# CIS657 Fall 2018
# Assignment Disclosure Form

Assignment #: Programming assignment 2

Name: Nidhi Surya Prakash

1. Did you consult with anyone other than instructor or TA/grader on parts of this assignment?
   If Yes, please give the details.

Yes, I did consult.

Vikas Agrawal
Malhar Ujawane
Manushi Sheth
Apurva Kemkar

2. Did you consult an outside source such as an Internet forum or a book on parts of this assignment?
   If Yes, please give the details.

References:

1) https://www.student.cs.uwaterloo.ca/~cs350/common/NachosTutorialF06.pdf
2) https://inst.eecs.berkeley.edu/~cs162/fa13/Nachos/nachos.pdf
3) https://en.wikipedia.org/wiki/Job_scheduler
4) http://www.cplusplus.com

I assert that, to the best of my knowledge, the information on this sheet is true.

Signature:___Nidhi Surya Prakash_____ Date : 11/18/2018

# CIS 657 Programming Assignment 2

Nidhi Surya Prakash
SUID: 215895619

## Objective:

## What has been developed:

The main objective of this assignment would be to furthermore understand and learn the concepts of C++ in Nachos by implementing system call routines for user programs. We are given 4 tasks where each one would support the other.

Task1: **`System Calls for user program`**

We are implementing 2 system calls – Fork_POS & Wait_POS and 1 function named Exit_POS.

In Fork_POS, we are enabling the user programs to spawn and fork a child thread. Fork will start a new process that runs a user function specified by the argument of the call. Fork_POS takes an argument which is a pointer to a function. The function must be compiled as part of the user program that is currently running. By making this system call Fork_POS, the user program expects that a new thread will be generated for use by the user program; and this thread will run the function in an address space that is an exact copy of the current one. To make the system call for the user program, we find the entry point of the function which is passed as the parameter. The parameter is determined by the cross-compiler which produces executable code from the user source program. We then start with the file exception.cc to see that this entry point, which is an address in the executable code's address space, is already loaded into register 4 when the trap to the exception handler occurs. Here we also create a map to keep track of the parent and its appropriate child threads.

In Wait_POS, we focus on providing the routine for the user program to wait for the execution of its child thread. Here, we put the parent process to sleep and put the child thread into running. It takes an integer parameter which is child's process ID. Once the child runs the test function provided, it encounters the function Exit_POS. At this stage we wake up the parent of the child and then Finish the child. Here, we also make a few validations that care of handling certain errors that may arise because of invalid child process ID or no parent process waiting.

Code:

```
/*******************************/
   //Fork_POS()enables the user program to fork a kernel child thread. It takes integer 1, 2 or 3 as
   //parameter, creates a corresponding kernel thread, which executes one of the test functions basing
   //on the parameter value, and returns an integer as the process ID of the child. For example, when
   //child = Fork_POS(2) is called in a user program, a kernel thread will be created in ExceptionHandler()
   //and this thread will call the ForkTest2 function (using the Fork()call of thread.cc).
   //The child process ID will be assigned to child.


   //FORK POS - thread gets created,Forks and placed into the readytorun list; control goes back to
   //main process
   // Require: id as parameter - getid getter method and variable
   //2)switch case or if statement to run test function according to function pointer.

      case SC_FORK_POS:
         {

            int num = (int)kernel->machine->ReadRegister(4);
            //result = 4;
            //cout<<"Entering fork_os and val of v is:"<<v<<"\n";
            Thread *t = new Thread("forked thread");
            int id = t->getThreadId();
            switch(num){
            case 1:
            t->Fork((VoidFunctionPtr) ForkTest1,(void *)t->getThreadId());
            break;
            case 2:
            t->Fork((VoidFunctionPtr) ForkTest2,(void *)t->getThreadId());
            break;
            case 3:
            t->Fork((VoidFunctionPtr) ForkTest3,(void *)t->getThreadId());
            break;
            default:
            break;
            }
```

```
            kernel->machine->WriteRegister(2, (int)t->getThreadId());
            //int i = childTh->getThreadID;
            kernel->mapTracker[t->getThreadId()] = kernel->currentThread;

            /* Modify return point */
            {
            /* set previous programm counter (debugging only)*/
               kernel->machine->WriteRegister(PrevPCReg, kernel->machine->ReadRegister(PCReg));
            /* set programm counter to next instruction (all Instructions are 4 byte wide)*/
               kernel->machine->WriteRegister(PCReg, kernel->machine->ReadRegister(PCReg) + 4);
            /* set next programm counter for brach execution */
               kernel->machine->WriteRegister(NextPCReg, kernel->machine->ReadRegister(PCReg)+4);
            }
            return;
            ASSERTNOTREACHED();
         }
      break;
```

```
//•   Wait_POS() provides the routine for the user program to wait for the execution of its child thread.
   //It takes the child process ID as parameter. The parent will be put into sleep until the child awakens it.
   //The child wakes up the parent when it finishes by calling Exit_POS, which you'll implement. Wait_POS() and
   //Exit_POS() should be able to handle certain errors such as invalid child process ID or no parent process waiting.

   //wait fork: the parent process is put to sleep and searches the findNextTorun list
   //runs the next element,
   //Also implementing the validations to make sure if parent of the chld exists in the map and
   //if the child belongs to the right parent

      case SC_WAIT_POS:
         {
            int c = (int)kernel->machine->ReadRegister(4);
            if ( kernel->mapTracker.find(c) == kernel->mapTracker.end() ) {

               cout<<"The child ID given does not exist or is not valid or child does not have any parent"<<"\n";
               Exit(0);
            }
            else
```

```
                if(kernel->mapTracker[c]==kernel->currentThread)
                {
                    IntStatus oldLevel = kernel->interrupt->SetLevel(IntOff);
                    cout << "Sleep thread" << endl;
                    kernel->currentThread->Sleep(FALSE);

                    (void) kernel->interrupt->SetLevel(oldLevel);


                    {
                    /* set previous programm counter (debugging only)*/
                    kernel->machine->WriteRegister(PrevPCReg, kernel->machine->ReadRegister(PCReg));
                    /* set programm counter to next instruction (all Instructions are 4 byte wide)*/
                    kernel->machine->WriteRegister(PCReg, kernel->machine->ReadRegister(PCReg) + 4);
                    /* set next programm counter for brach execution */
                    kernel->machine->WriteRegister(NextPCReg, kernel->machine->ReadRegister(PCReg)+4);
                    }
                }


          return;
          }
       break;
```

```
void Exit_POS(int val){
   IntStatus oldLevel = kernel->interrupt->SetLevel(IntOff);
   kernel->scheduler->ReadyToRun(kernel->mapTracker[val]);
   kernel->currentThread->Finish();
   (void) kernel->interrupt->SetLevel(oldLevel);
}
```

```
//Exit_POS() is called by a child and wakes up its parent.
       //This function is not a system call but you need to implement in exception.cc

       //exit pos: child finishes and wakes up the parent

       case SC_Exit:
```

```
    {
        cerr<<"The program has ended and is about to exit"<<"\n";
        int exitVal = kernel->machine->ReadRegister(4);
        cout<<"exiting value:"<<exitVal<<"\n";
        Exit(exitVal);
    }
    break;
```

Thread.h and thread.cc

```
int getThreadId()
  {
    return threadID;
  }
```

```
Thread::Thread(char* threadName)
{
    name = threadName;
    stackTop = NULL;
    stack = NULL;
    status = JUST_CREATED;

    for (int i = 0; i < MachineStateSize; i++) {
    machineState[i] = NULL;     // not strictly necessary, since
                // new thread ignores contents
                // of machine registers
    }
    tid++;
    threadID = tid;
    space = NULL;
    cout<<"Forking thread "<<name<<" at constructor now with ID:"<<getThreadId()<<endl;
}
```

Task2: **Multitasking (running multiple user-level programs)**

**&**

Task3: **Memory Allocation (Address Space for user programs)**

The main goal of this task is for us to run will load multiple user programs into its main memory and start executing them as threads with round-robin scheduling. In the terminal on typing the command with the '-x' flag, the 'main' function of Nachos calls the RunUserProg() function with the string following '-x' as parameter. We also implement the write system call. What the write system call does is reads the string from the user program and writes it to a destination which here is the console/screen.It take 3 arguments - a char* buffer, an int size, and an OpenFileId. We obtain the first two arguments from user program and print each character at a time.

Code:

Write function:

```
//•  Write()writes string to a specified destination. We're faking it using printf,
      //therefore it always writes to the screen. This system call takes three arguments:
      //a char* buffer, an int size, and an OpenFileId. Since we're faking it, we can ignore the
      //OpenFileId parameter. You need to obtain the first two arguments from the user program and
      //print an character at a time using printf.
      case SC_Write:
      {
         int o;
         cout<<"In write"<<endl;
         IntStatus oldLevel = kernel->interrupt->SetLevel(IntOff);
         int loc = (int)kernel->machine->ReadRegister(4);
         int bytesToread  = (int)kernel->machine->ReadRegister(5);
         o = loc;
         cout<<"\n";

         for(int i=0; i<bytesToread; i++)
         {
            if(kernel->machine->ReadMem(loc, 1, &o))
               printf("%c", o);
               loc++;
         }
```

```
        cout << endl;
        {
            /* set previous programm counter (debugging only)*/
            kernel->machine->WriteRegister(PrevPCReg, kernel->machine->ReadRegister(PCReg));


            /* set programm counter to next instruction (all Instructions are 4 byte wide)*/
            kernel->machine->WriteRegister(PCReg, kernel->machine->ReadRegister(PCReg) + 4);

            /* set next programm counter for brach execution */
            kernel->machine->WriteRegister(NextPCReg, kernel->machine->ReadRegister(PCReg)+4);
        }
        (void) kernel->interrupt->SetLevel(oldLevel);
        return;
        ASSERTNOTREACHED();
      }
      break;
```

```
case PageFaultException:
    {
      IntStatus oldLevel = kernel->interrupt->SetLevel(IntOff);
      pageFaulthandler((int)kernel->machine->ReadRegister(39));
      kernel->machine->WriteRegister(PrevPCReg, kernel->machine->ReadRegister(PCReg));

      /* set programm counter to next instruction (all Instructions are 4 byte wide)*/
      kernel->machine->WriteRegister(PCReg, kernel->machine->ReadRegister(PCReg) + 4);

      /* set next programm counter for brach execution */
      kernel->machine->WriteRegister(NextPCReg, kernel->machine->ReadRegister(PCReg) + 4);
      (void)kernel->interrupt->SetLevel(oldLevel);
      return;
    }
      break;
```

```
void pageFaulthandler(int va){
```

```
    int virtualp = (unsigned) va / PageSize;
    int physical = kernel->bm->FindAndSet();
    char *buf = new char[PageSize];


    TranslationEntry *te = kernel->currentThread->space->getTVPNentry(virtualp);
    if(physical == -1){
        int first = kernel->ppn->RemoveFront();
        Thread *t = kernel->ptmap[first];
        TranslationEntry *te2 = t->space->getTPPNentry(first);
        int s = te2->swapLoc;
        kernel->sF->WriteAt(&kernel->machine->mainMemory[first * PageSize],
                    PageSize, s * PageSize);
        te2->physicalPage = -1;
        te2->valid = FALSE;
        te2->use = FALSE;
        kernel->bm->Clear(first);
        physical = kernel->bm->FindAndSet();
    }

    kernel->sF->ReadAt(buf, PageSize, te->swapLoc * PageSize);
    te->valid = TRUE;
    te->use = TRUE;
    te->physicalPage = physical;
    bcopy(buf, &(kernel->machine->mainMemory[te->physicalPage * PageSize]), PageSize);
    kernel->ptmap.insert(std::pair<int, Thread *> (kernel->currentThread->space->pageTable[virtualp].physicalPage,
                                    kernel->currentThread));
        kernel->ppn->Append(te->physicalPage);
}
```

```
void
AddrSpace::buffmain(char* a, int initial_loc)
{
    int frm = 0;
    for(int i=initial_loc; i<(initial_loc+PageSize); i++)
    {
```

```
        kernel->machine->mainMemory[i] = a[frm];
        frm++;
    }
}

TranslationEntry*
AddrSpace::getTVPNentry(int v)
{
    return &pageTable[v];
}

TranslationEntry*
AddrSpace::getTPPNentry(int p)
{
    for (int i = 0; i < numPages; i++)
    {
        if(p == pageTable[i].physicalPage)
            return &pageTable[i];
    }
}
```

Addrspace.cc: changes

```
TranslationEntry* getTPPNentry(int p); //for translation entry of physical page numbr
    TranslationEntry* getTVPNentry(int v); // for translation entry to virtual one
    TranslationEntry *pageTable;

    // Translate virtual address _vaddr_
    // to physical address _paddr_. _mode_
    // is 0 for Read, 1 for Write.
    ExceptionType Translate(unsigned int vaddr, unsigned int *paddr, int mode);


  private:
    //TranslationEntry *pageTable;  // Assume linear page table translation
          // for now!
    unsigned int numPages;    // Number of pages in the virtual
```

```
        // address space
    static int pnum;
    void InitRegisters();   // Initialize user-level CPU registers,
        // before jumping to user code

};

#endif // ADDRSPACE_H
```

Main.cc

```
#include <vector>
#include "list.h"
```

```
//vector: Returns a direct pointer to the memory array used internally by the vector to store its owned elements.
    int tracker=0;
    if (!v.empty()) {
       while(tracker<v.size()){
           cout<<"Currently Processing :"<<v[tracker]<<endl;
           Thread *t = new Thread("userprogram thread");
           t->Fork((VoidFunctionPtr) RunUserProg,  (void *)v[tracker]);
           tracker++;
       }
      //RunUserProg(userProgName);
    }
```

Kernel.h and  kernel.cc

```
int getswaps(){
      return countS++;
     }
    std::map<int,Thread*> mapTracker;
    static Bitmap *bm;
    std::map<int, Thread *> ptmap;

    int hostName;               // machine identifier
```

```
    List<int> *ppn = new List<int>();
    OpenFile *sF;



  private:
    bool randomSlice;   // enable pseudo-random time slicing
    bool debugUserProg;         // single step user program
    double reliability;         // likelihood messages are dropped
    char *consoleIn;            // file to read console input from
    char *consoleOut;
    static int countS;
    int quantum;
#ifndef FILESYS_STUB
    bool formatFlag;          // format the disk if this is true
#endif
};
#endif // KERNEL_H
```

```
Kernel::Initialize()
{
    // We didn't explicitly allocate the current thread we are running in.
    // But if it ever tries to give up the CPU, we better have a Thread
    // object to save its state.
    currentThread = new Thread("main");
    currentThread->setStatus(RUNNING);

    stats = new Statistics();       // collect statistics
    interrupt = new Interrupt;      // start up interrupt handling
    scheduler = new Scheduler();    // initialize the ready queue
    alarm = new Alarm(randomSlice, quantum);    // start up time slicing
    machine = new Machine(debugUserProg);
    synchConsoleIn = new SynchConsoleInput(consoleIn); // input from stdin
    synchConsoleOut = new SynchConsoleOutput(consoleOut); // output to stdout
    synchDisk = new SynchDisk();
#ifdef FILESYS_STUB
    fileSystem = new FileSystem();
```

```
    bool checked = fileSystem->Create("sF");
    if(checked)
        sF = fileSystem->Open("sF");
#else
    fileSystem = new FileSystem(formatFlag);
#endif // FILESYS_STUB
    postOfficeIn = new PostOfficeInput(10);
    postOfficeOut = new PostOfficeOutput(reliability);

    interrupt->Enable();
}
```

```
randomSlice = FALSE;
    debugUserProg = FALSE;
    consoleIn = NULL;          // default is stdin
    consoleOut = NULL;         // default is stdout
        quantum = 3500;
```

```
else if(strcmp(argv[i],"-quantum") == 0){
            ASSERT(i + 1 < argc);
            quantum = atoi(argv[i + 1]);
            i++;
        }
```

Makefile

```
mprog1.o: mprog1.c
    $(CC) $(CFLAGS) -c mprog1.c
mprog1: mprog1.o start.o
    $(LD) $(LDFLAGS) start.o mprog1.o -o mprog1.coff
    $(COFF2NOFF) mprog1.coff mprog1

mprog2.o: mprog2.c
    $(CC) $(CFLAGS) -c mprog2.c
mprog2: mprog2.o start.o
    $(LD) $(LDFLAGS) start.o mprog2.o -o mprog2.coff
```

```
	$(COFF2NOFF) mprog2.coff mprog2

prog1.o: prog1.c
	$(CC) $(CFLAGS) -c prog1.c
prog1: prog1.o start.o
	$(LD) $(LDFLAGS) start.o prog1.o -o prog1.coff
	$(COFF2NOFF) prog1.coff prog1

prog2.o: prog2.c
	$(CC) $(CFLAGS) -c prog2.c
prog2: prog2.o start.o
	$(LD) $(LDFLAGS) start.o prog2.o -o prog2.coff
	$(COFF2NOFF) prog2.coff prog2

prog3.o: prog3.c
	$(CC) $(CFLAGS) -c prog3.c
prog3: prog3.o start.o
	$(LD) $(LDFLAGS) start.o prog3.o -o prog3.coff
	$(COFF2NOFF) prog3.coff prog3
```

**Start.s**

```
Wait_POS:
        addiu $2,$0,SC_WAIT_POS
        syscall
        j       $31
        .end Wait_POS

        .globl Fork_POS
        .ent    Fork_POS
Fork_POS:
        addiu $2,$0,SC_FORK_POS
        syscall
        j       $31
        .end Fork_POS
```

Task4: **Round-robin scheduling**

Here we make the appropriate changes to implement the RR scheduling giving each process a fixed quantum time to run.

Code:

Timer.h

```
int quantum;
```

timer.cc

```
Timer::Timer(bool doRandom, CallBackObj *toCall,int quantum)
{
    randomize = doRandom;
    callPeriodically = toCall;
    disable = FALSE;
    this->quantum = quantum;
    SetInterrupt();
}
```

```
void
Timer::SetInterrupt()
{
    if (!disable) {
        int delay = quantum;

        if (randomize) {
          delay = 1 + (RandomNumber() % (quantum * 2));
         }
        // schedule the next timer device interrupt
        kernel->interrupt->Schedule(this, delay, TimerInt);
    }
}
```

alarm.h

```
class Alarm : public CallBackObj {
  public:
```

```
    Alarm(bool doRandomYield,int quantum);
```

Alarm.cc

```
Alarm::Alarm(bool doRandom,int quantum)
{
    timer = new Timer(doRandom, this,quantum);
}
```

Kernel.h

```
private:
    bool randomSlice;       // enable pseudo-random time slicing
    bool debugUserProg;         // single step user program
    double reliability;         // likelihood messages are dropped
    char *consoleIn;            // file to read console input from
    char *consoleOut;
    int  quantum;
#ifndef FILESYS_STUB
    bool formatFlag;          // format the disk if this is true
#endif
```

Kernel.cc

```
void
Kernel::Initialize()
{
    // We didn't explicitly allocate the current thread we are running in.
    // But if it ever tries to give up the CPU, we better have a Thread
    // object to save its state.
    currentThread = new Thread("main");
    currentThread->setStatus(RUNNING);

    stats = new Statistics();       // collect statistics
    interrupt = new Interrupt;      // start up interrupt handling
    scheduler = new Scheduler();    // initialize the ready queue
```

```
    cout << "Quantum value: " << quantum << endl;
    alarm = new Alarm(randomSlice,quantum); // start up time slicing
    machine = new Machine(debugUserProg);
    synchConsoleIn = new SynchConsoleInput(consoleIn); // input from stdin
    synchConsoleOut = new SynchConsoleOutput(consoleOut); // output to stdout
    synchDisk = new SynchDisk();    //
#ifdef FILESYS_STUB
    fileSystem = new FileSystem();
#else
    fileSystem = new FileSystem(formatFlag);
#endif // FILESYS_STUB
    postOfficeIn = new PostOfficeInput(10);
    postOfficeOut = new PostOfficeOutput(reliability);

    interrupt->Enable();
}
```

```
Kernel::Kernel(int argc, char **argv)
{
    randomSlice = FALSE;
    debugUserProg = FALSE;
    consoleIn = NULL;          // default is stdin
    consoleOut = NULL;
    quantum = 100;
#ifndef FILESYS_STUB
    formatFlag = FALSE;
#endif
    reliability = 1;            // network reliability, default is 1.0
    hostName = 0;               // machine id, also UNIX socket name
                                // 0 is the default machine id
    for (int i = 1; i < argc; i++) {
        if (strcmp(argv[i], "-rs") == 0) {
        ASSERT(i + 1 < argc);
        RandomInit(atoi(argv[i + 1]));// initialize pseudo-random
                    // number generator
```

```
        randomSlice = TRUE;
        i++;
        } else if (strcmp(argv[i], "-s") == 0) {
            debugUserProg = TRUE;
    } else if (strcmp(argv[i], "-ci") == 0) {
        ASSERT(i + 1 < argc);
        consoleIn = argv[i + 1];
        i++;
    } else if(strcmp(argv[i], "-quantum") == 0){
        ASSERT(i + 1 < argc);
        quantum = atoi(argv[i + 1]);
        i++;
    } else if (strcmp(argv[i], "-co") == 0) {
        ASSERT(i + 1 < argc);
        consoleOut = argv[i + 1];
        i++;
#ifndef FILESYS_STUB
    } else if (strcmp(argv[i], "-f") == 0) {
        formatFlag = TRUE;
#endif
        } else if (strcmp(argv[i], "-n") == 0) {
            ASSERT(i + 1 < argc);   // next argument is float
            reliability = atof(argv[i + 1]);
            i++;
        } else if (strcmp(argv[i], "-m") == 0) {
            ASSERT(i + 1 < argc);   // next argument is int
            hostName = atoi(argv[i + 1]);
            i++;
        } else if (strcmp(argv[i], "-u") == 0) {
            cout << "Partial usage: nachos [-rs randomSeed]\n";
        cout << "Partial usage: nachos [-s]\n";
            cout << "Partial usage: nachos [-ci consoleIn] [-co consoleOut]\n";
#ifndef FILESYS_STUB
        cout << "Partial usage: nachos [-nf]\n";
#endif
            cout << "Partial usage: nachos [-n #] [-m #]\n";
    }
```

```
    }
}
```

**How to test your solution**

Step 1: Login with your server link and password. Start filezilla using the same credentials.
Step 2: Go into the ‘code’ folder of the ‘nachos’ folder. Access ‘build.linux’ folder.
Step 3: Run the following commands:

```
make distclean
make depend
make nachos
```

Step 4: Make sure the updated files are uploaded using filezilla
Step 5: Run the command: ./nachos -x ../test/”userprogramname”
While running multiple programs
./nachos -x ../test/”userprogramname1” -x ../test/”userprogramname2”

⇒ **Assumption: while running prog2 we must provide quantum that is above 2000 while running it**

**Files modified / Added: Give Fully qualified path of the files Added:**

Modified:

1) syscall.h
2) exception.cc
3) start.s
4) thread.cc
5) thread.h

6) kernel.h
7) kernel.cc
8) timer.h
9) timer.cc
10) alarm.h
11) alarm.cc
12) main.cc
13) list.cc
14) addrspace.h
15) addrspace.cc
16) makefile

Added:

1) prog1.c
2) prog2.c
3) prog3.c
4) mprog1.c
5) mprog2.c
6) mprog3.c

# Outputs:

```
                                                                                ^
../network/post.cc: In constructor 'PostOfficeInput::PostOfficeInput(int)':
../network/post.cc:155:60: warning: deprecated conversion from string constant to 'char*' [-Wwrite-strings]
     messageAvailable = new Semaphore("message available", 0);
                                                            ^
../network/post.cc:162:43: warning: deprecated conversion from string constant to 'char*' [-Wwrite-strings]
     Thread *t = new Thread("postal worker");
                                           ^
../network/post.cc: In constructor 'PostOfficeOutput::PostOfficeOutput(double)':
../network/post.cc:268:50: warning: deprecated conversion from string constant to 'char*' [-Wwrite-strings]
     messageSent = new Semaphore("message sent", 0);
                                                 ^
../network/post.cc:269:44: warning: deprecated conversion from string constant to 'char*' [-Wwrite-strings]
     sendLock = new Lock("message send lock");
                                            ^
In file included from ../threads/synchlist.h:49:0,
                 from ../network/post.h:34,
                 from ../network/post.cc:20:
../threads/synchlist.cc: In instantiation of 'SynchList<T>::SynchList() [with T = Mail*]':
../network/post.cc:52:38:   required from here
../threads/synchlist.cc:26:10: warning: deprecated conversion from string constant to 'char*' [-Wwrite-strings]
     lock = new Lock("list lock");
          ^
../threads/synchlist.cc:27:15: warning: deprecated conversion from string constant to 'char*' [-Wwrite-strings]
     listEmpty = new Condition("list empty cond");
               ^
g++ -m32 -P -I../network -I../filesys -I../userprog -I../threads -I../machine -I../lib -I- -Dx86 -DLINUX -c ../threads/switch.S
cc1: note: obsolete option -I- used, please use -iquote instead
g++ bitmap.o debug.o libtest.o sysdep.o interrupt.o stats.o timer.o console.o machine.o mipssim.o translate.o network.o disk.o alarm.o kernel.o main.o scheduler.o synch.o thread.o t
hreadtest.o addrspace.o exception.o synchconsole.o directory.o filehdr.o filesys.o pbitmap.o openfile.o synchdisk.o post.o switch.o -m32 -o nachos
nsuryapr@lcs-vc-cis486:~/nachos_pa2_fin/nachos/code/build.linux$ ./nachos -x ../test/prog1
Forking thread main at constructor now with ID:1
Forking thread postal worker at constructor now with ID:2
Currently Processing :../test/prog1
Forking thread userprogram thread at constructor now with ID:3
Forking thread forked thread at constructor now with ID:4
Sleep thread
ForkTest1 is called, its PID is 4
ForkTest1 is in loop 0
ForkTest1 is in loop 1
ForkTest1 is in loop 2
Forking thread forked thread at constructor now with ID:5
Sleep thread
ForkTest2 is called, its PID is 5
ForkTest2 is in loop 0
ForkTest2 is in loop 1
ForkTest2 is in loop 2
Forking thread forked thread at constructor now with ID:6
Sleep thread
ForkTest3 is called, its PID is 6
ForkTest3 is in loop 0
ForkTest3 is in loop 1
ForkTest3 is in loop 2
The program has ended and is about to exit
exiting value:0
nsuryapr@lcs-vc-cis486:~/nachos_pa2_fin/nachos/code/build.linux$
```

```
../threads/synchlist.cc:26:10: warning: deprecated conversion from string constant to 'char*' [-Wwrite-strings]
     lock = new Lock("list lock");
          ^
../threads/synchlist.cc:27:15: warning: deprecated conversion from string constant to 'char*' [-Wwrite-strings]
     listEmpty = new Condition("list empty cond");
               ^
g++ -m32 -P -I../network -I../filesys -I../userprog -I../threads -I../machine -I../lib -I- -Dx86 -DLINUX -c ../threads/switch.S
cc1: note: obsolete option '-I-' used, please use '-iquote' instead
g++ bitmap.o debug.o libtest.o sysdep.o interrupt.o stats.o timer.o console.o machine.o mipssim.o translate.o network.o disk.o alarm.o kernel.o main.o scheduler.o synch.o thread.o t
hreadtest.o addrspace.o exception.o synchconsole.o directory.o filehdr.o filesys.o pbitmap.o openfile.o synchdisk.o post.o switch.o -m32 -o nachos
nsuryapr@lcs-vc-cis486:~/nachos_pa2_fin/nachos/code/build.linux$ ./nachos -x ../test/prog1
Forking thread main at constructor now with ID:1
Forking thread postal worker at constructor now with ID:2
Currently Processing :../test/prog1
Forking thread userprogram thread at constructor now with ID:3
Forking thread forked thread at constructor now with ID:4
Sleep thread
ForkTest1 is called, its PID is 4
ForkTest1 is in loop 0
ForkTest1 is in loop 1
ForkTest1 is in loop 2
Forking thread forked thread at constructor now with ID:5
Sleep thread
ForkTest2 is called, its PID is 5
ForkTest2 is in loop 0
ForkTest2 is in loop 1
ForkTest2 is in loop 2
Forking thread forked thread at constructor now with ID:6
Sleep thread
ForkTest3 is called, its PID is 6
ForkTest3 is in loop 0
ForkTest3 is in loop 1
ForkTest3 is in loop 2
The program has ended and is about to exit
exiting value:0
nsuryapr@lcs-vc-cis486:~/nachos_pa2_fin/nachos/code/build.linux$ ./nachos -x ../test/prog2 -quantum 2000
Quantum is: 2000
Forking thread main at constructor now with ID:1
Forking thread postal worker at constructor now with ID:2
Currently Processing :../test/prog2
Forking thread userprogram thread at constructor now with ID:3
Forking thread forked thread at constructor now with ID:4
Forking thread forked thread at constructor now with ID:5
Sleep thread
ForkTest1 is called, its PID is 4
ForkTest1 is in loop 0
ForkTest1 is in loop 1
ForkTest2 is called, its PID is 5
ForkTest2 is in loop 0
ForkTest2 is in loop 1
ForkTest1 is in loop 2
ForkTest2 is in loop 2
Sleep thread
The program has ended and is about to exit
exiting value:0
nsuryapr@lcs-vc-cis486:~/nachos_pa2_fin/nachos/code/build.linux$
```

---

```
Forking thread main at constructor now with ID:1
Forking thread postal worker at constructor now with ID:2
Currently Processing :../test/prog1
Forking thread userprogram thread at constructor now with ID:3
Forking thread forked thread at constructor now with ID:4
Sleep thread
ForkTest1 is called, its PID is 4
ForkTest1 is in loop 0
ForkTest1 is in loop 1
ForkTest1 is in loop 2
Forking thread forked thread at constructor now with ID:5
Sleep thread
ForkTest2 is called, its PID is 5
ForkTest2 is in loop 0
ForkTest2 is in loop 1
ForkTest2 is in loop 2
Forking thread forked thread at constructor now with ID:6
Sleep thread
ForkTest3 is called, its PID is 6
ForkTest3 is in loop 0
ForkTest3 is in loop 1
ForkTest3 is in loop 2
The program has ended and is about to exit
exiting value:0
nsuryapr@lcs-vc-cis486:~/nachos_pa2_fin/nachos/code/build.linux$ ./nachos -x ../test/prog2 -quantum 2000
Quantum is: 2000
Forking thread main at constructor now with ID:1
Forking thread postal worker at constructor now with ID:2
Currently Processing :../test/prog2
Forking thread userprogram thread at constructor now with ID:3
Forking thread forked thread at constructor now with ID:4
Forking thread forked thread at constructor now with ID:5
Sleep thread
ForkTest1 is called, its PID is 4
ForkTest1 is in loop 0
ForkTest1 is in loop 1
ForkTest2 is called, its PID is 5
ForkTest2 is in loop 0
ForkTest2 is in loop 1
ForkTest1 is in loop 2
ForkTest2 is in loop 2
Sleep thread
The program has ended and is about to exit
exiting value:0
nsuryapr@lcs-vc-cis486:~/nachos_pa2_fin/nachos/code/build.linux$ ./nachos -x ../test/prog3 -quantum 2000
Quantum is: 2000
Forking thread main at constructor now with ID:1
Forking thread postal worker at constructor now with ID:2
Currently Processing :../test/prog3
Forking thread userprogram thread at constructor now with ID:3
Forking thread forked thread at constructor now with ID:4
Forking thread forked thread at constructor now with ID:5
Forking thread forked thread at constructor now with ID:6
The program has ended and is about to exit
exiting value:0
nsuryapr@lcs-vc-cis486:~/nachos_pa2_fin/nachos/code/build.linux$
```

```
The program has ended and is about to exit
exiting value:0
nsuryapr@lcs-vc-cis486:~/nachos_pa2_fin/nachos/code/build.linux$ ./nachos -x ../test/prog2 -quantum 2000
Quantum is: 2000
Forking thread main at constructor now with ID:1
Forking thread postal worker at constructor now with ID:2
Currently Processing :../test/prog2
Forking thread userprogram thread at constructor now with ID:3
Forking thread forked thread at constructor now with ID:4
Forking thread forked thread at constructor now with ID:5
Sleep thread
ForkTest1 is called, its PID is 4
ForkTest1 is in loop 0
ForkTest1 is in loop 1
ForkTest2 is called, its PID is 5
ForkTest2 is in loop 0
ForkTest2 is in loop 1
ForkTest1 is in loop 2
ForkTest2 is in loop 2
Sleep thread
The program has ended and is about to exit
exiting value:0
nsuryapr@lcs-vc-cis486:~/nachos_pa2_fin/nachos/code/build.linux$ ./nachos -x ../test/prog3 -quantum 2000
Quantum is: 2000
Forking thread main at constructor now with ID:1
Forking thread postal worker at constructor now with ID:2
Currently Processing :../test/prog3
Forking thread userprogram thread at constructor now with ID:3
Forking thread forked thread at constructor now with ID:4
Forking thread forked thread at constructor now with ID:5
Forking thread forked thread at constructor now with ID:6
The program has ended and is about to exit
exiting value:0
nsuryapr@lcs-vc-cis486:~/nachos_pa2_fin/nachos/code/build.linux$ ./nachos -x ../test/mprog1
Forking thread main at constructor now with ID:1
Forking thread postal worker at constructor now with ID:2
Currently Processing :../test/mprog1
Forking thread userprogram thread at constructor now with ID:3
In write

current user program: prog1
In write

current user program: prog1
In write

current user program: prog1
In write

current user program: prog1
In write

current user program: prog1
The program has ended and is about to exit
exiting value:0
nsuryapr@lcs-vc-cis486:~/nachos_pa2_fin/nachos/code/build.linux$
```

---

```
nsuryapr@lcs-vc-cis486:~/nachos_pa2_fin/nachos/code/build.linux$ ./nachos -x ../test/prog3 -quantum 2000
Quantum is: 2000
Forking thread main at constructor now with ID:1
Forking thread postal worker at constructor now with ID:2
Currently Processing :../test/prog3
Forking thread userprogram thread at constructor now with ID:3
Forking thread forked thread at constructor now with ID:4
Forking thread forked thread at constructor now with ID:5
Forking thread forked thread at constructor now with ID:6
The program has ended and is about to exit
exiting value:0
nsuryapr@lcs-vc-cis486:~/nachos_pa2_fin/nachos/code/build.linux$ ./nachos -x ../test/mprog1
Forking thread main at constructor now with ID:1
Forking thread postal worker at constructor now with ID:2
Currently Processing :../test/mprog1
Forking thread userprogram thread at constructor now with ID:3
In write

current user program: prog1
In write

current user program: prog1
In write

current user program: prog1
In write

current user program: prog1
In write

current user program: prog1
The program has ended and is about to exit
exiting value:0
nsuryapr@lcs-vc-cis486:~/nachos_pa2_fin/nachos/code/build.linux$ ./nachos -x ../test/mprog2
Forking thread main at constructor now with ID:1
Forking thread postal worker at constructor now with ID:2
Currently Processing :../test/mprog2
Forking thread userprogram thread at constructor now with ID:3
In write

current user program: prog2
In write

current user program: prog2
In write

current user program: prog2
In write

current user program: prog2
In write

current user program: prog2
The program has ended and is about to exit
exiting value:0
nsuryapr@lcs-vc-cis486:~/nachos_pa2_fin/nachos/code/build.linux$
```

```
current user program: prog2
In write

current user program: prog2
In write

current user program: prog2
In write

current user program: prog2
In write

current user program: prog2
The program has ended and is about to exit
exiting value:0
nsuryapr@lcs-vc-cis486:~/nachos_pa2_fin/nachos/code/build.linux$ ./nachos -x ../test/mprog1 -x ../test/mprog2
Forking thread main at constructor now with ID:1
Forking thread postal worker at constructor now with ID:2
Currently Processing :../test/mprog1
Forking thread userprogram thread at constructor now with ID:3
Currently Processing :../test/mprog2
Forking thread userprogram thread at constructor now with ID:4
In write

current user program: prog1
In write

current user program: prog2
In write

current user program: prog1
In write

current user program: prog2
In write

current user program: prog1
In write

current user program: prog2
In write

current user program: prog2
In write

current user program: prog1
In write

current user program: prog1
In write

current user program: prog2
The program has ended and is about to exit
exiting value:0
nsuryapr@lcs-vc-cis486:~/nachos_pa2_fin/nachos/code/build.linux$
```

---

```
current user program: prog2
The program has ended and is about to exit
exiting value:0
nsuryapr@lcs-vc-cis486:~/nachos_pa2_fin/nachos/code/build.linux$ ./nachos -x ../test/mprog1 -x ../test/mprog2
Forking thread main at constructor now with ID:1
Forking thread postal worker at constructor now with ID:2
Currently Processing :../test/mprog1
Forking thread userprogram thread at constructor now with ID:3
Currently Processing :../test/mprog2
Forking thread userprogram thread at constructor now with ID:4
In write

current user program: prog1
In write

current user program: prog2
In write

current user program: prog1
In write

current user program: prog2
In write

current user program: prog1
In write

current user program: prog2
In write

current user program: prog2
In write

current user program: prog1
In write

current user program: prog1
In write

current user program: prog2
The program has ended and is about to exit
exiting value:0
nsuryapr@lcs-vc-cis486:~/nachos_pa2_fin/nachos/code/build.linux$ ./nachos -x ../test/matmult -x ../test/mprog2
Forking thread main at constructor now with ID:1
Forking thread postal worker at constructor now with ID:2
Currently Processing :../test/matmult
Forking thread userprogram thread at constructor now with ID:3
Currently Processing :../test/mprog2
Forking thread userprogram thread at constructor now with ID:4
In write

current user program: prog2
The program has ended and is about to exit
exiting value:0
nsuryapr@lcs-vc-cis486:~/nachos_pa2_fin/nachos/code/build.linux$
```

```
Currently Processing :../test/mprog1
Forking thread userprogram thread at constructor now with ID:3
Currently Processing :../test/mprog2
Forking thread userprogram thread at constructor now with ID:4
In write

current user program: prog1
In write

current user program: prog2
In write

current user program: prog1
In write

current user program: prog2
In write

current user program: prog1
In write

current user program: prog2
In write

current user program: prog2
In write

current user program: prog1
In write

current user program: prog1
In write

current user program: prog2
The program has ended and is about to exit
exiting value:0
nsuryapr@lcs-vc-cis486:~/nachos_pa2_fin/nachos/code/build.linux$ ./nachos -x ../test/matmult -x ../test/mprog2
Forking thread main at constructor now with ID:1
Forking thread postal worker at constructor now with ID:2
Currently Processing :../test/matmult
Forking thread userprogram thread at constructor now with ID:3
Currently Processing :../test/mprog2
Forking thread userprogram thread at constructor now with ID:4
In write

current user program: prog2
The program has ended and is about to exit
exiting value:0
nsuryapr@lcs-vc-cis486:~/nachos_pa2_fin/nachos/code/build.linux$ ./nachos -x ../test/matmult
Forking thread main at constructor now with ID:1
Forking thread postal worker at constructor now with ID:2
Currently Processing :../test/matmult
Forking thread userprogram thread at constructor now with ID:3
The program has ended and is about to exit
exiting value:7220
nsuryapr@lcs-vc-cis486:~/nachos_pa2_fin/nachos/code/build.linux$
```