There are 3 forked processes that want to access the balance file: SON\_1, SON\_2, and dad. Here we have a problem that all 3 process are while loops trying to access the same file balance.

1. Dad process should open balance file, think for random time and add 60 dollars to the balance. Dad exits when he updates bank account N times.
2. SON\_1 reads balance, if there is money takes 20 dollars and updates balance.
3. SON\_2 is same as son 1

SON\_1 and SON\_2 attempt to take money N\_att time. Both share the N\_att value from attempt file. Both exit when attempt file has 0.

The possible problems I saw was that sometimes two or 3 process read the file. For example son\_1 opens a file and reads balance. Then due to race condition son\_2 opens the balance too. Now son\_2 updates the balance. Then son\_1 gets the turn and updates the file. Now we have son\_2 balance update deleted. Similarly son\_1 and son\_2 open and update attempt file with race condition so they end up getting more attempt than designated since sometimes one son overwrites the attempt update by other.

So we need to have semaphore in balance file access and attempt file access. But since sons read attempt file before balance file we can use same semaphore for both files. So I created semaphore of mutex. Then I added P(mutex) before a process opens file anf V(mutex) after process closes the file. So file can be updated without synchronization problem. So we can only have following output from inside critical setion:

1. When dad is in:
   1. Dear old dad is trying to do update, so start semaphore.
   2. Dear old dad reads balance = 120
   3. Dear old dad writes new balance = 180
   4. Dear old dad is done doing update so exit semaphore.
2. When son 1 is in
   1. SON\_1 is trying to access file so enter semaphore.
   2. Poor SON\_1 wants to withdraw money.
   3. Poor SON\_1 reads balance. Available Balance: 80
   4. Poor SON\_1 write new balance: 60
   5. poor SON\_1 done doing update so exit semaphore.
3. When son 2 is in
   1. SON\_2 is trying to access file so enter semaphore.
   2. Poor SON\_2 wants to withdraw money.
   3. Poor SON\_2 reads balance. Available Balance: 120
   4. Poor SON\_2 write new balance: 100
   5. poor SON\_2 done doing update so exit semaphore

These lines are from inside critical section. We can only have one sequence of these print out at a time. We can have other process outside critical section running so we can have some sequence like

* SON\_2 is trying to access file so enter semaphore.
* Process(pid = 3754) exited with the status 0.
* SON\_1 is trying to access file so enter semaphore.
* Process(pid = 3753) exited with the status 0.

Or

* Second Son's Pid: 3754
* SON\_2 is trying to access file so enter semaphore.
* Poor SON\_2 wants to withdraw money.
* Poor SON\_2 reads balance. Available Balance: 100
* First Son's Pid: 3753
* Poor SON\_2 write new balance: 80
* poor SON\_2 done doing update so exit semaphore.
* SON\_1 is trying to access file so enter semaphore.

It is not violation since the lines indicating process exited or started are not from critical section and therefore do not affect the syncrnoization.