ASSIGNMENT 2 REPORT
NAME: I-HSINE HUANG
SID: 862057578
<request>
• clone() system call: 15%
• thread_create(...) function: 15%
• Lock initialization, acquire & release functions for spin lock: 10%
• Lock initialization, acquire & release functions for array lock: 10%
• SeqLock implementation: 10%
• Frisbee test program: 10%
• Spinlock demo (see details in test example): 10%
• Lock array demo (see details in test example):10%
• SeqLock demo (see details in test example): 10%
• MCS lock demo(Extra credit): 5%

In this assignment, we need to write our clone() system call, and a thread_create() function to create thread library. And thread_create will use the system call.
clone():
we need to change new process tf->esp and tf->ebp
and make them sure the same file system
Finally use safestrcpy to memory copy

```
clone(void* stack,int size)//BR
{
  numofthread ++;//one clone add one thread
 // cprintf("stack %d\n",stack);//original place
  // cprintf("this is clone %d\n",size);
  int i,pid;
  struct proc *newp;//new process and in here is new thread
  if((newp=allocproc())==0)
    return -1;
  struct proc *curproc = myproc();
  newp -> pgdir = curproc ->pgdir;
  newp -> sz  = curproc ->sz; // size of process
  newp -> parent = curproc;//parent will be original process
  *newp -> tf = *curproc -> tf;
  newp -> thread = 1;
  newp -> tf -> eax = 0 ;// fork in child will retunr 0
  /*
   *in stack esp is the top of stack ebp is the down of the stack
    but in memory address, stack grows from top to down
   *
   *  */

  void *down_copy = (void*)curproc->tf->ebp +16;
  void *top_copy = (void*)curproc->tf->esp;
 // cprintf("esp :%d ebp %d\n",top_copy,down_copy);
  uint copysize = (uint)(down_copy - top_copy);
  newp->tf->esp = (uint) (stack - copysize);
  newp->tf->ebp = (uint) (stack -16);
 // cprintf("new stack esp %d ebp: %d\n",newp->tf->esp,newp->tf->ebp);
  memmove(stack-copysize,top_copy,copysize);
  for(i=0;i<NOFILE;i++)
  {
    if(curproc->ofile[i])
      newp->ofile[i]=filedup(curproc->ofile[i]);
  }
  newp->cwd = idup(curproc->cwd);
  pid = newp->pid;
  newp->state = RUNNABLE;
  safestrcpy(newp->name, curproc->name, sizeof(curproc->name));
  // cprintf("In clone function this is a pid %d\n",pid);
  return pid;
}
```

And I make a thead.c to do all thread_create and different kinds of lock release and init for different lock.
And there are frisbee.c frisbee_array.c and frisbee_seqlock.c to help me test my code.
in Spinlock
Each process try to get the resource and if the process got the resource is not the process thread we want. It will release the resource until the correct thread get the resource.
In Array based queue lock.
Each, which thread works depends on the array state. It will show this time which tickets will work. So each thread try to get the tickets but if the tickets are not allocated to right thread. The thread will give up the ticket. And in the state array, after each tickets complete their mission, the work ticket will be the next ticket.
In seqlock.
There are two important function, reader and writer. And writer has good priority. Every threads will stay in the reader's do while loop and keep reading until the data is updated by the function writer and the right thread gets out of the loop and prints out what we want. Next time will be the writer time. So the sequnce will be W R W R W R. And it works more smoothly than the other two lock.

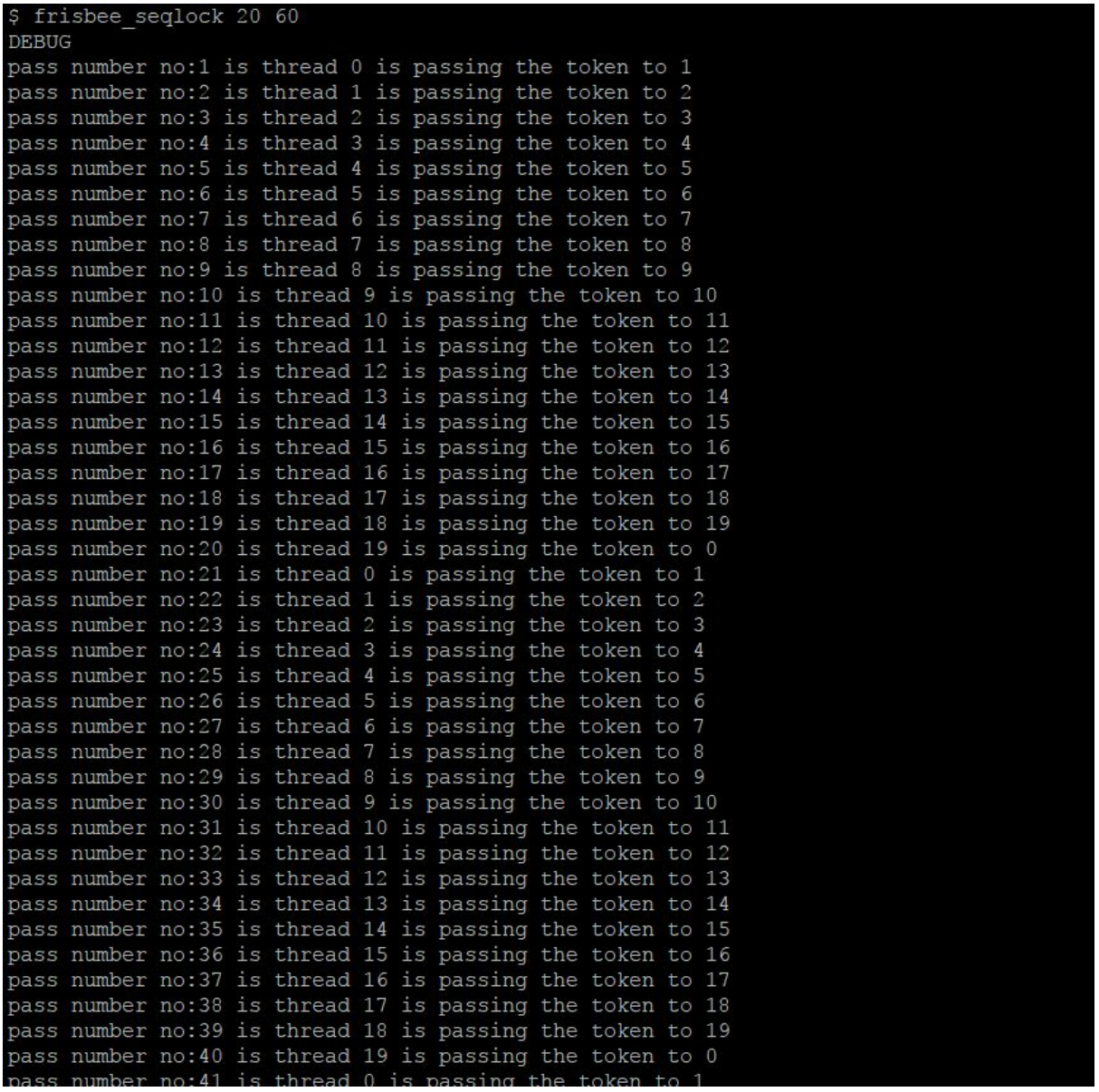
result for frisbee

```
init: starting sh
$ frisbee 20 60
pass number no:1 is thread 0 is passing the token to  1
pass number no:2 is thread 1 is passing the token to  2
pass number no:3 is thread 2 is passing the token to  3
pass number no:4 is thread 3 is passing the token to  4
pass number no:5 is thread 4 is passing the token to  5
pass number no:6 is thread 5 is passing the token to  6
pass number no:7 is thread 6 is passing the token to  7
pass number no:8 is thread 7 is passing the token to  8
pass number no:9 is thread 8 is passing the token to  9
pass number no:10 is thread 9 is passing the token to  10
pass number no:11 is thread 10 is passing the token to  11
pass number no:12 is thread 11 is passing the token to  12
pass number no:13 is thread 12 is passing the token to  13
pass number no:14 is thread 13 is passing the token to  14
pass number no:15 is thread 14 is passing the token to  15
pass number no:16 is thread 15 is passing the token to  16
pass number no:17 is thread 16 is passing the token to  17
pass number no:18 is thread 17 is passing the token to  18
pass number no:19 is thread 18 is passing the token to  19
pass number no:20 is thread 19 is passing the token to  0
pass number no:21 is thread 0 is passing the token to  1
pass number no:22 is thread 1 is passing the token to  2
pass number no:23 is thread 2 is passing the token to  3
pass number no:24 is thread 3 is passing the token to  4
pass number no:25 is thread 4 is passing the token to  5
pass number no:26 is thread 5 is passing the token to  6
pass number no:27 is thread 6 is passing the token to  7
pass number no:28 is thread 7 is passing the token to  8
pass number no:29 is thread 8 is passing the token to  9
pass number no:30 is thread 9 is passing the token to  10
pass number no:31 is thread 10 is passing the token to  11
pass number no:32 is thread 11 is passing the token to  12
pass number no:33 is thread 12 is passing the token to  13
pass number no:34 is thread 13 is passing the token to  14
pass number no:35 is thread 14 is passing the token to  15
pass number no:36 is thread 15 is passing the token to  16
pass number no:37 is thread 16 is passing the token to  17
pass number no:38 is thread 17 is passing the token to  18
pass number no:39 is thread 18 is passing the token to  19
pass number no:40 is thread 19 is passing the token to  0
pass number no:41 is thread 0 is passing the token to  1
pass number no:42 is thread 1 is passing the token to  2
pass number no:43 is thread 2 is passing the token to  3
pass number no:44 is thread 3 is passing the token to  4
```

result for frisbee_array

```
$ frisbee_array 20 60
DEBUG
pass number is no:1 thread 0 is passing the token to thread 1
pass number is no:2 thread 1 is passing the token to thread 2
pass number is no:3 thread 2 is passing the token to thread 3
pass number is no:4 thread 3 is passing the token to thread 4
pass number is no:5 thread 4 is passing the token to thread 5
pass number is no:6 thread 5 is passing the token to thread 6
pass number is no:7 thread 6 is passing the token to thread 7
pass number is no:8 thread 7 is passing the token to thread 8
pass number is no:9 thread 8 is passing the token to thread 9
pass number is no:10 thread 9 is passing the token to thread 10
pass number is no:11 thread 10 is passing the token to thread 11
pass number is no:12 thread 11 is passing the token to thread 12
pass number is no:13 thread 12 is passing the token to thread 13
pass number is no:14 thread 13 is passing the token to thread 14
pass number is no:15 thread 14 is passing the token to thread 15
pass number is no:16 thread 15 is passing the token to thread 16
pass number is no:17 thread 16 is passing the token to thread 17
pass number is no:18 thread 17 is passing the token to thread 18
pass number is no:19 thread 18 is passing the token to thread 19
pass number is no:20 thread 19 is passing the token to thread 0
pass number is no:21 thread 0 is passing the token to thread 1
pass number is no:22 thread 1 is passing the token to thread 2
pass number is no:23 thread 2 is passing the token to thread 3
pass number is no:24 thread 3 is passing the token to thread 4
pass number is no:25 thread 4 is passing the token to thread 5
pass number is no:26 thread 5 is passing the token to thread 6
pass number is no:27 thread 6 is passing the token to thread 7
pass number is no:28 thread 7 is passing the token to thread 8
pass number is no:29 thread 8 is passing the token to thread 9
pass number is no:30 thread 9 is passing the token to thread 10
pass number is no:31 thread 10 is passing the token to thread 11
pass number is no:32 thread 11 is passing the token to thread 12
pass number is no:33 thread 12 is passing the token to thread 13
pass number is no:34 thread 13 is passing the token to thread 14
pass number is no:35 thread 14 is passing the token to thread 15
pass number is no:36 thread 15 is passing the token to thread 16
pass number is no:37 thread 16 is passing the token to thread 17
pass number is no:38 thread 17 is passing the token to thread 18
pass number is no:39 thread 18 is passing the token to thread 19
pass number is no:40 thread 19 is passing the token to thread 0
```

result for frisbee_lock



My github: it is easier to watch the code
https://github.com/IHSIENHUANG/xv6-kernel-thread