# Lab4 notes

Lab4 has a user program shm_cnt.c

Running it without implementing shm_open() and shm_close() system calls prints:

```
$ shm_cnt
Counter in Parent is 1 at address 0
Counter in Parent is 1001 at address 0
Counter in Parent is 2001 at address 0
Counter in Parent is 3001 at address 0
Counter in Parent is 4001 at address 0
Counter in Parent is 5001 at address 0
Counter in Parent is 6001 at address 0
Counter in Parent is 7001 at address 0
Counter in Parent is 8001 at address 0
Counter in Parent is 9001 at address 0
Counter in parent is 10000
Counter in Child is 1 at address 0
Counter in Child is 1001 at address 0
Counter in Child is 2001 at address 0
Counter in Child is 3001 at address 0
Counter in Child is 4001 at address 0
Counter in Child is 5001 at address 0
Counter in Child is 6001 at address 0
Counter in Child is 7001 at address 0
Counter in Child is 8001 at address 0
Counter in Child is 9001 at address 0
Counter in child is 10000
```

Because they are processes and not threads.

Each of the parent and child processes have their own memory and each increases the counter separately. So at the end they both display the same count (10,000) The count is printed out once the counter is divisible by 1000.

Actually, count is printed when loop counter i is divisible by 1000 and the value of counter in every iteration i is (i+1).

After implementing shm_open() and shm_close() system calls prints:

```
$ shm_cnt
Counter in Parent is 1 at address 4000
Counter in Parent is 1001 at address 4000
Counter in Parent is 2001 at address 4000
Counter in Parent is 3001 at address 4000
Counter in Child is 3002 at address 4000
Counter in Child is 4002 at address 4000
Counter in Child is 5002 at address 4000
Counter in Child is 6002 at address 4000
Counter in Parent is 7002 at address 4000
Counter in Parent is 8002 at address 4000
Counter in Parent is 9002 at address 4000
Counter in Parent is 10002 at address 4000
Counter in Child is 11002 at address 4000
Counter in Child is 12002 at address 4000
Counter in Child is 13002 at address 4000
Counter in Child is 14002 at address 4000
Counter in Child is 15002 at address 4000
Counter in Child is 16002 at address 4000
Counter in child is 17001
Counter in Parent is 18001 at address 4000
Counter in Parent is 19001 at address 4000
Counter in parent is 20000
```

Each of the parent and child processes share the memory location and they both increase the same counter. The lock guarantees that no race condition happens. At the end of the child process the, counter is 17,000 while at the end of the parent process the counter reaches 20,000.

Notes:

1) These outputs are not deterministic. The final output of either the parent or the child will always be 20,000 but any of them could reach that number while the other will have finished at some number less than 20,000. Why?)

2) Notice that both processes always have 1 or 2 in the first digit when printing the number, this is because the printf statement is not protected by the lock.
   Try placing this statement:

*if(i%1000 == 0) printf(1,"Counter in %s is %d at address %x\n",pid? "Parent" : "Child", counter->cnt, counter);*

right after *uacquire(&(counter->lock));*