Evan Nichols

EECS 678

Lab 5 Report

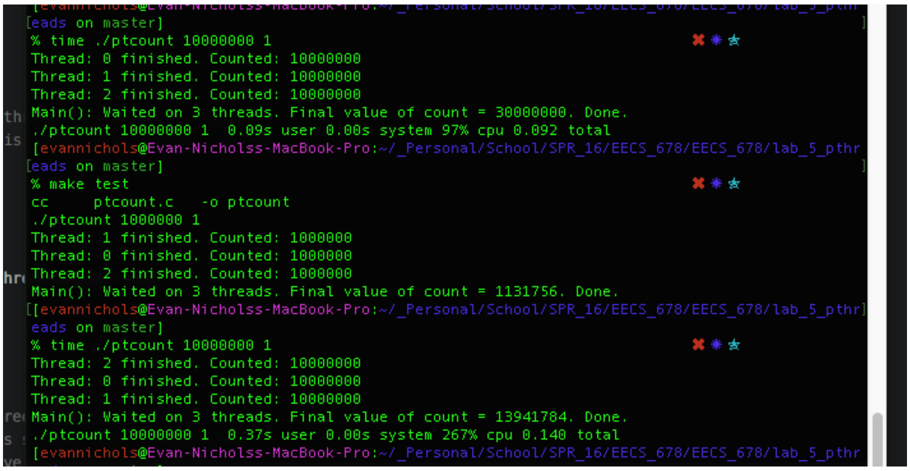
1. Each thread updates the count using the = and + operators in C. A single line of instructions incrementing the count variable actually translates to three machine level instructions - mov, add and mov. This can lead to an *interleaving* problem if multiple threads are accessing a shared data variable (e.g. count) without the use of locks. This can be solved by using the pthread_mutex_lock before a thread enters its critical section, and calling pthread_mutex_unlock upon completion

2. As stated above, it depends on the *interleaving* done by the scheduler. The scheduler has the ability to stop a running process, with the intention of resuming it later, and switching context to another process in the ready state.

3. Why are the local variables that are printed out always consistent? When you create a new thread, only the components of the process control block that are necessary to create a new thread of control of are actually allocated for the child. It gets a new PC, registers image, user and kernel stack. A local variable will not be different because, well, it's local. Only a single thread has access to it, and it will maintain an accurate counter.

4. How does your solution ensure the final value of count will always be consistent (with any loop bound and increment values)? It uses the pthread_mutex_lock() function immediately before the thread enters the for loop to increment count, and uses the pthread_mutex_unlock() function right after completition.

5. The times are so different because not implementing locks and unlocks causes for a lot more work on the scheduler's end, interleaving processes. While the lock and unlock are functions which take time to execute, they actually help reduce the overall runtime because it ensures only 1 thread at a time has access to the shared data variable. A screenshot is below highlighting the differences:



```
[eads on master]                                                              ]
% time ./ptcount 10000000 1                                         ✖ ✹ ✭
Thread: 0 finished. Counted: 10000000
Thread: 1 finished. Counted: 10000000
Thread: 2 finished. Counted: 10000000
Main(): Waited on 3 threads. Final value of count = 30000000. Done.
./ptcount 10000000 1  0.09s user 0.00s system 97% cpu 0.092 total
[evannichols@Evan-Nicholss-MacBook-Pro:~/_Personal/School/SPR_16/EECS_678/EECS_678/lab_5_pthr
[eads on master]                                                              ]
% make test                                                         ✖ ✹ ✭
cc     ptcount.c   -o ptcount
./ptcount 1000000 1
Thread: 1 finished. Counted: 1000000
Thread: 0 finished. Counted: 1000000
Thread: 2 finished. Counted: 1000000
Main(): Waited on 3 threads. Final value of count = 1131756. Done.
[[evannichols@Evan-Nicholss-MacBook-Pro:~/_Personal/School/SPR_16/EECS_678/EECS_678/lab_5_pthr]
eads on master]
% time ./ptcount 10000000 1                                         ✖ ✹ ✭
Thread: 2 finished. Counted: 10000000
Thread: 0 finished. Counted: 10000000
Thread: 1 finished. Counted: 10000000
Main(): Waited on 3 threads. Final value of count = 13941784. Done.
./ptcount 10000000 1  0.37s user 0.00s system 267% cpu 0.140 total
[evannichols@Evan-Nicholss-MacBook-Pro:~/_Personal/School/SPR_16/EECS_678/EECS_678/lab_5_pthr
```