Lab 6.2 : IPC Mechanism and Synchronization using Semaphores

November 14, 2016

Objective : The purpose of this lab is to teach students to use named Unix pipes and introduce to System V IPC through shared memory and message queues.

You should complete lab 6.1 and 7.1 before you proceed with this lab.

Write a C program to implement the following game. The parent program P first creates two pipes, and then spawns two child processes C and D. One of the two pipes is meant for communications between P and C, and the other for communications between P and D. Now, a loop runs as follows. In each iteration (also called round), P first randomly chooses one of the two flags: MIN and MAX (the choice randomly varies from one iteration to another). Each of the two child processes C and D generates a random positive integer and sends that to P via its pipe. P reads the two integers; let these be c and d. If P has chosen MIN, then the child who sent the smaller of c and d gets one point. If P has chosen MAX, then the sender of the larger of c and d gets one point. If c = d, then this round is ignored. The child process who first obtains ten points wins the game. When the game ends, P sends a user-defined signal to both C and D, and the child processes exit after handling the signal (in order to know who was the winner). After C and D exit, the parent process P exits. During each iteration of the game, P should print appropriate messages (like P's choice of the flag, the integers received from C and D, which child gets the point, the current scores of C and D) in order to let the user know how the game is going on. Name your program childsgame.c .