Socratis Katehis

11/12/2017

Dad – Son Problem

After analyzing the execution traces of the original “bank.c”, I saw that there was a race condition to getting the account balance to zero before the dad added money to the account. When the account got to zero, the two sons would keep reading and writing into the account until the dad added a balance. Also, because the sons were accessing separately, withdrawals wouldn’t be reflected correctly.

This behavior is unwanted. We want each process to have exclusive access to the account at a time (dad, son 1, son 2) and when the account is empty the children should wait for their father to add money to the account. Also, processes should be accessing the “same” account, so a shared memory location should be given to all three processes. This would take care of the observed problems, and can be solved with semaphores and mutexes.

A buffer kept track of the three processes through their PID’s and kept track of the waiting of each process by incrementing wait times. Before entering the critical section, PID’s would be set into the buffer. If the process wanted to enter the critical section, but couldn’t, the following process would identify that those process/processes are waiting in the buffer and would increment the wait time for each of them. If the process can go into the critical section, the process will check if other processes are waiting to enter the critical section. When the process leaves the critical section, the buffer values holding the PID of the process is set to zero and allows other processes to enter the CS.

Below are the waiting times for the processes that wanted to enter the critical section.

|  |  |  |  |
| --- | --- | --- | --- |
| Number of Attempts | Dad Process Waited | Son1 Process Waited | Son2 Process Waited |
| 5 | 6 | 6 | 5 |
| 10 | 8 | 10 | 9 |
| 15 | 8 | 12 | 9 |
| 20 | 10 | 15 | 15 |
| 25 | 8 | 17 | 17 |