Problem 1:

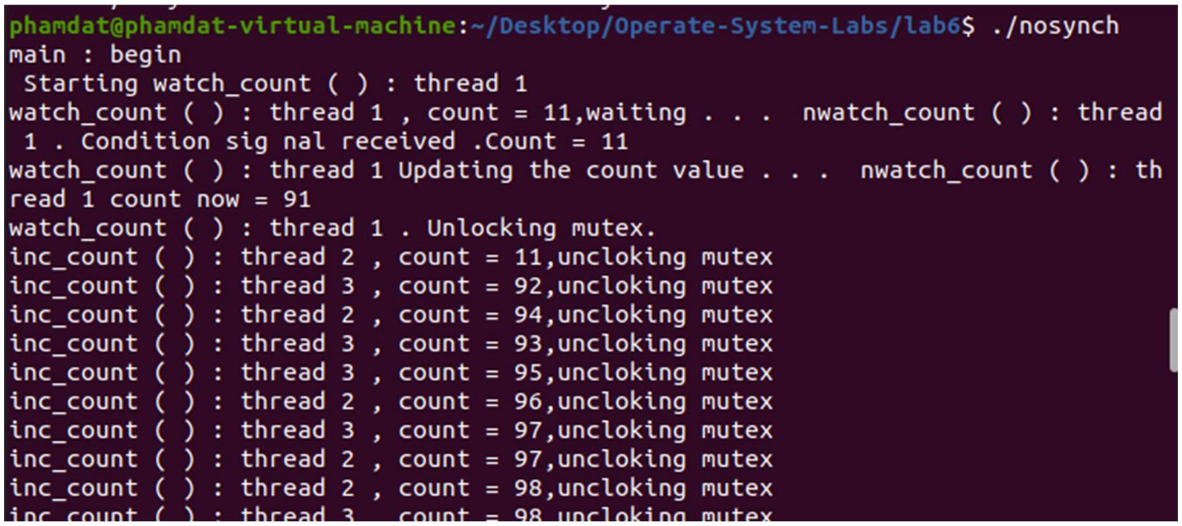
There are two cases can happen when husband and wife call function at the same time.

On the one hand, the money is taken from the function withdraw() will not subtract to the original money in account. It happens when variable in withdraw() set value before deposit().

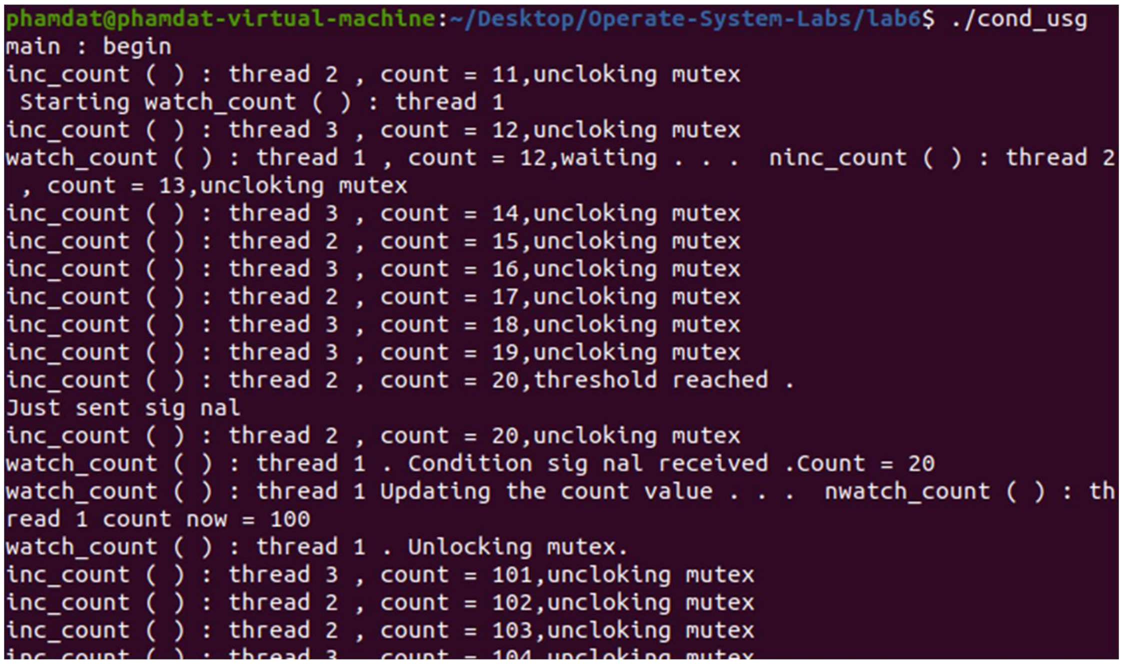
On other hand, the money which wife deposited will not add to the money in account. It happens when variable in deposit() set value before withdraw().

We can use mutex, condition variable or semaphore to solve race condition.

Problem 2:



```
phamdat@phamdat-virtual-machine:~/Desktop/Operate-System-Labs/lab6$ ./nosynch
main : begin
 Starting watch_count ( ) : thread 1
watch_count ( ) : thread 1 , count = 11,waiting . . .  nwatch_count ( ) : thread
 1 . Condition sig nal received .Count = 11
watch_count ( ) : thread 1 Updating the count value . . .  nwatch_count ( ) : th
read 1 count now = 91
watch_count ( ) : thread 1 . Unlocking mutex.
inc_count ( ) : thread 2 , count = 11,uncloking mutex
inc_count ( ) : thread 3 , count = 92,uncloking mutex
inc_count ( ) : thread 2 , count = 94,uncloking mutex
inc_count ( ) : thread 3 , count = 93,uncloking mutex
inc_count ( ) : thread 3 , count = 95,uncloking mutex
inc_count ( ) : thread 2 , count = 96,uncloking mutex
inc_count ( ) : thread 3 , count = 97,uncloking mutex
inc_count ( ) : thread 2 , count = 97,uncloking mutex
inc_count ( ) : thread 2 , count = 98,uncloking mutex
inc_count ( ) : thread 3 , count = 98,uncloking mutex
```



```
phamdat@phamdat-virtual-machine:~/Desktop/Operate-System-Labs/lab6$ ./cond_usg
main : begin
inc_count ( ) : thread 2 , count = 11,uncloking mutex
 Starting watch_count ( ) : thread 1
inc_count ( ) : thread 3 , count = 12,uncloking mutex
watch_count ( ) : thread 1 , count = 12,waiting . . .  ninc_count ( ) : thread 2
 , count = 13,uncloking mutex
inc_count ( ) : thread 3 , count = 14,uncloking mutex
inc_count ( ) : thread 2 , count = 15,uncloking mutex
inc_count ( ) : thread 3 , count = 16,uncloking mutex
inc_count ( ) : thread 2 , count = 17,uncloking mutex
inc_count ( ) : thread 3 , count = 18,uncloking mutex
inc_count ( ) : thread 3 , count = 19,uncloking mutex
inc_count ( ) : thread 2 , count = 20,threshold reached .
Just sent sig nal
inc_count ( ) : thread 2 , count = 20,uncloking mutex
watch_count ( ) : thread 1 . Condition sig nal received .Count = 20
watch_count ( ) : thread 1 Updating the count value . . .  nwatch_count ( ) : th
read 1 count now = 100
watch_count ( ) : thread 1 . Unlocking mutex.
inc_count ( ) : thread 3 , count = 101,uncloking mutex
inc_count ( ) : thread 2 , count = 102,uncloking mutex
inc_count ( ) : thread 2 , count = 103,uncloking mutex
inc_count ( ) : thread 3 , count = 104,uncloking mutex
```

Variable count increases from 11 to 91 instead of 12, 13, 14 … 20, 100…

This is because we removed “pthread_cond_wait(&count_threshold_cv , &count_mutex) ;” in watch_count(). It is no longer has to waiting for signal from pthread_cond_signal(&count_threshold_cv) ; (in inc_count()) which will send when meet condition count == COUNT_LIMIT.

- ⇨ It means count+=80 (in watch_count()) execute instead of count++ (in inc_count);