Jerrely Vizueta

Operating Systems

Lab 6

During the lab I used the sem.h file to implement changes in the bank.c file. The requirements were:

* Prevent race conditions
* Prevent a son from withdrawing money when there is no balance.
* Prevent undefined outputs like negative balance
* Prevent a process from continuously requesting access to the shared memory. Prevent unnecessary cpu cycles.
* The problem should be solved with as few semaphore variables as possible.

In order to complete the tasks at hand I created two semaphores: semEmpty and semFull. semEmpty and SemFull would be initiated with a count of 1. The function of semEmpty revolves around supplying the account with funds so the starter process that would use V(semEmpty) would be the parent.

The 2 children started the semFull, whoever got it first would be allowed to access an “ATM machine”. There is an if statement on each process to check if the amount requested is higher than the balance, if it is then we decrement semEmpty which would stop the process until the parent finishes depositing, this would be a failed attempt to withdraw. If there is enough money in the account then it proceeds to withdraw it and increases semFull to allow the next one waiting on the semFull to go. Also if the parent doesn’t deposit any more money and there are no funds for successful withdrawals then the semEmpty is immediately decremented and moves on without withdrawing until all attempts are wasted..

The if statements prevent the withdrawal of amounts above the balance so there is never a balance. The use of two semaphores allows two processes to view the balance, get an “ATM machine”. The semaphores allow the children and the parent to stop when incremented or decremented so there are no continuous requests to the shared memory.