Spring 2017 **EECS 338 Assignment #1** Due: Feb. 8, 2017 (Wednesday)
G. Ozsoyoglu (*100 Points*)

This assignment is just a practice assignment for you to

- Make system calls involving Linux process control functions, and observe their effects (e.g., the *execx()* system call),
- Become aware of and use environment variables, and
- Create multiple processes, and simulate (in an artificial and hopefully fun manner) concurrent execution of these processes.

So, for those of you who are familiar with Linux, don't read too much into this assignment, and don't ask why we are doing this and that.

So, here is what you will be doing: your main program, called the (parent) process *P*, will print its *pid* (*getpid()* call), hostname (*gethostname()* call), username (*cuserid()* call), the time of the day (*ctime()* or *time()* calls), its working directory (*getcwd()* call) and some text indicating that it is the parent process. Then *P* will create an environment variable HIPPO (why not just a variable? Don't ask!), and initialize it to 11 (*putenv()* call). *P* will then fork out two child processes, processes C_1 and C_2. Each of the two child processes will then print its pid and its parent pid (*getppid()* call) together with a meaningful text explanation. After these, the three processes will take turns to count backwards and decrement HIPPO, and print as follows:

```
P:    10 little hippopotamus (HIPPO value is now  xx)
C1:    9 little hippopotamus (HIPPO value is now  xx)
C2:    8 little hippopotamus (HIPPO value is now  xx)
P:     7 little hippopotamus  (…)
C1:    6 little hippopotamus
C2:    5 little hippopotamus
P:     4 little hippopotamus
C1:    3 little hippopotamus
C2:    2 little hippopotamus
P:     1 little hippopotamus
```

The three processes achieve this decrementing of HIPPO and printing its value by repetitively sleeping (*sleep()* call), awakening, then printing and flushing the printed line (*fflush()* call). Then, the parent process P waits until both C_1 and C_2 terminate, decrements and prints the final value of HIPPO, and exits. The process C_1 changes its directory (*chdir()* call) to some other directory, changes its address space (*execXX()* call) to execute the "*ls*" command, and exits. The process C_2 prints the current working directory (*getcwd()* call), gets the variable HIPPO (*getenv()* call), decrements and prints it, and exits.
And, finally, make sure that each process takes the following additional actions.

1. Find out the execution time, and parent and child processes' real and effective user ids, and their group ids.
   Question 1: Explain briefly the purposes of these ids.
2. For all three processes, at the end of their execution, find and print the current time (*ctime()* or *time()* calls). Have your processes loop for a while in order to accumulate some nonzero time values. Otherwise, you may observe zero time values.
   Question 2: why?

Documentation for all the calls you will be using can be found in Chapters 2 and 3 of the Linux textbook, and/or in the *man* pages.

Remember to error-check all system calls: check return values for success, and use *perror()* when possible on failure. See the man pages for a description of possible errors for each system call.

Once again, you are to use a *Makefile* to compile your assignment, and include the *Makefile* with your submission. Note that your program will be run and tested on the *eecslab-X* virtual machine; so make sure that it runs there. Also include a file containing the output of your program, which you can obtain via redirecting standard out as in: *"./yourprogram &> output.txt"*. On the due date, submit your code, *Makefile*, and program output to Blackboard as a single zip file named "*as2.zip*".