CSE 422- Studio 10

1. Selamawit Tegegn(sntegegn@wustl.edu) , Leiquan Pan(lpan22@wustl.edu), yiying Lu([lu.yiying@wustl.edu](mailto:lu.yiying@wustl.edu))

2.

Use function kthread\_bind() to bind a thread to specific core. i.e. kthread\_bind(task\_pointer\_00,0), which means assign task\_pointer\_00 thread run on core 0

3.

(1) each thread individually run 1000000 times to increment shared\_data, so the result should be 4000000

(2)when more than two threads load the shared\_data, they may overwrite the other’s change to shared\_data.

4. [ 290.174935] The shared\_data is: 1103056

5.

atomic\_t share\_data;

atomic\_add(1,&share\_data);

atomic\_set(&share\_data,0);

printk(KERN\_INFO "The shared\_data is: %d", atomic\_read(&share\_data));

6.

[ 99.594885] The shared\_data is: 4000000

[ 218.169448] The shared\_data is: 4000000

[ 408.799434] The shared\_data is: 4000000

7.

[ 1675.081832] share\_data at the beginning in 0: 0

[ 1675.081837] share\_data at the beginning in 1: 0

[ 1675.083450] share\_data at the beginning in 2: 0

[ 1675.232498] share\_data at the beginning in 3: 25936

[ 1675.244111] share data at the end in 2: 3545372

[ 1675.244170] share data at the end in 3: 3908869

[ 1675.245279] share data at the end in 0: 3911619

[ 1675.245279] share data at the end in 1: 4000000

[ 1683.739080] The shared\_data is: 4000000

Thread 0:0.163447

Thread 1:0.657243

Thread 2 : 0.16066

Thread 3 : 0.01168

8.

volatile int  share\_data=0;

DEFINE\_MUTEX(mutex\_name);

mutex\_lock(&mutex\_name);

share\_data++;

mutex\_unlock(&mutex\_name);

9.

[ 2621.319520] share\_data at the beginning in 0: 0

[ 2621.319526] share\_data at the beginning in 1: 0

[ 2621.320147] share\_data at the beginning in 2: 0

[ 2621.953341] share\_data at the beginning in 3: 1071

[ 2622.013264] share data at the end in 3: 2683833

[ 2622.090024] share data at the end in 2: 3109668

[ 2622.100209] share data at the end in 0: 3840553

[ 2622.100209] share data at the end in 1: 4000000

[ 2639.500784] The shared\_data is: 4000000

Thread 0:0.780689

Thread 1:0.780683

Thread 2 : 0.769877

Thread 3 : 0.059923

(1) Atomic\_t performs better on time perspective.

(2) Impossible to suspend a thread during atomic\_t the memory bus lock. It’s possible to suspend a thread during a mutex lock. Most nontrivial operations cannot be made atomic, so you must either use a lock to block other threads from operating while the critical section executes, or else you must carefully design a [lock-free algorithm](https://en.wikipedia.org/wiki/Non-blocking_algorithm) that ensures that all the critical state-changing s can be safely implemented using atomic operations.