University of Massachusetts Dartmouth
CIS 370, Fall 2013
10/08/2013, 10/10/2013
BONUS Lab: Process Family Trees
Due: 10/22/2013 (Tuesday), 10/24/2013 (Thursday)

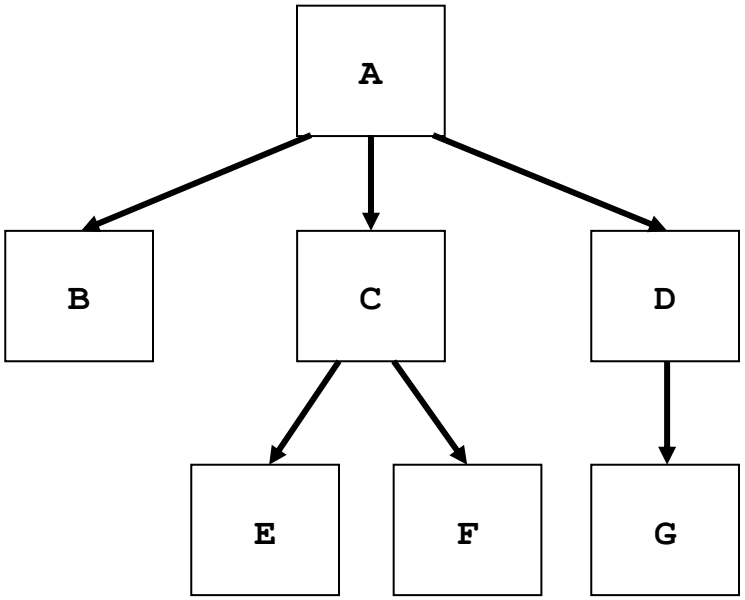
Objective

In this lab you will once again be familiarizing yourself with the intricacies of the `fork` system call and its ability to create child processes differently depending on how carefully it is used. You will have two weeks to complete this assignment.

Description

This lab is optional and will not negatively affect your grade, but it is recommended that you try it out (bonus points worth an entire lab grade are achievable)!

This lab will help you understand the many different effects caused by forking processes in strategic ways. Consider the family tree of processes shown below (where **A** is the initial parent process):



- This particular family tree can be generated using exactly **SIX** `fork()` calls.
- This family tree can *also* be generated with exactly **FIVE** `fork()` calls.
- This family tree can *also* be generated with exactly **FOUR** `fork()` calls.
- This family tree can *also* be generated with exactly **THREE** `fork()` calls.

Your assignment is to explain, through pictures and/or detailed step-by-step instructions, how each of the above forking scenarios can generate this same tree. You should explain which processes should fork a child and in what order, to create the tree in the specified

number of steps. You can submit your responses as .doc/.docx/.pdf/.txt/.rtf, etc. DO NOT simply submit code!!!

You should provide your own written responses, but may also incorporate tables similar to that provided below to organize your thoughts. If using the table, you should put in each cell, the PID (corresponding letter) of the process created by the particular parent at that fork call. If a particular process does not create a child, place an X in the appropriate cell. For example, let's assume that only *Process A* creates *Process D* during the first fork, and then only *Process D* creates *Process G* during the second fork. The first two columns of the table should look like the example shown below.

Process	Fork 1	Fork 2	Fork 3	Fork 4	Fork 5	Fork 6
A	D	X				
B	X	X				
C	X	X				
D	X	G				
E	X	X				
F	X	X				
G	X	X				

WARNING: Calling `fork()` inside of a loop counts as multiple fork calls, even though it only appears once in the code. Solutions using forks within loops may be correct, but you need to be very careful in your approach. No partial credit will be given for *any* solution. Your explanations/tables/diagrams must be completely correct for a forking scenario in order to get the points for that scenario. (There may be more than one correct solution for some scenarios.)

MORE BONUS POINTS

If you choose to code any of the four scenarios you've described in your written responses, you will earn some extra credit towards your lab grade (1.25 points per solution – All 4 correct solutions will get you 5 points). Forking must be handled *perfectly* and must correctly generate the provided family tree or you will receive no credit for a particular solution (no partial credit will be awarded for an individual forking scenario). Remember to be cautious if using loops and forks together.

IMPORTANT: Coding the solutions without providing the written responses will NOT earn you any extra credit!