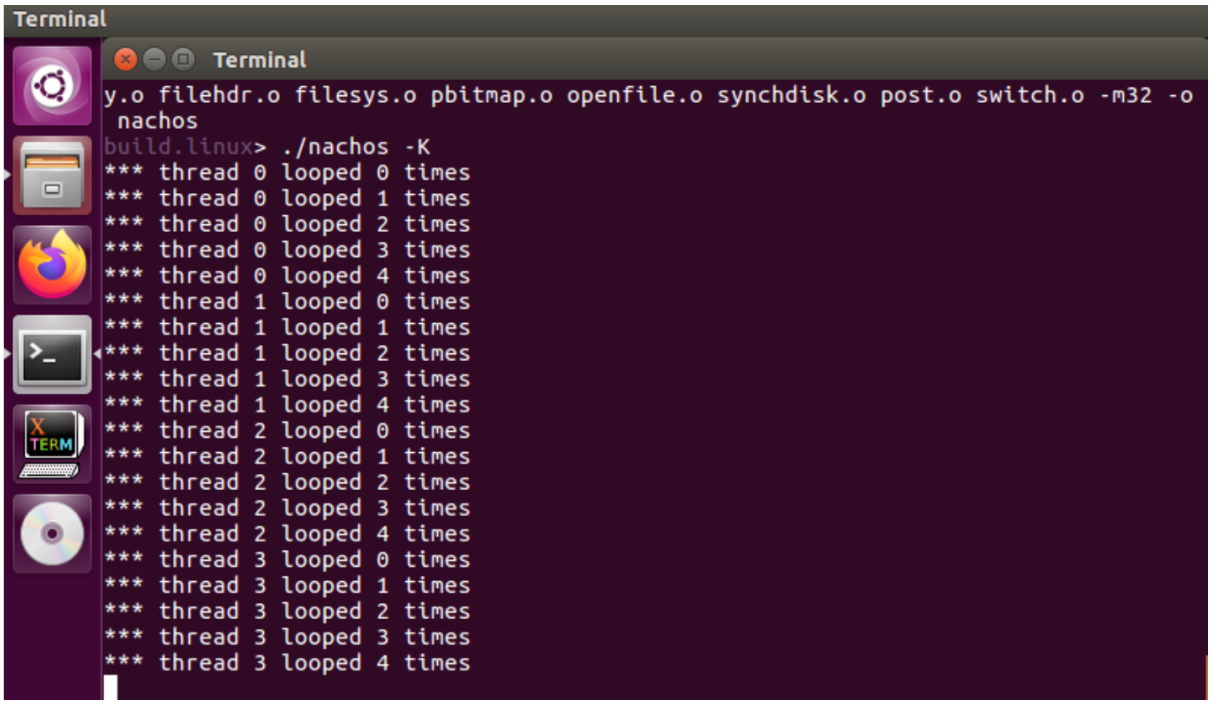
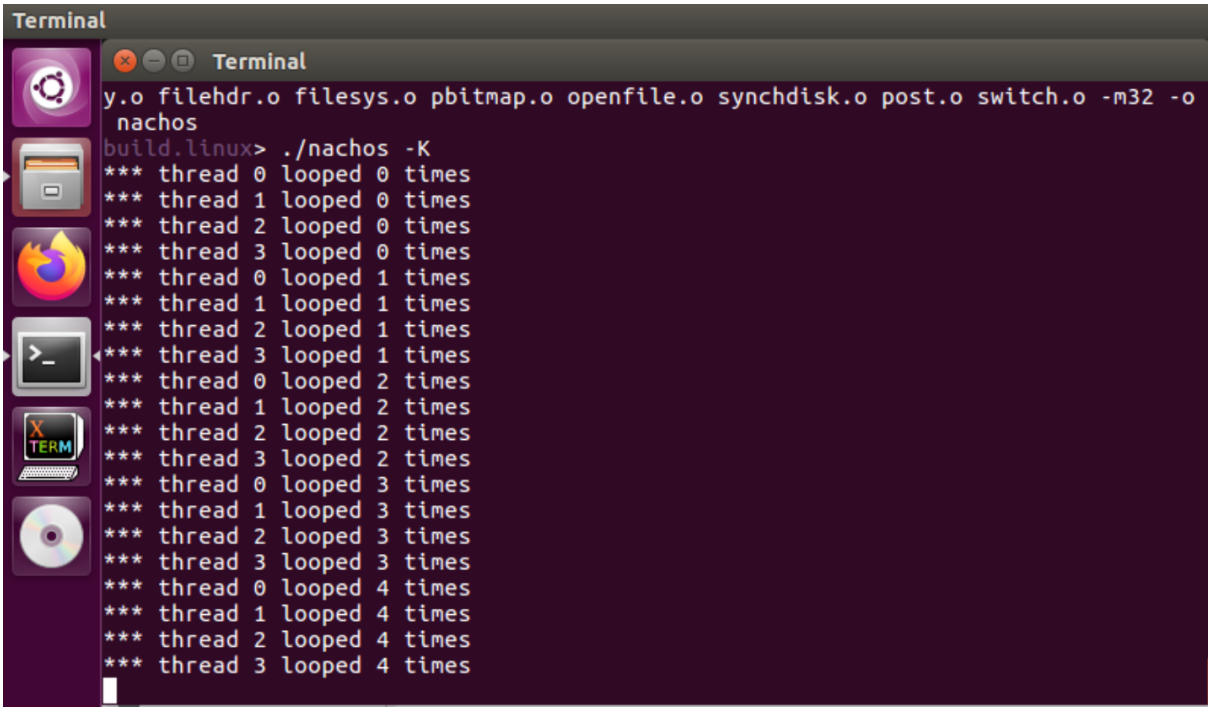
Question1:

The result before commenting out.



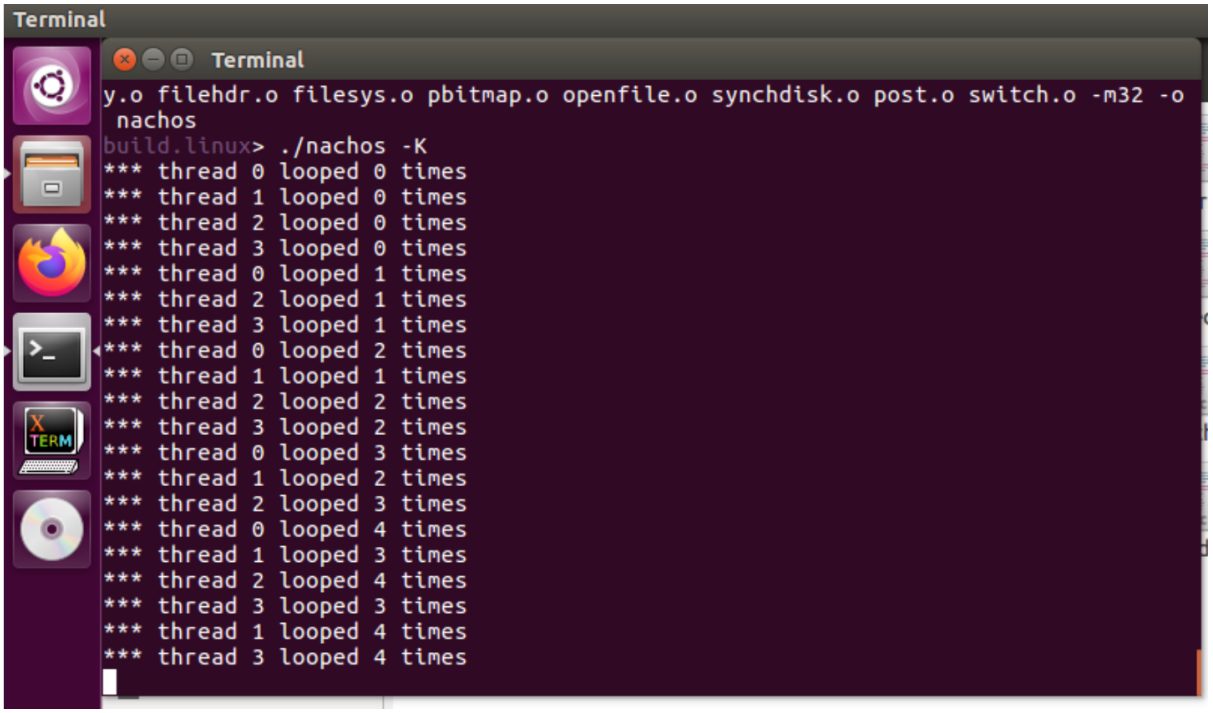
The result after commenting out.



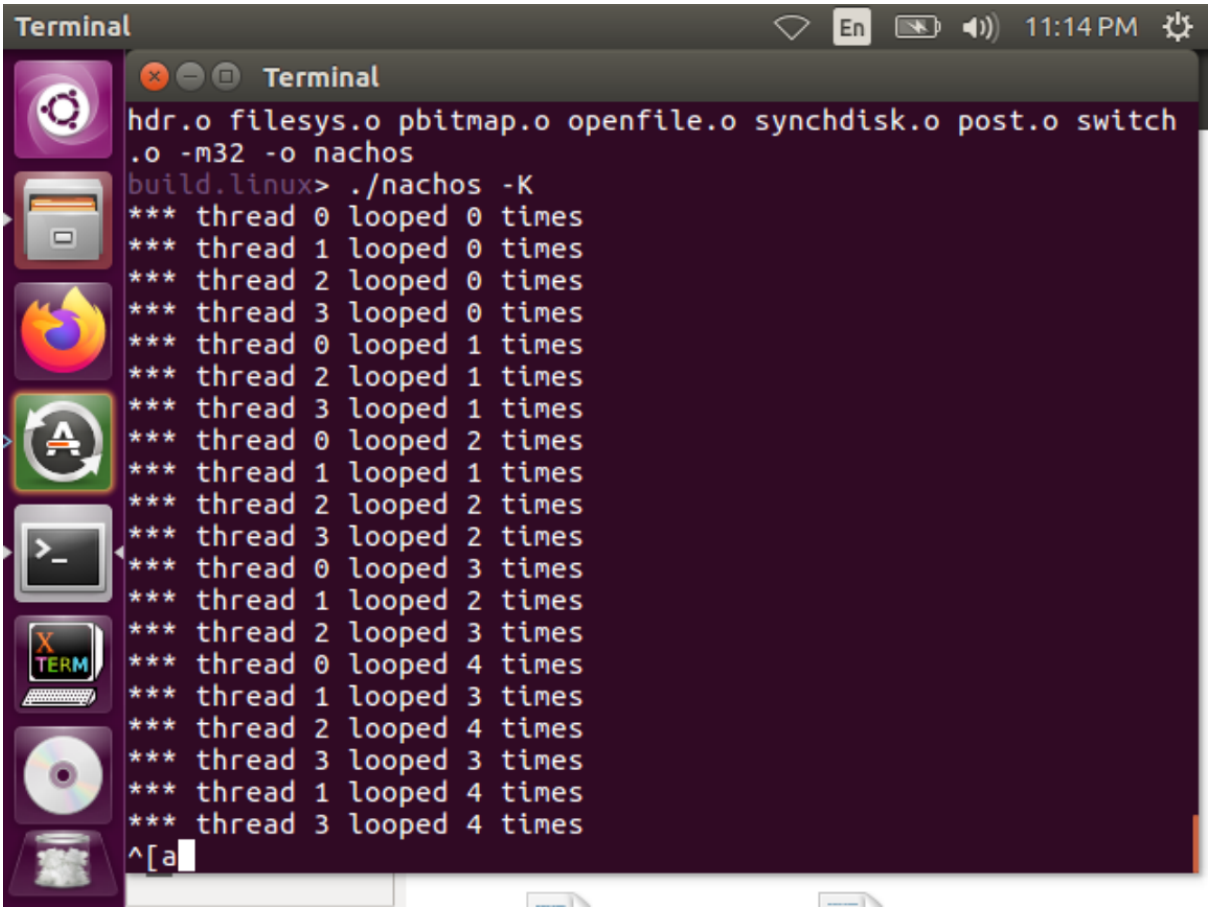
The reason why the output is different is by using yield() function, making the current thread pause and let other threads run. The yield() function would make the currently running thread head back to runnable to allow other threads of the same priority to get their turn.

Question2:

before activating the timer device



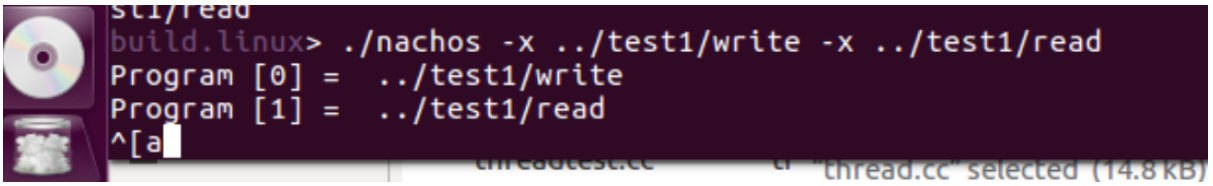
after activating the timer device



By slicing the alarm time randomly , different threads will have a different alarm time to be able to back to the ready queue. For example, thread 0 has the least alarm time, so it will go back to the ready queue first and run the second time earliest. So the thread will finish 5 loops earliest.

Question3:

Only 1 thread user program is created in current Nachos. The output will only run write or read rather than both.



main.cc creates the address space to run a user program by the following block, RunUserProg function.

```
void
RunUserProg(void *filename) {
    AddrSpace *space = new AddrSpace;
    ASSERT(space != (AddrSpace *)NULL);
    if (space->Load((char*)filename)) {  // load the program into the space
        space->Execute();         // run the program
    }
    ASSERTNOTREACHED();
}
```

AddrSpace * space =new AddrSpace; this line creates the address space for program.

Question4:

1.
Fork(*func)(arg) can allow caller and callee to execute concurrently.
The first argument *func is the procedure to run concurrently.
And the second argument arg is a single argument to be passed to the procedure.

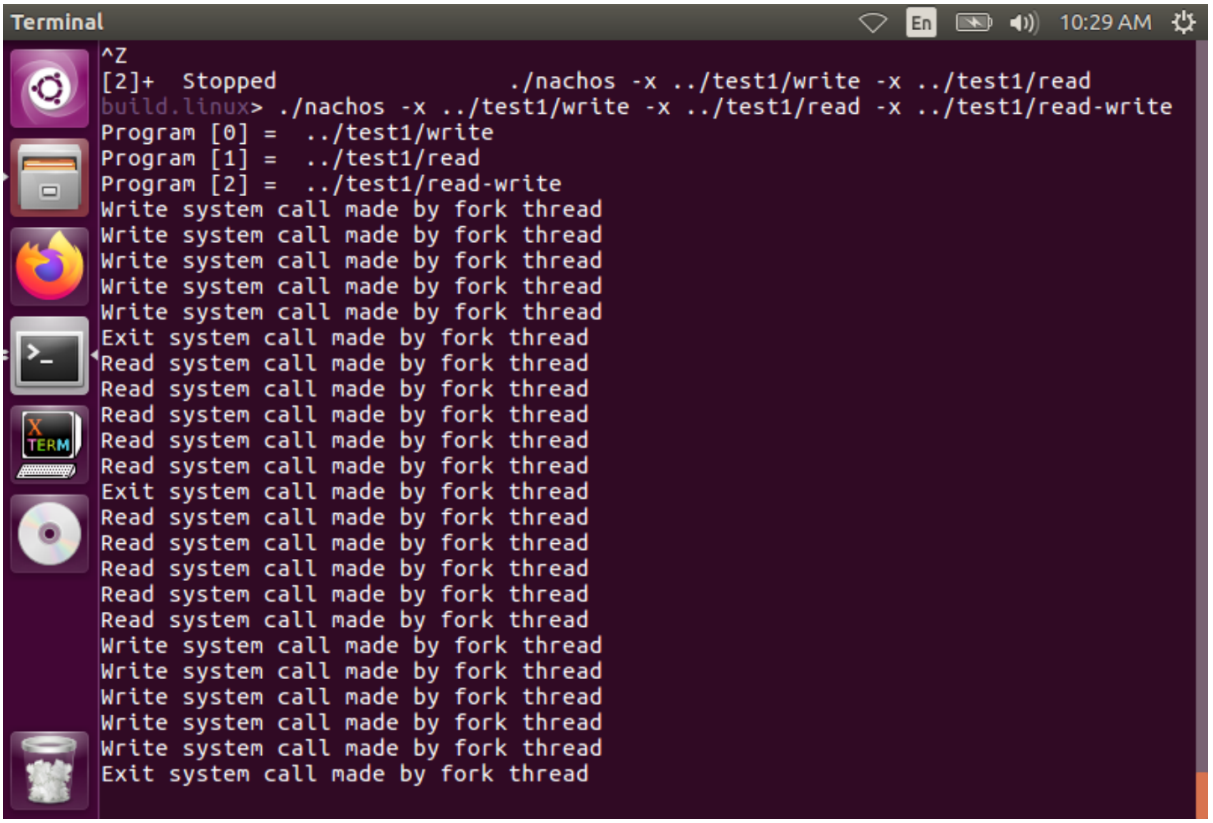
2.
If we use a pointer of function() RunUserProg, it will directly use the second argument arg as a single argument to be passed to the first argument func's procedure to run concurrently.
And the RunUserProg function will create address space for the program, load the program into the space and then run the program.

Question5:

By studying the threadtest.cc file , i slightly modified the original code and add another block, which is followed to make each user program have a user-level thread to run.

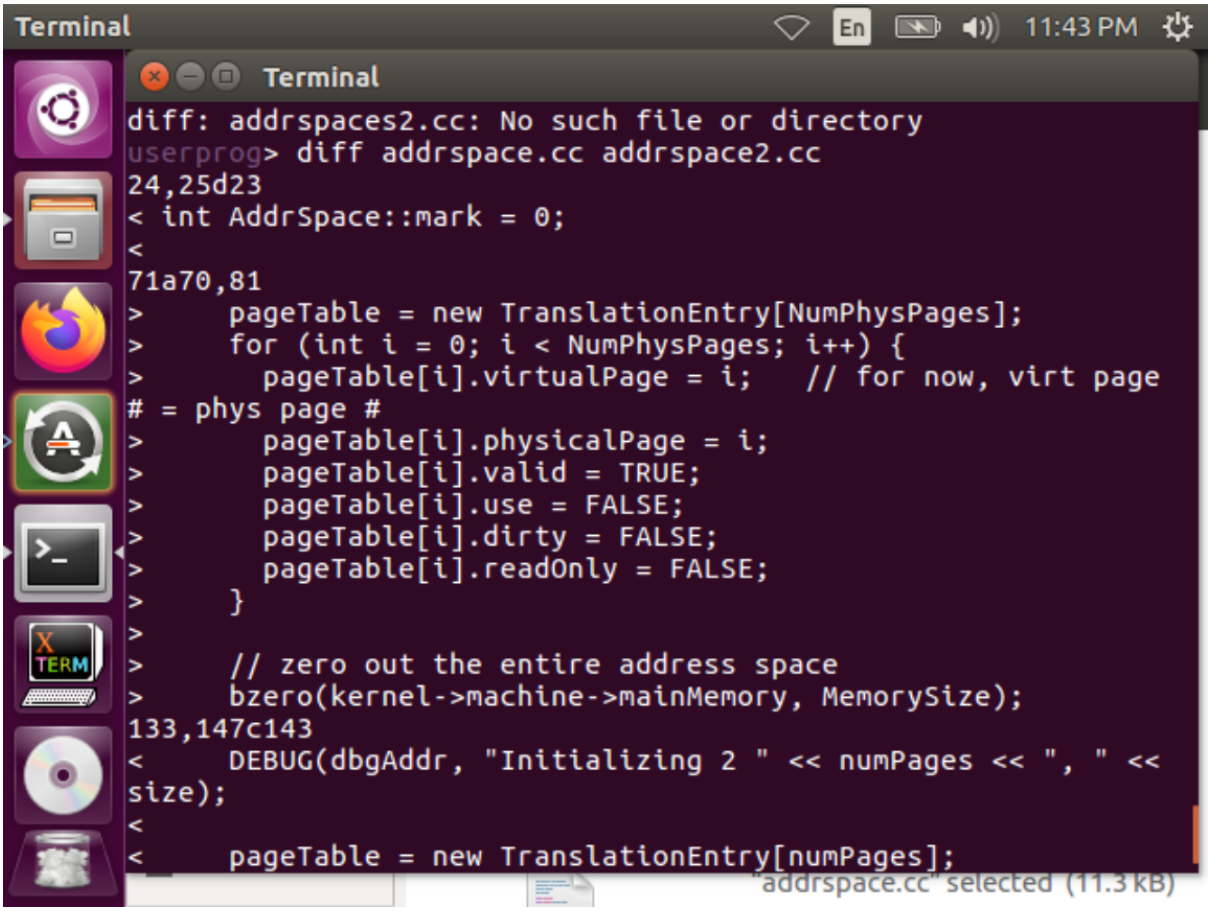
```
for(int i=0;i<Counts;i++){
Thread *t=new Thread("fork thread");// fork a new thread
t->Fork((VoidFunctionPtr) RunUserProg,(void *) UserProgram[i]);//Use RUnUserProg
 and give the function required argument UserProgram[i], which is the stored pro
grams names to run.
}
```

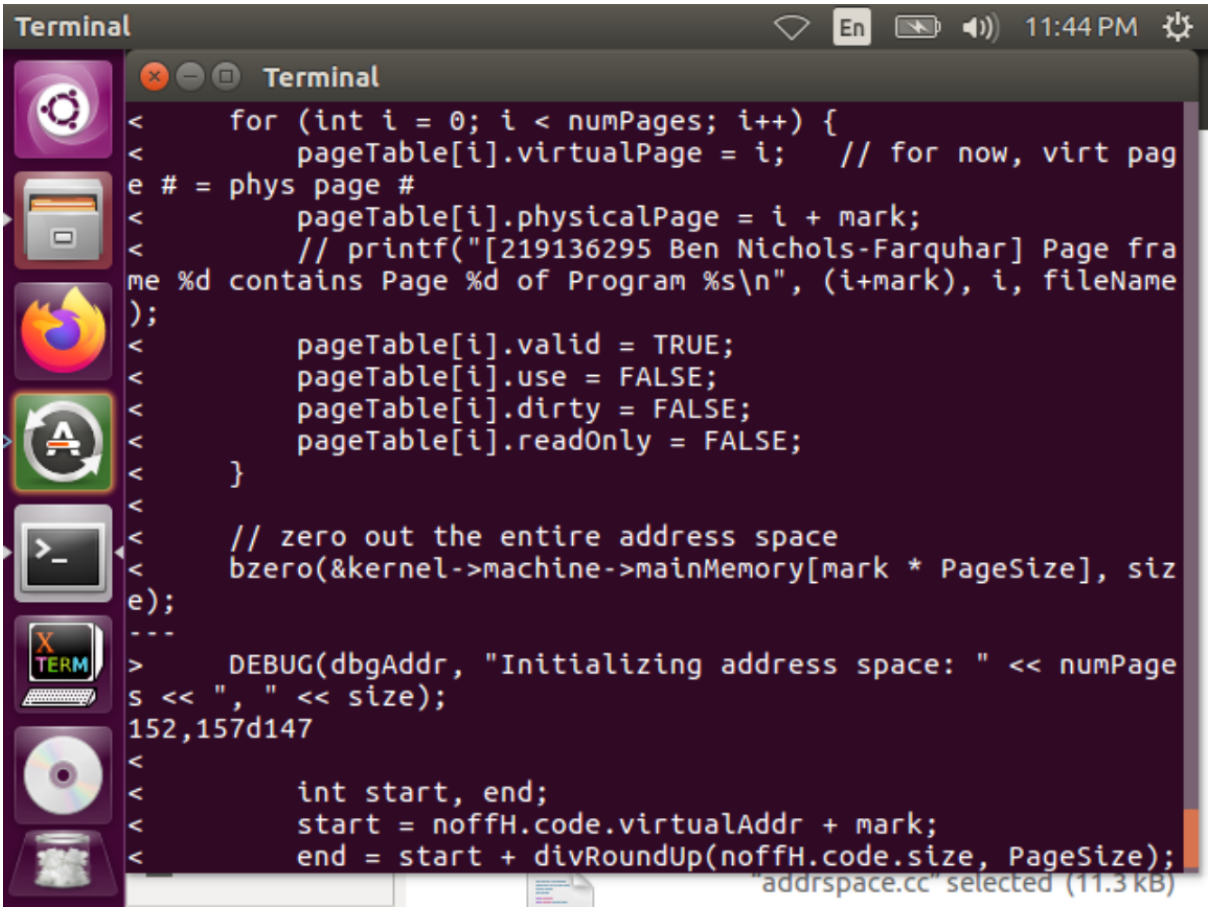
Question6 : Test and Output



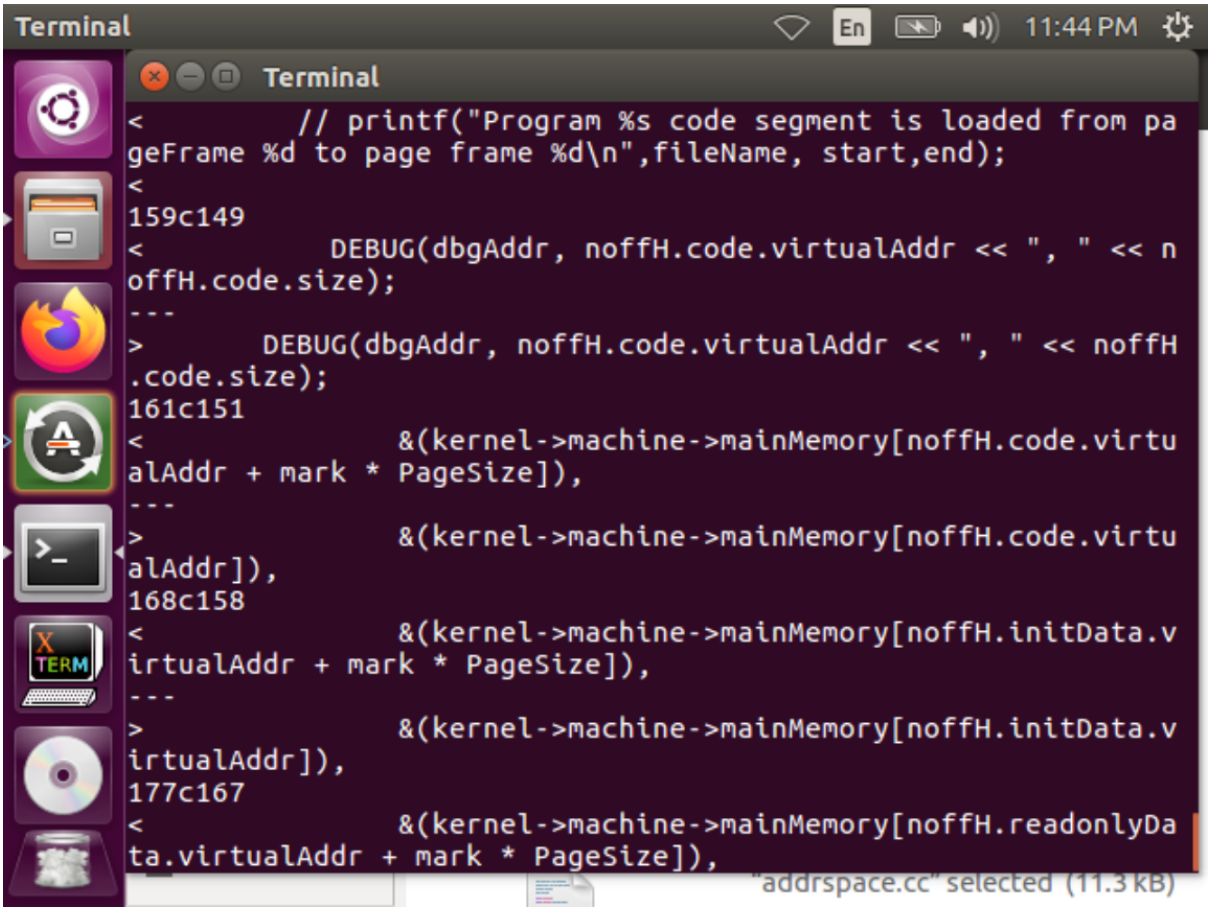
Attached

difference between original address,cc and new address.cc


Terminal
En
11:43 PM
Terminal
diff: addrspaces2.cc: No such file or directory
userprog> diff addrspace.cc addrspace2.cc
24,25d23
< int AddrSpace::mark = 0;
<
71a70,81
>   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);
133,147c143
<   DEBUG(dbgAddr, "Initializing 2 " << numPages << ", " <<
size);
<
<   pageTable = new TranslationEntry[numPages];
"addrspace.cc" selected (11.3 kB)


Terminal
En
11:44 PM
Terminal
<   for (int i = 0; i < numPages; i++) {
<   pageTable[i].virtualPage = i;   // for now, virt pag
e # = phys page #
<   pageTable[i].physicalPage = i + mark;
<   // printf("[219136295 Ben Nichols-Farquhar] Page fra
me %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], siz
e);

>   DEBUG(dbgAddr, "Initializing address space: " << numPage
s << ", " << size);
152,157d147
<
<   int start, end;
<   start = noffH.code.virtualAddr + mark;
<   end = start + divRoundUp(noffH.code.size, PageSize);
"addrspace.cc" selected (11.3 kB)


Terminal
En
11:44 PM
Terminal
< // printf("Program %s code segment is loaded from pa
geFrame %d to page frame %d\n",fileName, start,end);
<
159c149
< DEBUG(dbgAddr, noffH.code.virtualAddr << ", " << n
offH.code.size);

> DEBUG(dbgAddr, noffH.code.virtualAddr << ", " << noffH
.code.size);
161c151
< &(kernel->machine->mainMemory[noffH.code.virtu
alAddr + mark * PageSize]),

> &(kernel->machine->mainMemory[noffH.code.virtu
alAddr]),
168c158
< &(kernel->machine->mainMemory[noffH.initData.v
irtualAddr + mark * PageSize]),

> &(kernel->machine->mainMemory[noffH.initData.v
irtualAddr]),
177c167
< &(kernel->machine->mainMemory[noffH.readonlyDa
ta.virtualAddr + mark * PageSize]),
"addrspace.cc" selected (11.3 kB)