Points (100 pts)

- 5 points – comments explaining what your program does
- 10 points – indent your code so it is readable
- 15 points – submitted screenshot as required above
- 15 points – compiles successfully with -Wall – no warnings
- 15 points – submits the output files containing your ps aux output and your finger output
- 10 points – submitted source code
- 7 points – uses fork
- 7 points – uses execl
- 6 points – uses wait
- 5 points – prints the correct output
- 5 points – free each line you read after you process it