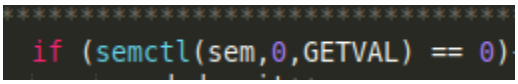
Lianming Wu
CSC332 LAB5
Instructor: Prajwal Khatiwada,
Dad-Son Problem

Reports:

I. The Critical Region

- Semaphore is a kind of mutual exclusion lock that allows only a certain amount of process to enter the critical region depends on the number of resources we have. By setting the semaphore cap to one, we replicate the property of a mutex, which allows just one process to enter the critical region.
- Critical region in this task is the balance and attempt files shared between the processes. So, what we needed to do is to lock the files whenever a process is accessing the critical region, i.e. no sons could access the balance file while Dad is updating it, so sons could not overwrite the balance Dad just updated and mess up the result.
- I put P(sem) before opening the first file and put V(sem) which unlocks the critical region at the end of the while and for loops when the file handling procedures are all done. As there were no exec or exit function calls inside the while loop, V(sem) is guaranteed to be reached so other process could access the critical region.

II. Time measurement

- I used two parameters to measure the T(P), one is the number of times when a process tries to access the semaphore but can't enter. This is done by calling the 

which returns the current value of the semaphore, if it's zero, current process need to wait for other process, and we increment the wait time counter.
- Another parameter is the number of cycle elapsed before and after we called P(sem), this measures how many cycles is burned in the process of calling P(sem). Done by using the <time.h> clock( ) function.

N	N = 5	N = 7	N = 10	N =3	N = 2
Dad	4 / 103.000	6 / 360.000	9 / 303.000	2 / 74.000	1 / 34.000
Son1	11/ 216.000	11 / 372.00	11/ 208.00	8 / 232.000	7/ 225.000
Son2	11/ 312.000	11 / 537.00	11/ 294.00	8 / 290.000	6/ 211.000
Attempt left	0	0	0	6	9

Time/Cycle

N_ATT remains 20, which is the number of times Son1 and Son2 have permission to withdraw money.