Jia Ming Ma

CSc 322 Task 5 Dad-Son Problem

My solution prevents race condition because I am using semaphores to synchronize all the processes so that they do not all update at once. Whenever a process wants to access the critical section, the balance.txt file, the program would call the SEM_P() function. It will decrement the semaphore by the sem_op (usually -1) if it is bigger than 0 (meaning there is resource for the process to access). Then when this process is inside the critical section, if other processes want to access it, they wouldn't be able to because the semaphore is 0, which tells them there is no resources for them to access right now. They will be able to access it after the first process releases it by calling the SEM_V() function which will increment our semaphore by sem_op (usually +1), returning the resource/critical section back by setting semaphore to 1.

Semaphores allow us to know when a process is interested in the critical section so that we can know when to increment their T(P) counter. I did this by making an interest.txt file with 3 lines each representing the interest of Dad, Son 1 and Son 2 with values 0 or 1, meaning false or true. Whenever a process is waiting for to enter the critical section, their respective interest would be updated to 1 and when they leave the critical section, the interest would be updated to 0. And then when a process enters the critical section, it would read the interest.txt file and if the processes besides itself have interest of 1, it would update their T(P) counter which is in the attempt.txt file. In attempt.txt, line 0 represents the attempts, line 1 represents T(P) of Dad, line 2 is T(P) of Son 1 and line 3 is T(P) of Son 2.

N	T(Dad)	T(Son 1)	T(Son 2)
5	8	14	11
7	12	17	16
9	16	18	19
15	22	20	21
20	21	20	20

N_Att is 20 for all trials