Fall 2015 **EECS 338 Assignment #1** Due: September 17, 2015

G. Ozsoyoglu (*100 Points*)

Binomial coefficient computations via Linux processes

The purpose of this assignment is to create concurrently executing Linux processes, and have them execute simple Linux system calls. The Unix system calls you will be required to use are *fork(), exec(), getpid(), getppid(), exit(), wait(), sleep(),* etc.

Your parent process will create four child processes by using the *fork()*system call. Execution of the child processes will be synchronized by using the *sleep()* system call as follows.

a. The first child process, child 1, prints "**(n (n-2)) binomial coefficient computations of integers n=2, 3, 10, start now!**", and terminates.

b. The second and the third child processes, i.e., child 2 and child 3 processes, alternate in computing $\binom{n}{n-2}$ ***binomial coefficient*** for n=2, 3, 4, 5, 6, 7, 8, 9, and 10. So, child 2 computes and prints the binomial coefficient $\binom{\mathbf{2}}{\mathbf{0}}$; then child 3 computes and prints the binomial coefficient $\binom{\mathbf{3}}{\mathbf{1}}$; then child 2 computes and prints $\binom{\mathbf{4}}{\mathbf{2}}$, etc., until child 2 computes and prints $\binom{\mathbf{10}}{\mathbf{8}}$.

The formula for the binomial coefficient $\binom{n}{k}$ ("n choose k") where n and k are nonnegative integers and n≥k is:

$$\binom{n}{k} \equiv \frac{n!}{k!\,(n-k)!}$$

$$\text{and} \quad n! \equiv \begin{cases} 1 & if\ n = 0 \\ n * (n-1)! & if\ n > 1 \end{cases}$$

Thus, binomial coefficients of n=0, n=1, n=2, and n=3 are <1>, <1,1>, <1, 2, 1> , <1, 3, 3, 1>, respectively.

The recursive formula for binomial coefficients is:

$$\binom{n}{k} + \binom{n}{k+1} = \binom{n+1}{k+1}$$

Note that child 2 and child 3 processes stop their binomial coefficient calculations once the computation and printing of the binomial coefficient $\binom{\mathbf{10}}{\mathbf{8}}$ is reached, and terminate.

c. The fourth child process, namely, child 4, sleeps long enough for processes child 1, 2, and 3 to terminate, and then executes "*ls -l*"; prints the results to standard out (stdout), and terminates.

d. The parent process waits for the termination of each child process, and then prints their termination status.

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 produce meaningful print statements like "*The process id xxx produced the binomial coefficient ….. for integer n …* " , etc., instead of just numbers. And, finally, make sure that each process takes the following additional actions.

1. Find out the execution time, the username (*cuserid()*), and parent and child processes' real and effective user ids, and their group ids.
   Question 1: Explain briefly the purposes of these ids.

2. Force the two alternating child 2 and child 3 processes to sleep long enough for the user to visually observe their printouts sequentially.

3. For both the parent process and the child processes, at the end of their execution, find and print the current time (*ctime()* and *time()* calls), user CPU time, and system CPU time that they use. (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?)

4. Force the parent process to wait for the termination of child processes and print the termination status of each child process.

Don't forget to make sure that the output is in the correct order (using *sleep()* calls to ensure the order). Use the Linux shell command "*script*" to record the entire session. Please print meaningful statements ("*The hostname is ...*") instead of just numbers.

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.

You must use a Makefile to compile your assignment (see the recitation code page for numerous Makefile examples), and include the Makefile with your submission. The program will be tested on the *eecslinab3* virtual machine; so make sure it compiles there. Note: an incorrect program that compiles typically receives more points than any program that doesn't. 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 Assignment 1). In addition, submit a separate document containing your answers to questions 1 and 2.