Spring 2017, G. Ozsoyoglu **EECS 338 Assignment #3:** *Pipes* *Due: Feb. 18, 2017 (Sat)*
(100 points)

In this assignment you will develop a program that uses Linux pipes to play the *Prisoner's Dilemma game* (http://en.wikipedia.org/wiki/Prisoner's_dilemma). The *Prisoner's dilemma* is as follows: two suspects, you and another person, are arrested by the police. The police have insufficient evidence for a conviction, and having separated the both of you, visit each of you and offer the same deal: if you confess and your accomplice remains silent, he gets the full 10-year sentence and you go free. If he confesses (i.e., he *defects*) and you remain silent (i.e., you *cooperate*—with the other prisoner, not with the police), you get the full 10-year sentence and he goes free. If you both stay silent, all they can do is give you both 6 months for a minor charge. If you both confess, you each get 6 years. *The Prisoner's Dilemma* has been extensively studied in *Game Theory* as well as being the subject of a computer competition (see http://www.iam.ecs.soton.ac.uk/news/612).

You will write a parent program that will fork two prisoner children processes. Before forking, the parent will setup **four unnamed pipes**, two for communication with each child. The parent will then act as referee for the prisoner's dilemma game. You will create a protocol for communication between the processes. The children should first send their process ids to the parent. The parent will then coordinate the prisoner's dilemma game. The parent should conduct at least ten iterations of the game with the original children, keeping track of the score, where score is computed by adding up the sentences received in each game according to the description above. The parent will then signal the prisoners to exit and output the final score. (Note: The protocol you create can be extremely simple, i.e., 0's & 1's or single characters). The children can use any strategy for the game, though *random choice* may be the simplest option.

The output of the referee should be like the following (assume 1147 and 1148 are the process id's of the two child processes):

Game 1:
1147: Cooperate
1148: Defect
Game 2:
1147: Cooperate
1148: Cooperate
Game 3:
1147: Defect
1148: Defect
Game 4:
1147: Cooperate
1148: Defect

Score:
1147: 26.5 years
1148: 6.5 years

Remember to error-check all system calls: check return values for success, and use *perror* when possible on failure.
Use the Unix shell command "*script*" to record the entire session. Exit of the scripting session by typing *exit*.
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. You can develop your code on your own linux/Unix environment. However, do not forget that your program will be run and tested on the *eecslab-X* virtual machine, so make sure that it compiles 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 "*as3.zip*"