PA-3

Name: Raghunandan Gajanan Bhat

SUNetID: rgbhat@syr.edu

Task-1: Nachos Pre-emptive Multi-programming

Task-1.1

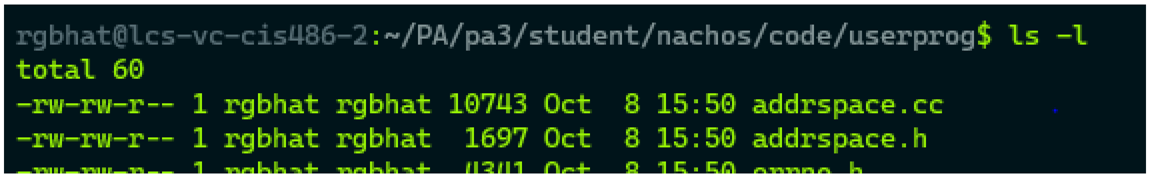
✪ **[R-1]:** The function - `void RunUserProg(void *filename)` in `threads/main.cc` creates the address space to run a user program.

✪ **[R-2]:**

1. The first argument to the `Fork()` is a pointer to a function which we have to run concurrently. It tells us which function has to be run by allocating the stack.
2. If we use a pointer to `RunUserProg()` as an argument to `Fork()`, it will work just fine. The first argument to `Fork()`, which is of type `VoidFunctionPtr` is defined as a pointer to a function which takes an arbitrary pointer argument and returns nothing. The `RunUserProgram()` exactly matches with the description of the `VoidFunctionPtr` – it takes a pointer as an argument and returns nothing. If we try to run a single user program, it will work just fine. The `main` thread will run the user program. But if there are multiple user programs, `Fork()` will not be able to run all of them. It simply runs the first user program using the `main` thread and exits.

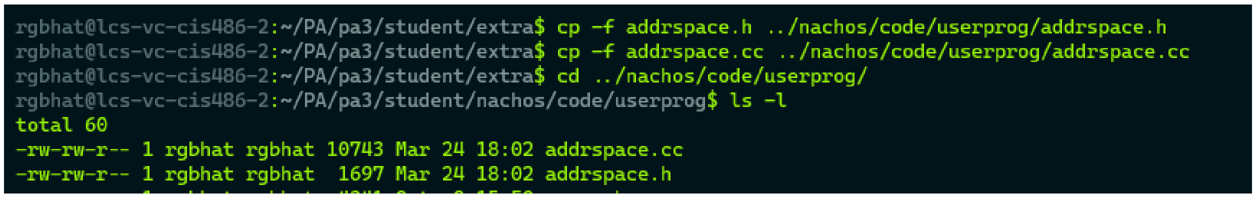
✪ **[R-3]:**

Before



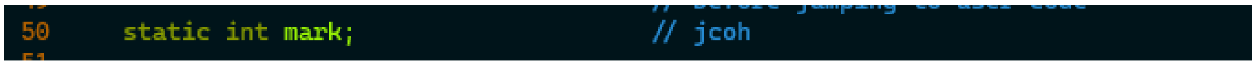
```
rgbhat@lcs-vc-cis486-2:~/PA/pa3/student/nachos/code/userprog$ ls -l
total 60
-rw-rw-r-- 1 rgbhat rgbhat 10743 Oct  8 15:50 addrspace.cc
-rw-rw-r-- 1 rgbhat rgbhat  1697 Oct  8 15:50 addrspace.h
```

After replacing the files



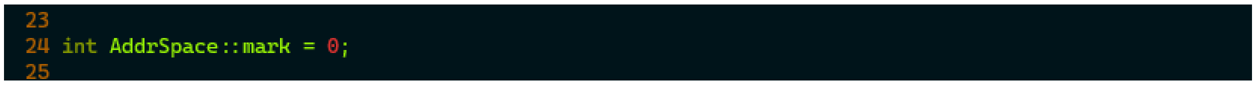
```
rgbhat@lcs-vc-cis486-2:~/PA/pa3/student/extra$ cp -f addrspace.h ../nachos/code/userprog/addrspace.h
rgbhat@lcs-vc-cis486-2:~/PA/pa3/student/extra$ cp -f addrspace.cc ../nachos/code/userprog/addrspace.cc
rgbhat@lcs-vc-cis486-2:~/PA/pa3/student/extra$ cd ../nachos/code/userprog/
rgbhat@lcs-vc-cis486-2:~/PA/pa3/student/nachos/code/userprog$ ls -l
total 60
-rw-rw-r-- 1 rgbhat rgbhat 10743 Mar 24 18:02 addrspace.cc
-rw-rw-r-- 1 rgbhat rgbhat  1697 Mar 24 18:02 addrspace.h
```

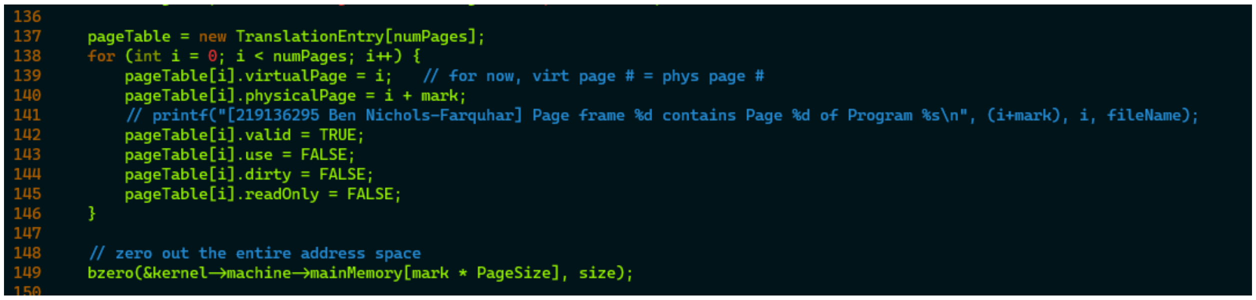
`addrspace.h`:

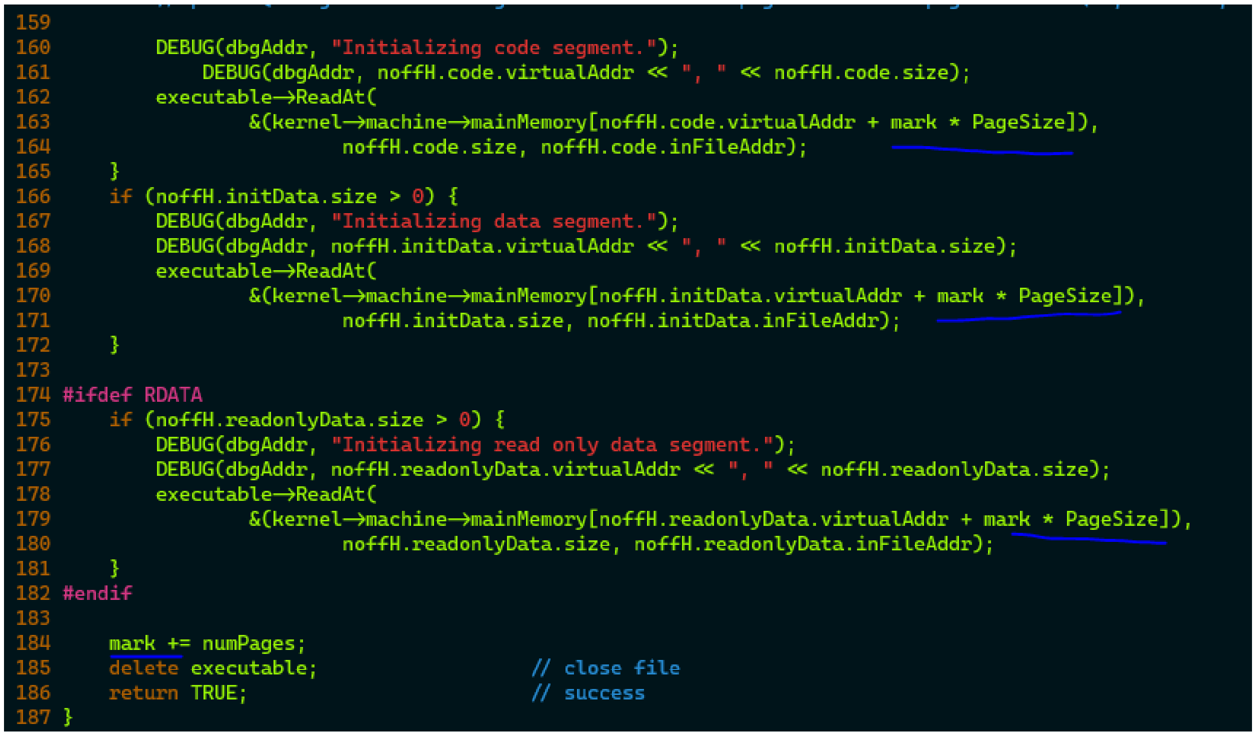


```
    static int mark;                          // jcoh
```

addrspace.cc:

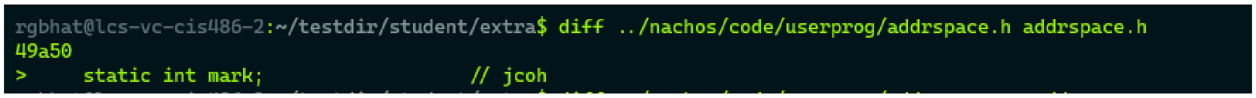

```
int AddrSpace::mark = 0;
```


```
    pageTable = new TranslationEntry[numPages];
    for (int i = 0; i < numPages; i++) {
        pageTable[i].virtualPage = i;   // for now, virt page # = phys page #
        pageTable[i].physicalPage = i + mark;
        // printf("[219136295 Ben Nichols-Farquhar] Page frame %d contains Page %d of Program %s\n", (i+mark), i, fileName);
        pageTable[i].valid = TRUE;
        pageTable[i].use = FALSE;
        pageTable[i].dirty = FALSE;
        pageTable[i].readOnly = FALSE;
    }

    // zero out the entire address space
    bzero(&kernel->machine->mainMemory[mark * PageSize], size);
```

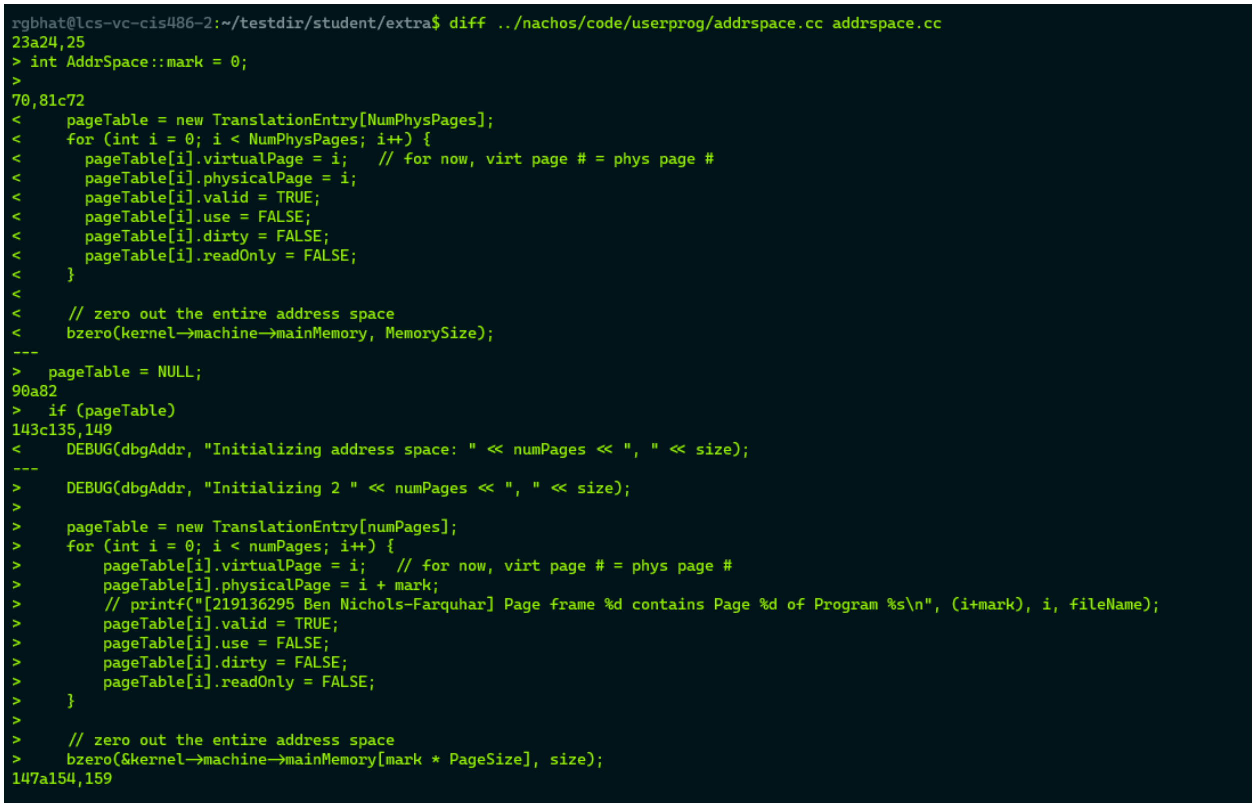

```
        DEBUG(dbgAddr, "Initializing code segment.");
            DEBUG(dbgAddr, noffH.code.virtualAddr << ", " << noffH.code.size);
        executable->ReadAt(
                &(kernel->machine->mainMemory[noffH.code.virtualAddr + mark * PageSize]),
                        noffH.code.size, noffH.code.inFileAddr);
    }
    if (noffH.initData.size > 0) {
        DEBUG(dbgAddr, "Initializing data segment.");
        DEBUG(dbgAddr, noffH.initData.virtualAddr << ", " << noffH.initData.size);
        executable->ReadAt(
                &(kernel->machine->mainMemory[noffH.initData.virtualAddr + mark * PageSize]),
                        noffH.initData.size, noffH.initData.inFileAddr);
    }

#ifdef RDATA
    if (noffH.readonlyData.size > 0) {
        DEBUG(dbgAddr, "Initializing read only data segment.");
        DEBUG(dbgAddr, noffH.readonlyData.virtualAddr << ", " << noffH.readonlyData.size);
        executable->ReadAt(
                &(kernel->machine->mainMemory[noffH.readonlyData.virtualAddr + mark * PageSize]),
                        noffH.readonlyData.size, noffH.readonlyData.inFileAddr);
    }
#endif

    mark += numPages;
    delete executable;                  // close file
    return TRUE;                        // success
}
```

Using `diff` command:

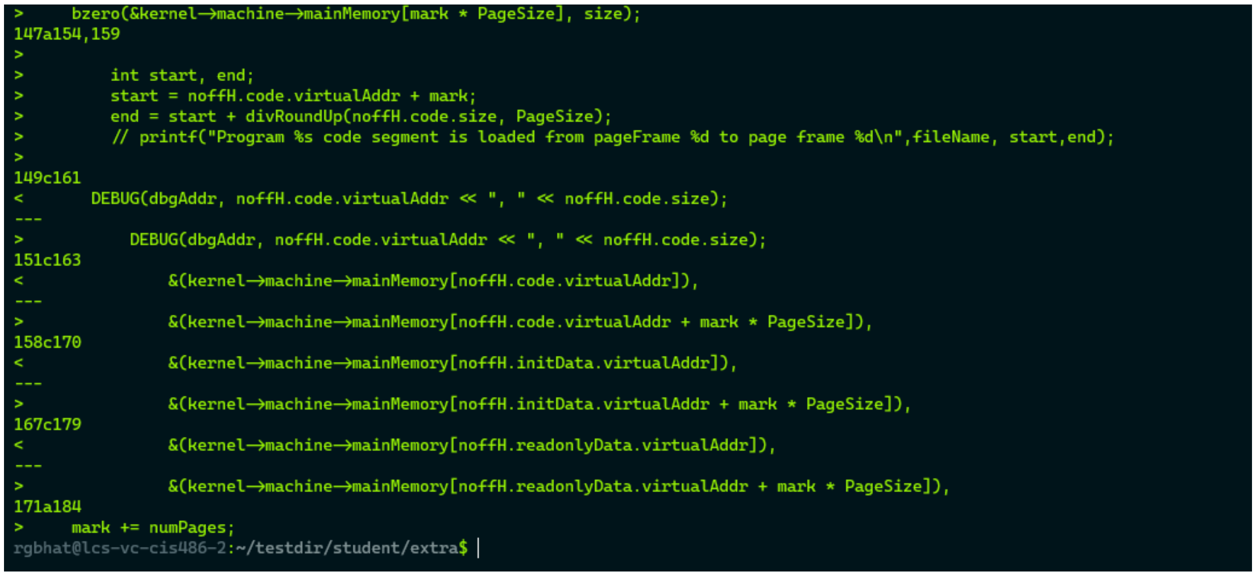
addrspace.h:


```
rgbhat@lcs-vc-cis486-2:~/testdir/student/extra$ diff ../nachos/code/userprog/addrspace.h addrspace.h
49a50
>     static int mark;                  // jcoh
```

addrspace.cc:



```
rgbhat@lcs-vc-cis486-2:~/testdir/student/extra$ diff ../nachos/code/userprog/addrspace.cc addrspace.cc
23a24,25
> int AddrSpace::mark = 0;
>
70,81c72
<     pageTable = new TranslationEntry[NumPhysPages];
<     for (int i = 0; i < NumPhysPages; i++) {
<       pageTable[i].virtualPage = i;   // for now, virt page # = phys page #
<       pageTable[i].physicalPage = i;
<       pageTable[i].valid = TRUE;
<       pageTable[i].use = FALSE;
<       pageTable[i].dirty = FALSE;
<       pageTable[i].readOnly = FALSE;
<     }
<
<     // zero out the entire address space
<     bzero(kernel->machine->mainMemory, MemorySize);
---
>   pageTable = NULL;
90a82
>   if (pageTable)
143c135,149
<     DEBUG(dbgAddr, "Initializing address space: " << numPages << ", " << size);
---
>     DEBUG(dbgAddr, "Initializing 2 " << numPages << ", " << size);
>
>     pageTable = new TranslationEntry[numPages];
>     for (int i = 0; i < numPages; i++) {
>         pageTable[i].virtualPage = i;   // for now, virt page # = phys page #
>         pageTable[i].physicalPage = i + mark;
>         // printf("[219136295 Ben Nichols-Farquhar] Page frame %d contains Page %d of Program %s\n", (i+mark), i, fileName);
>         pageTable[i].valid = TRUE;
>         pageTable[i].use = FALSE;
>         pageTable[i].dirty = FALSE;
>         pageTable[i].readOnly = FALSE;
>     }
>
>     // zero out the entire address space
>     bzero(&kernel->machine->mainMemory[mark * PageSize], size);
147a154,159
```



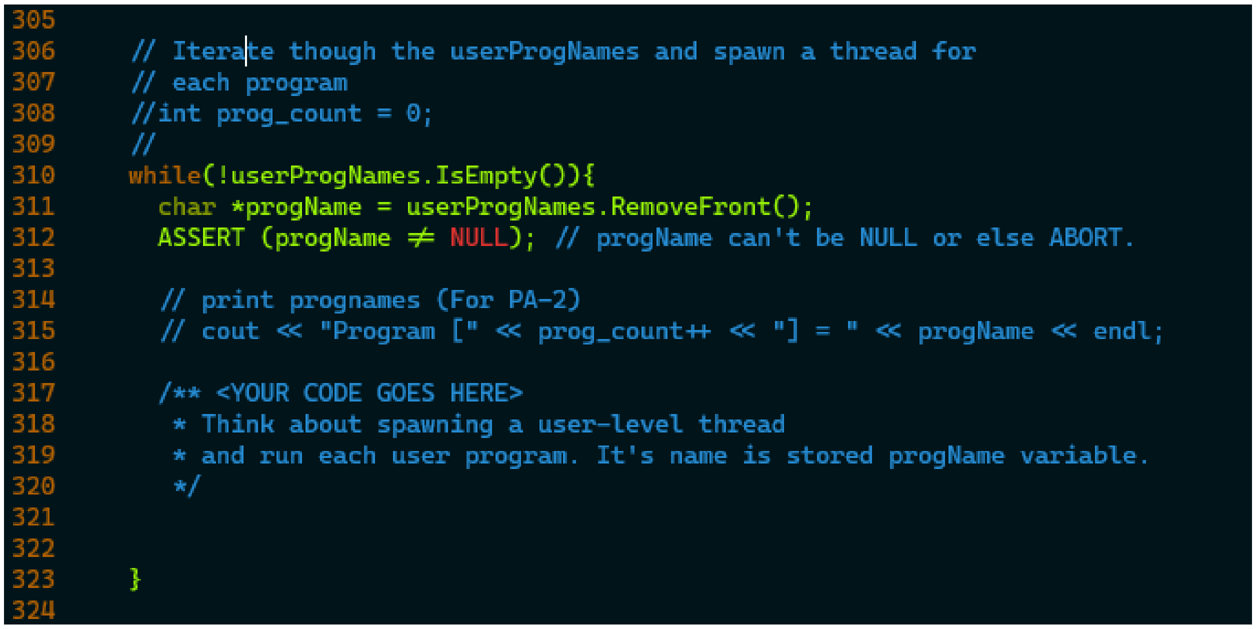
```
>     bzero(&kernel->machine->mainMemory[mark * PageSize], size);
147a154,159
```

```
>
>         int start, end;
>         start = noffH.code.virtualAddr + mark;
>         end = start + divRoundUp(noffH.code.size, PageSize);
>         // printf("Program %s code segment is loaded from pageFrame %d to page frame %d\n",fileName, start,end);
>
149c161
<       DEBUG(dbgAddr, noffH.code.virtualAddr << ", " << noffH.code.size);
---
>           DEBUG(dbgAddr, noffH.code.virtualAddr << ", " << noffH.code.size);
151c163
<               &(kernel->machine->mainMemory[noffH.code.virtualAddr]),
---
>               &(kernel->machine->mainMemory[noffH.code.virtualAddr + mark * PageSize]),
158c170
<               &(kernel->machine->mainMemory[noffH.initData.virtualAddr]),
---
>               &(kernel->machine->mainMemory[noffH.initData.virtualAddr + mark * PageSize]),
167c179
<               &(kernel->machine->mainMemory[noffH.readonlyData.virtualAddr]),
---
>               &(kernel->machine->mainMemory[noffH.readonlyData.virtualAddr + mark * PageSize]),
171a184
>     mark += numPages;
rgbhat@lcs-vc-cis486-2:~/testdir/student/extra$ |
```

✪ [R-4]:

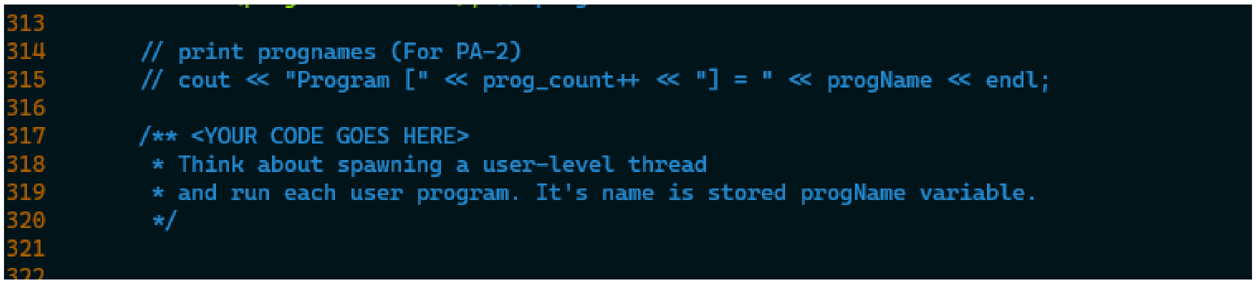
1. `List<char*> userProgNames`
2. `List<char*>` - List of character pointers

3. Iteration of `userProgNames`



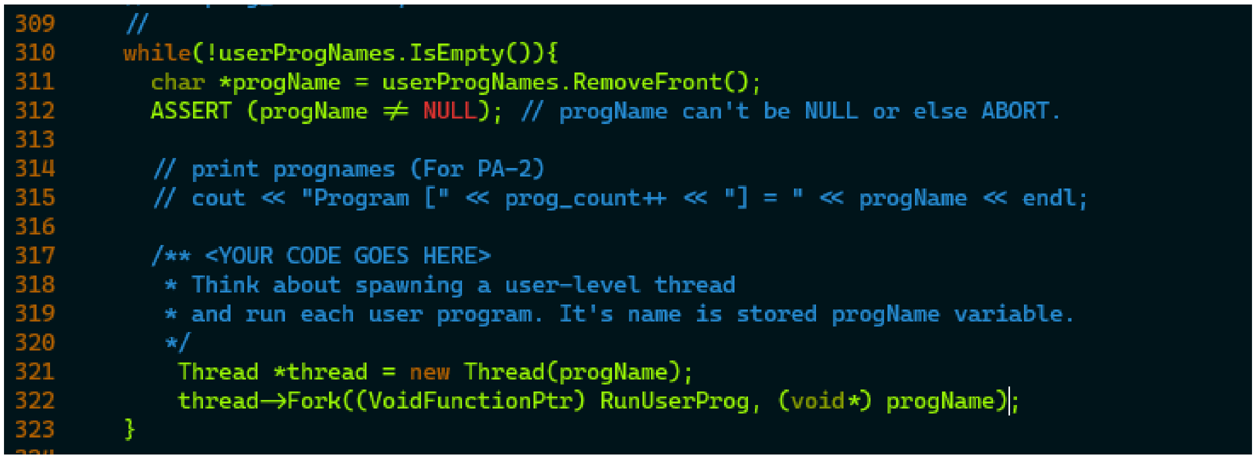
```
305
306     // Iterate though the userProgNames and spawn a thread for
307     // each program
308     //int prog_count = 0;
309     //
310     while(!userProgNames.IsEmpty()){
311       char *progName = userProgNames.RemoveFront();
312       ASSERT (progName != NULL); // progName can't be NULL or else ABORT.
313
314       // print prognames (For PA-2)
315       // cout << "Program [" << prog_count++ << "] = " << progName << endl;
316
317       /** <YOUR CODE GOES HERE>
318        * Think about spawning a user-level thread
319        * and run each user program. It's name is stored progName variable.
320        */
321
322
323     }
324
```

4. Variable name: `progName`, Type: `char*` - character pointer
5. Line 317



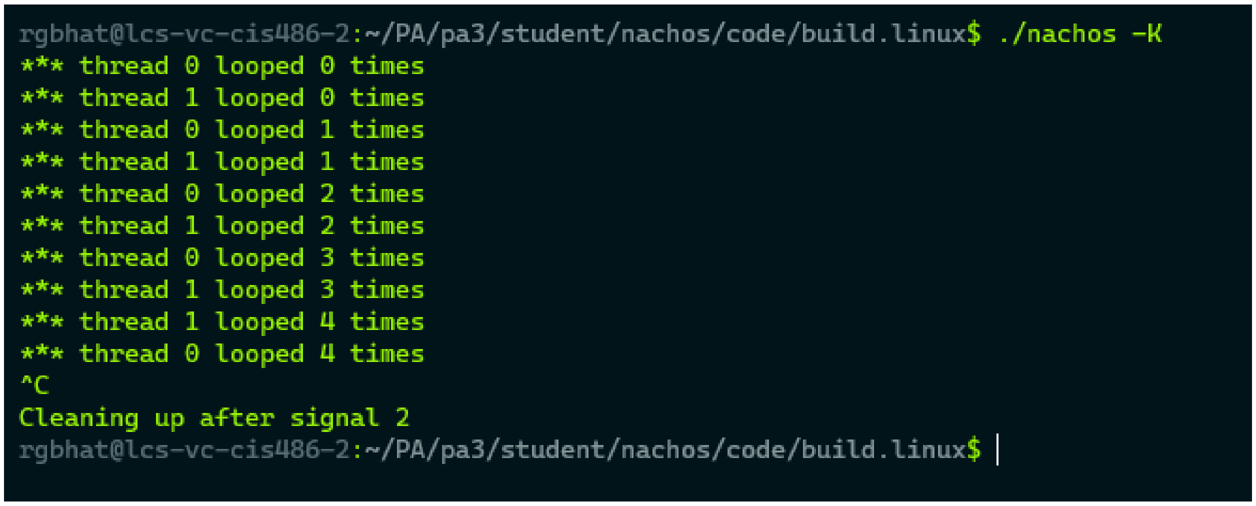
```
313
314       // print prognames (For PA-2)
315       // cout << "Program [" << prog_count++ << "] = " << progName << endl;
316
317       /** <YOUR CODE GOES HERE>
318        * Think about spawning a user-level thread
319        * and run each user program. It's name is stored progName variable.
320        */
321
```

✪ **[R-5]:** Modified `main.cc`



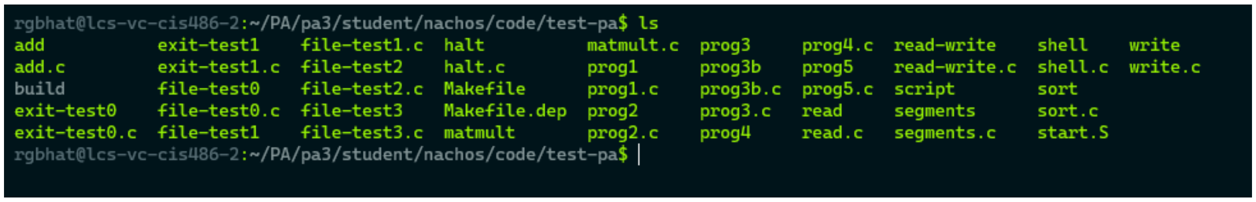
```
309     //
310     while(!userProgNames.IsEmpty()){
311       char *progName = userProgNames.RemoveFront();
312       ASSERT (progName != NULL); // progName can't be NULL or else ABORT.
313
314       // print prognames (For PA-2)
315       // cout << "Program [" << prog_count++ << "] = " << progName << endl;
316
317       /** <YOUR CODE GOES HERE>
318        * Think about spawning a user-level thread
319        * and run each user program. It's name is stored progName variable.
320        */
321         Thread *thread = new Thread(progName);
322         thread->Fork((VoidFunctionPtr) RunUserProg, (void*) progName);
323     }
```

✪ **[R-6]:** `./nachos -K`



```
rgbhat@lcs-vc-cis486-2:~/PA/pa3/student/nachos/code/build.linux$ ./nachos -K
*** thread 0 looped 0 times
*** thread 1 looped 0 times
*** thread 0 looped 1 times
*** thread 1 looped 1 times
*** thread 0 looped 2 times
*** thread 1 looped 2 times
*** thread 0 looped 3 times
*** thread 1 looped 3 times
*** thread 1 looped 4 times
*** thread 0 looped 4 times
^C
Cleaning up after signal 2
rgbhat@lcs-vc-cis486-2:~/PA/pa3/student/nachos/code/build.linux$ 
```

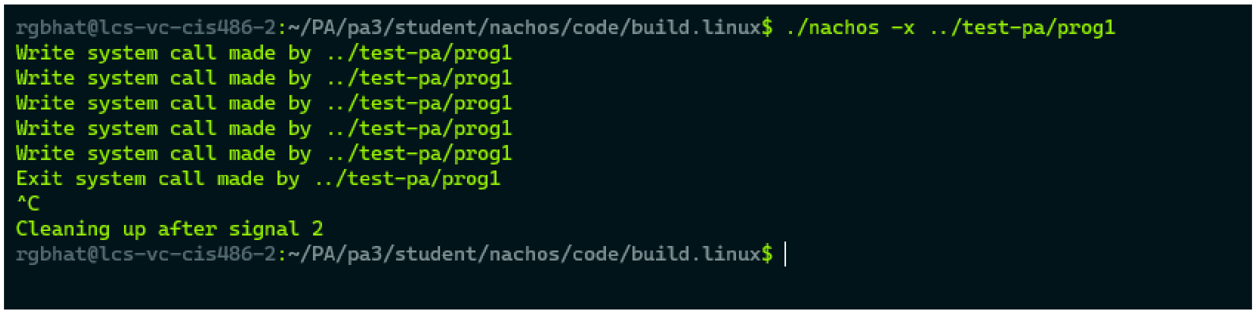
✪ **[R-7]:** Build programs in `test-pa/`



```
rgbhat@lcs-vc-cis486-2:~/PA/pa3/student/nachos/code/test-pa$ ls
add          exit-test1    file-test1.c  halt          matmult.c  prog3     prog4.c  read-write    shell    write
add.c        exit-test1.c  file-test2    halt.c        prog1      prog3b    prog5    read-write.c  shell.c  write.c
build        file-test0    file-test2.c  Makefile      prog1.c    prog3b.c  prog5.c  script        sort
exit-test0   file-test0.c  file-test3    Makefile.dep  prog2      prog3.c   read     segments      sort.c
exit-test0.c file-test1    file-test3.c  matmult       prog2.c    prog4     read.c   segments.c    start.S
rgbhat@lcs-vc-cis486-2:~/PA/pa3/student/nachos/code/test-pa$ 
```

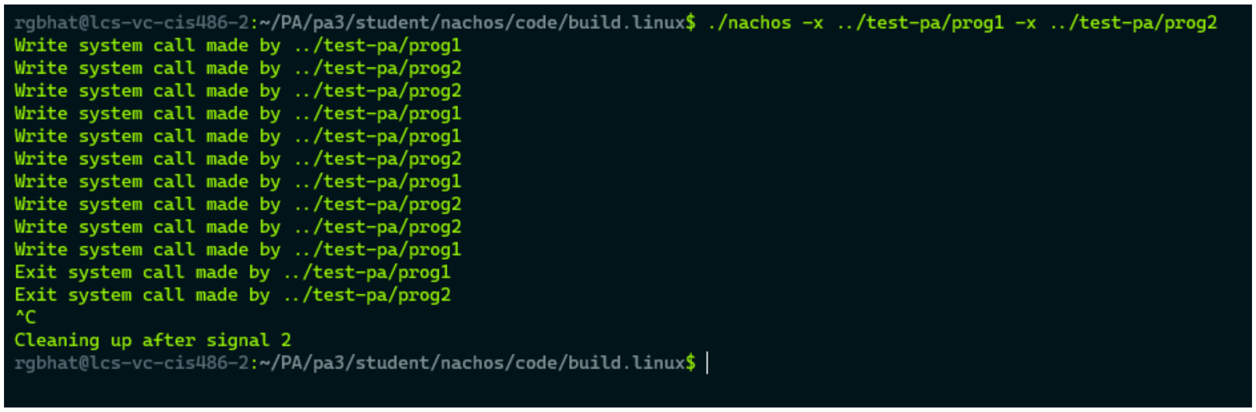
✪ **[R-8]:** In `prog1.c`, `Write()` function is called 5 times and `Exit()` is called only once. These functions are system calls – `Write()` system call writes certain number bytes from buffer to the open file and `Exit()` system call returns exit status of the user program when it is done.

✪ **[R-9]:** `./nachos -x ../test-pa/prog1`



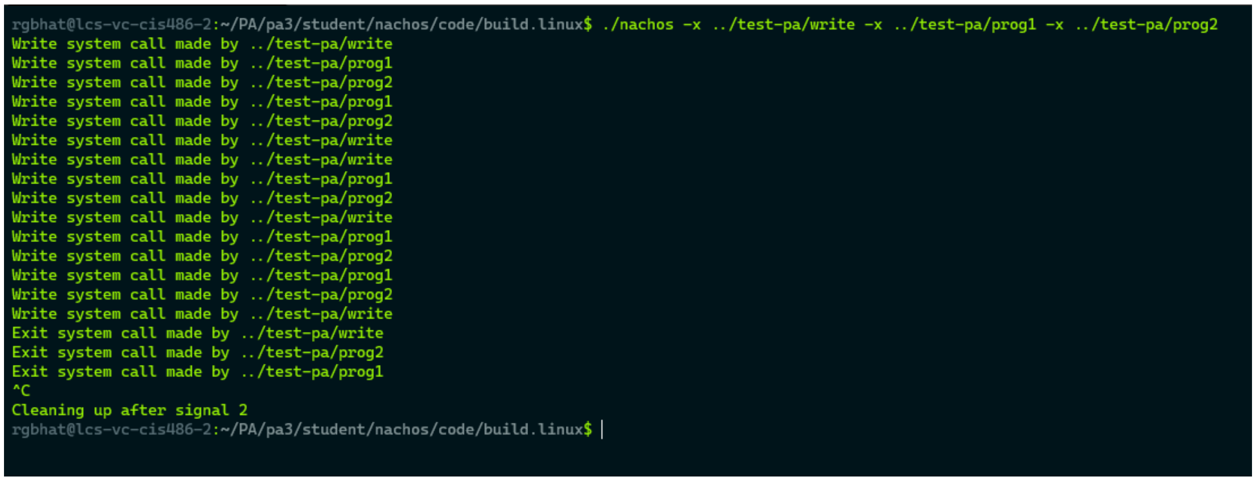
```
rgbhat@lcs-vc-cis486-2:~/PA/pa3/student/nachos/code/build.linux$ ./nachos -x ../test-pa/prog1
Write system call made by ../test-pa/prog1
Write system call made by ../test-pa/prog1
Write system call made by ../test-pa/prog1
Write system call made by ../test-pa/prog1
Write system call made by ../test-pa/prog1
Exit system call made by ../test-pa/prog1
^C
Cleaning up after signal 2
rgbhat@lcs-vc-cis486-2:~/PA/pa3/student/nachos/code/build.linux$ 
```

✪ **[R-10]:** `./nachos -x ../test-pa/prog1 -x ../test-pa/prog2`



```
rgbhat@lcs-vc-cis486-2:~/PA/pa3/student/nachos/code/build.linux$ ./nachos -x ../test-pa/prog1 -x ../test-pa/prog2
Write system call made by ../test-pa/prog1
Write system call made by ../test-pa/prog2
Write system call made by ../test-pa/prog2
Write system call made by ../test-pa/prog1
Write system call made by ../test-pa/prog1
Write system call made by ../test-pa/prog2
Write system call made by ../test-pa/prog1
Write system call made by ../test-pa/prog2
Write system call made by ../test-pa/prog2
Write system call made by ../test-pa/prog1
Exit system call made by ../test-pa/prog1
Exit system call made by ../test-pa/prog2
^C
Cleaning up after signal 2
rgbhat@lcs-vc-cis486-2:~/PA/pa3/student/nachos/code/build.linux$
```

✪ **[R-11]:** `./nachos -x ../test-pa/write -x ../test-pa/prog1 -x ../test-pa/prog2`



```
rgbhat@lcs-vc-cis486-2:~/PA/pa3/student/nachos/code/build.linux$ ./nachos -x ../test-pa/write -x ../test-pa/prog1 -x ../test-pa/prog2
Write system call made by ../test-pa/write
Write system call made by ../test-pa/prog1
Write system call made by ../test-pa/prog2
Write system call made by ../test-pa/prog1
Write system call made by ../test-pa/prog2
Write system call made by ../test-pa/write
Write system call made by ../test-pa/write
Write system call made by ../test-pa/prog1
Write system call made by ../test-pa/prog2
Write system call made by ../test-pa/write
Write system call made by ../test-pa/prog1
Write system call made by ../test-pa/prog2
Write system call made by ../test-pa/prog1
Write system call made by ../test-pa/prog2
Write system call made by ../test-pa/write
Exit system call made by ../test-pa/write
Exit system call made by ../test-pa/prog2
Exit system call made by ../test-pa/prog1
^C
Cleaning up after signal 2
rgbhat@lcs-vc-cis486-2:~/PA/pa3/student/nachos/code/build.linux$
```