CIS-657 Fall-2019

Principals of Operating System

Programming Assignment -2

Submitted By:
Surabhi Shail

Design

AIM: To implement your own Virtual Memory System and user defined syscalls in Nachos.

Design for TASK1: Implement system calls and exception handling for user programs.

To achieve Task 1 we have to implement different system calls. We need to implement ConsoleRead,ConsoleWrite, ThreadFork,Exec and ThreadExit system calls.

To successfully implement this task I have made changes in the following files.

1. In test folder I have added ConsoleRead.c, ConsoleWrite.c ,thread_fork.c and thread_Exec.c files.Then changes are done in Makefile of test folder to add output files for new user programs.Also Start.c file is updated.

2.In userprog folder we have made changes in Exception.cc and added system call cases to handle the newly defined system calls such as SC_ConsoleRead and SC_ConsoleWrite and predefined system calls like SC_ThreadFork, SC_Exec and SC_ThreadExit.

Design for TASK 2: Implement multiprogramming

To implement multiprogramming, changes are done in main.cc. Initially by using –x option only a single user program can execute.

So,to enable the functionality to run multiple user programs at the same time on a uniprocessor we have implemented a data structure which can maintain a list of user program ready for execution. Now, each program will be loaded into main memory and a new thread for each program is created to start the execution.

We have also implemented a quantum of 400 for setting time slice for the execution of each user program so that they get a fair share of CPU.

Design for TASK 3: Implement Memory Manager (Virtual Memory) with software TLB

In this task we need to create a virtual address space whose size is unlimited so that it gives the illusion of user program executing in large size physical memory. To implement this task we need to make sure that the address space for each user program don't overlap with other.

To achieve this task we our nachos should run programs part by part depending upon the part of address space which is loaded currently in the main memory

The steps followed to implement this task are:

For this step changes are done in load function of addrspace.cc during loading of data segment and code segment.

In translate.h we can see the structure of TranslationEntry. In translate.cc we have written the logic to translate the virtual address into physical address using pagetable.

For the swapping in and out of page tables we have implemented a swapFile. Also to find free pages I have implemented a hashmap which maps each thread with an int value as Thread locations.

In Kernel.h I have added swapFile as SwapTable and threadLocations as frame table and Bitmap to hold memory status

For swapping of pages when the number of pages requested is greater than the available frames we have implemented random replacement algorithm. This algorithm replaces any random page from the memory. To generate this random page we have used rand() function. The advantage of using this algorithm is elimination of the overhead cost related to LRU algorithm of tracking the references of page.

Implementation

Implementation in TASK 1:

Syscall.h

Defined 2 new cases

```
#define SC_ConsoleRead   16
#define SC_ConsoleWrite  17
```

Added declaration of ConsoleRead and write

```
int ConsoleRead(char *buffer, int size);
int ConsoleWrite(char *buffer, int size);
```

Exception.cc

Added cases and how to handle them

```
case SC_ConsoleRead:
{
std::cout << "----------ConsoleRead Case in exception.cc----------\n"
<< endl;
int a = (int)kernel->machine->ReadRegister(4);
int b = (int)kernel->machine->ReadRegister(5);
int result = SysRead((int)a, (int)b);

kernel->machine->WriteRegister(2, (int)result);
```

```
{
/* 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;
}
// case for Write System call
case SC_ConsoleWrite:
{

std::cout << "----------ConsoleWrite  call in exception.cc--------\n"
<< endl;
int a = (int)kernel->machine->ReadRegister(4);
int b = (int)kernel->machine->ReadRegister(5);
int result = WriteSys((int)a, (int)b);

kernel->machine->WriteRegister(2, (int)result);
{
kernel->machine->WriteRegister(PrevPCReg, kernel->machine->ReadRegister(PCReg));

kernel->machine->WriteRegister(PCReg, kernel->machine->ReadRegister(PCReg) + 4);

kernel->machine->WriteRegister(NextPCReg, kernel->machine->ReadRegister(PCReg) + 4);
}
return;

ASSERTNOTREACHED();
break;
}

case SC_ThreadFork: // case for Fork System call
{
std::cout << "---------ThreadFork  call in exception.cc---------\n"
<< endl;

Thread *th = new Thread("Forking");
th->Fork((VoidFunctionPtr)execute, (void *)0);
```

```
{
kernel->machine->WriteRegister(PrevPCReg, kernel->machine->ReadRegister(PCReg));

kernel->machine->WriteRegister(PCReg, kernel->machine->ReadRegister(PCReg) + 4);

kernel->machine->WriteRegister(NextPCReg, kernel->machine->ReadRegister(PCReg) + 4);
}
return;

ASSERTNOTREACHED();
break;
}

case SC_Exec:
{
std::cout << "---------Exec call in exception.cc---------\n"
<< endl;

char file[100];
int k = 0;
int memval;

int vaddr = (int)kernel->machine->ReadRegister(4);
kernel->machine->ReadMem(vaddr, 1, &memval);
while ((*(char *)&memval) != '\0')
{
file[i] = (char)memval;
++k;
vaddr++;
kernel->machine->ReadMem(vaddr, 1, &memval);
}
filename[k] = (*(char *)&memval);
OpenFile *executable = kernel->fileSystem->Open(file);
AddrSpace *space;

if (executable == NULL)
{
printf("Unable to open file %s\n", file);
return;
}
space = new AddrSpace(executable);
kernel->currentThread->space = space;

delete executable;
space->RestoreState();
space->InitRegisters();

kernel->machine->Run();
```

```
return;

ASSERTNOTREACHED();
break;
}

//Adding new case to handle Exit of current thread
case SC_ThreadExit:
{
std::cout << "\n---------ThreadExit  call in exception.cc---------\n"
<< endl;
std::cout << "Exit called for the current thread :" << kernel->currentThread->getName() << "with
ThreadId " << kernel->currentThread->thread_id << "   "<<kernel->machine->ReadRegister(4) << 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);
}
kernel->currentThread->Finish();
return;
ASSERTNOTREACHED();

break;
}
```

Implementation of TASK2:

To implement multiprogramming
Main.cc:

```
 std::vector<char *> Proglist;

for(int i =0; i<ProgList.size();i++){

      Thread *userp = NULL;

       cout << "Thread Created for execution of Program\n" << ProgList[i];

       userp = new Thread("ProgramList");

       userp->Fork((VoidFunctionPtr) RunUserProg, (void *) ProgList[i]);
```

To implement quantum:

Timer.cc

```
kernel->interrupt->Schedule(this,400, TimerInt);
```

Interrupt.cc

```
ListIterator<PendingInterrupt *> *it = new ListIterator<PendingInterrupt *>(pending);
        PendingInterrupt* pi;
        while( !it->IsDone()) {
                PendingInterrupt* obj= it->Item();
                it->Next();
                if(obj->type==TimerInt)
                {
                        pending->Remove(obj);
                        pi=obj;
                }
        }
        pi->when=kernel->stats->totalTicks+t;
        pending->Insert(pi);
        delete it;
```

Implementation of TASK3:

1.addrspace.cc

Addrspace::Load()
{
Intializing code segment and data segment

```
int lastposition = kernel->threadLocations[kernel->currentThread];

for (int pn = 0; pn < numPages; pn++)
{
char * data = new char[PageSize];
executable->ReadAt(data, PageSize, noffH.code.inFileAddr + pn * PageSize);
kernel->swapFile->WriteAt(data, PageSize, lastposition + pn* PageSize);

int availableFrame = kernel->machine->memoryStatus->FindAndSet();
if (availableFrame != -1)
{
kernel->machine->pageTable[availableFrame].virtualPage = pn;
kernel->machine->pageTable[availableFrame].physicalPage = availableFrame;
kernel->machine->pageTable[availableFrame].valid = TRUE;
kernel->machine->pageTable[availableFrame].threadId = kernel->currentThread;
```

```
bzero(&(kernel->machine->mainMemory[availableFrame * PageSize]), PageSize);
bcopy(codeVal, &(kernel->machine->mainMemory[availableFrame * PageSize]), PageSize);
}
delete data;
}
```

Translate.cc

```
virtual_pn = (unsigned) virtAddr / PageSize;
pos_set = (unsigned) virtAddr % PageSize;

        int pPN = -1;
        if (tlb == NULL)
        {
                int page_cnt = 0;

                while (page_cnt < NumPhysPages)
                {
                        if (pageTable[page_cnt].threadId == kernel->currentThread
                                && pageTable[page_cnt].virtualPage == virtual_pn)
                        {
                                entry = &(pageTable[pages]);
                                pPN = page_cnt;
                                page_cnt = NumPhysPages;
                        }
                        Page_cnt++;
                }
                if (pPN == -1 || !entry->valid)
                {
                        std::cout << "Page Fault Exception Occured" << endl;
                        return PageFaultException;
                }
        }
```

Page Fault case handling:

In Exception.cc

```
        //Adding new case to handle Page fault Exception
case PageFaultException: // case for handling the page fault exceptions
{
unsigned vPage = (unsigned)(kernel->machine->ReadRegister(BadVAddrReg) / PageSize);


char *vmData = new char[PageSize];
```

```
int nPLocation = kernel->threadLocations[kernel->currentThread];

kernel->swapFile->ReadAt(vmData, PageSize, nPLocation + vPage * PageSize);

int virtualAddress = kernel->threadLocations[kernel->currentThread] + (vPage * PageSize);


int availableFrame = kernel->machine->memoryStatus->FindAndSet();

if (availableFrame == -1)
{
Thread *beforeThreadid = kernel->machine->pageTable[randomPage].threadId;

int randomPage = rand() % NumPhysPages;
char *pageData = new char[PageSize];
int beforeVPage = kernel->machine->pageTable[randomPage].vPage;

bcopy(&(kernel->machine->mainMemory[kernel->machine->pageTable[randomPage].physicalPage *
PageSize]), pageData, PageSize);
kernel->swapFile->WriteAt(pageData, PageSize, kernel->threadLocations[beforeThreadid] +
(beforeVPage * PageSize));
kernel->machine->pageTable[randomPage].valid = TRUE;

kernel->machine->pageTable[randomPage].vPage = vPage;
kernel->machine->pageTable[randomPage].threadId = kernel->currentThread;

bzero(&(kernel->machine->mainMemory[randomPage * PageSize]), PageSize);
bcopy(vmData, &(kernel->machine->mainMemory[randomPage * PageSize]), PageSize);

delete pageData;
}
else
{
kernel->machine->pageTable[availableFrame].vPage = vPage;
kernel->machine->pageTable[availableFrame].threadId = kernel->currentThread;
kernel->machine->pageTable[availableFrame].physicalPage = availableFrame;

kernel->machine->pageTable[availableFrame].valid = TRUE;
bcopy(vmData, &(kernel->machine->mainMemory[availableFrame * PageSize]), PageSize);

bzero(&(kernel->machine->mainMemory[availableFrame * PageSize]), PageSize);
}
delete vmData;
break;
}
```

Implementation of Demand paging:

Kernel.h

```
std::map<Thread*, int> threadLocations;

//initialising the variable for swap file space
OpenFile* swapFile;
int swap_pos;
//initialising a Bitmap
 Bitmap *memoryStatus;
```

Testing

Building the program:

1. Make clean in coff2noff folder
2. Make depend in coff2noff folder
3. Make clean in code/test folder
4. Make console_write.c, console_read.c,thread_fork.c and thread_Exec.c

Testing the program:

1. Make depend in build.linux folder
2. Make nachos in build.linux folder
3. Run ./nachos –quantum 400 –x ../test/program(replace program name with your program) command in build.linux command folder to get the output.
4. To test multiple program . run ./nachos –quantum 400 –x ../test/prg1 –x../test/prg2 … command in build.linux folder to get the output.

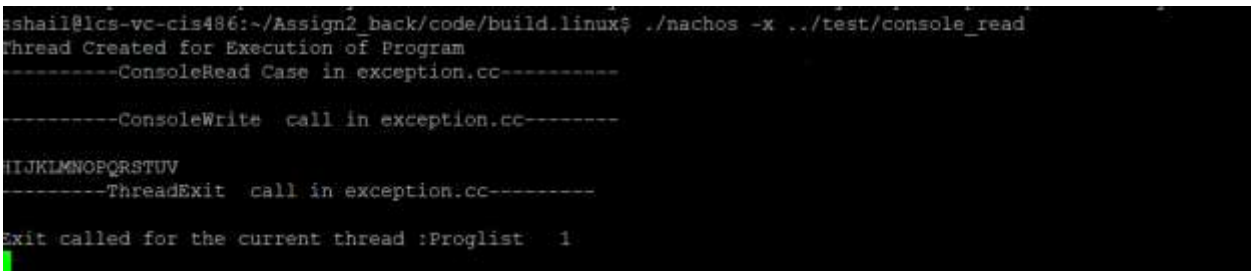
Implementation of Task4:

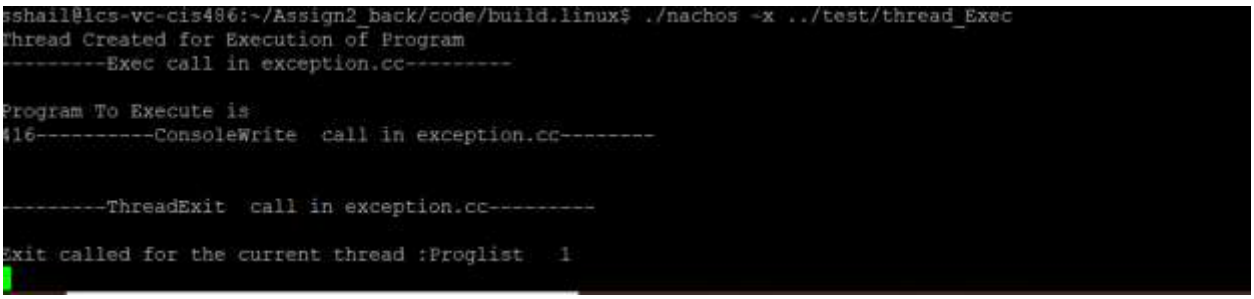
- Written test programs to test each Task

Task 1, run a single user program

- To test read/write/exit syscalls.
- To test exec/exit syscalls by calling Exec x times
- To test fork syscalls.

Test Result:

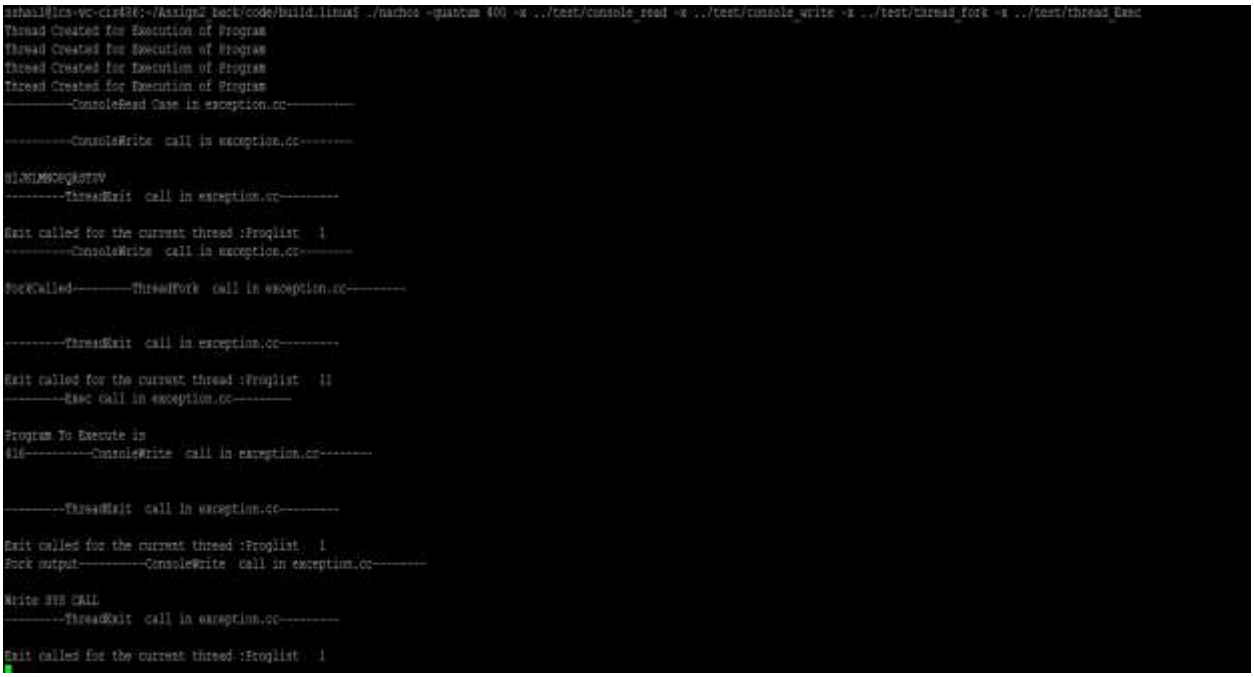




Task 2, to run multiple programs

- Running 4 programs in parallel.

Test Result:



Task 3, running larger memory need programs like three matmult

Program.

Test Result:

<u>Output</u>

Final Output

```
sshail@lcs-vc-cis486:~/Assign2_nachos/code/build.linux$ ./nachos -x ../test/console_write
Thread Created for the Execution of Program
Intializing Address space:12,1536
----------Intialializing Code Segment-------------
Last Location in Page Table :0
Reading at filelocation52
Writing at location 0
Frames Allocated0
Reading at filelocation180
Writing at location 128
Frames Allocated1
Reading at filelocation308
Writing at location 256
Frames Allocated2
Reading at filelocation436
Writing at location 384
Frames Allocated3
Reading at filelocation564
Writing at location 512
Frames Allocated4
Reading at filelocation692
Writing at location 640
Frames Allocated5
Reading at filelocation820
Writing at location 768
Frames Allocated6
Reading at filelocation948
Writing at location 896
Frames Allocated7
Reading at filelocation1076
Writing at location 1024
Frames Allocated8
Reading at filelocation1204
Writing at location 1152
Frames Allocated9
Reading at filelocation1332
Writing at location 1280
Frames Allocated10
Reading at filelocation1460
Writing at location 1408
Frames Allocated11
----------ConsoleWrite  call in exception.cc--------

Write SYS CALL
---------ThreadExit  call in exception.cc---------

Exit called for the current thread :Programlistwith ThreadId 1   1
```

```
 After Matrix Multiplication Value is7220sshail@lcs-vc-cis486:~/Assign2_nachos/code/build.linux$ ./nachos -x ../test/thread_fork
Thread Created for the Execution of Program
Intializing Address space:13,1664
----------Intialializing Code Segment-------------
Last Location in Page Table :0
Reading at filelocation52
Writing at location 0
Frames Allocated0
Reading at filelocation100
Writing at location 128
Frames Allocated1
Reading at filelocation308
Writing at location 256
Frames Allocated2
Reading at filelocation436
Writing at location 384
Frames Allocated3
Reading at filelocation564
Writing at location 512
Frames Allocated4
Reading at filelocation692
Writing at location 640
Frames Allocated5
Reading at filelocation820
Writing at location 768
Frames Allocated6
Reading at filelocation940
Writing at location 896
Frames Allocated7
Reading at filelocation1076
Writing at location 1024
Frames Allocated8
Reading at filelocation1204
Writing at location 1152
Frames Allocated9
Reading at filelocation1332
Writing at location 1280
Frames Allocated10
Reading at filelocation1460
Writing at location 1408
Frames Allocated11
Reading at filelocation1588
Writing at location 1536
Frames Allocated12
----------ConsoleWrite  call in exception.cc--------

ForkCalled---------ThreadFork  call in exception.cc---------
```

Multiprogramming :Running Matmult

```
sshail@lcs-vc-cis406:~/Assign2_nachos/code/build.linux$ ./nachos -x ../test/matmult ./nachos -x ../test/matmult ./nachos -x ../test/matmult
Thread Created for the Execution of Program
Thread Created for the Execution of Program
Thread Created for the Execution of Program
Intializing Address space:55,7040
----------Intialializing Code Segment------------
Last Location in Page Table :0
Reading at filelocation52
Writing at location 0
Frames Allocated0
Reading at filelocation180
Writing at location 128
Frames Allocated1
Reading at filelocation308
Writing at location 256
Frames Allocated2
Reading at filelocation436
Writing at location 384
Frames Allocated3
Reading at filelocation564
Writing at location 512
Frames Allocated4
Reading at filelocation692
Writing at location 640
Frames Allocated5
Reading at filelocation820
Writing at location 768
Frames Allocated6
Reading at filelocation948
Writing at location 896
Frames Allocated7
Reading at filelocation1076
Writing at location 1024
Frames Allocated8
Reading at filelocation1204
Writing at location 1152
Frames Allocated9
Reading at filelocation1332
Writing at location 1280
Frames Allocated10
Reading at filelocation1460
Writing at location 1408
Frames Allocated11
Reading at filelocation1588
Writing at location 1536
```

```
Reading at filelocation6324
Writing at location 13312
Frames Allocated104
Reading at filelocation6452
Writing at location 13440
Frames Allocated105
Reading at filelocation6580
Writing at location 13568
Frames Allocated106
Reading at filelocation6708
Writing at location 13696
Frames Allocated107
Reading at filelocation6836
Writing at location 13824
Frames Allocated108
Reading at filelocation6964
Writing at location 13952
Frames Allocated109
Intializing Address space:55,7040
----------Intialializing Code Segment-------------
Last Location in Page Table :14080
Reading at filelocation52
Writing at location 14080
Frames Allocated110
Reading at filelocation180
Writing at location 14208
Frames Allocated111
Reading at filelocation308
Writing at location 14336
Frames Allocated112
Reading at filelocation436
Writing at location 14464
Frames Allocated113
Reading at filelocation564
Writing at location 14592
Frames Allocated114
Reading at filelocation692
Writing at location 14720
Frames Allocated115
Reading at filelocation820
Writing at location 14848
Frames Allocated116
Reading at filelocation948
Writing at location 14976
Frames Allocated117
Reading at filelocation1076
Writing at location 15104
Frames Allocated118
Reading at filelocation1204
Writing at location 15232
```

```
Frames Allocated-1
Page Fault Exception Occured
---------------PageFaultExecpection in Exeception.cc -------------
------------Available Frames in RAM--------- =-1
Swapping Pages
Random Page Selected for replacement is =103Swap out from Virtual Page:48Thread id :161892816
Swap in at Virtual Page:54Thread id :161926096
Page Fault Exception Occured
---------------PageFaultExecpection in Exeception.cc -------------
------------Available Frames in RAM--------- =-1
Swapping Pages
Random Page Selected for replacement is =70Swap out from Virtual Page:15Thread id :161892816
Swap in at Virtual Page:21Thread id :161926096
Page Fault Exception Occured
---------------PageFaultExecpection in Exeception.cc -------------
------------Available Frames in RAM--------- =-1
Swapping Pages
Random Page Selected for replacement is =105Swap out from Virtual Page:50Thread id :161892816
Swap in at Virtual Page:33Thread id :161926096
Page Fault Exception Occured
---------------PageFaultExecpection in Exeception.cc -------------
------------Available Frames in RAM--------- =-1
Swapping Pages
Random Page Selected for replacement is =115Swap out from Virtual Page:5Thread id :161926096
Swap in at Virtual Page:34Thread id :161926096
Page Fault Exception Occured
---------------PageFaultExecpection in Exeception.cc -------------
------------Available Frames in RAM--------- =-1
Swapping Pages
Random Page Selected for replacement is =81Swap out from Virtual Page:26Thread id :161892816
Swap in at Virtual Page:22Thread id :161926096
Page Fault Exception Occured
---------------PageFaultExecpection in Exeception.cc -------------
```

```
---------------PageFaultExecpection in Exeception.cc -------------
------------Available Frames in RAM--------- =-1
Swapping Pages
Random Page Selected for replacement is =58Swap out from Virtual Page:25Thread id :161926096
Swap in at Virtual Page:18Thread id :161892816
Page Fault Exception Occured
---------------PageFaultExecpection in Exeception.cc -------------
------------Available Frames in RAM--------- =-1
Swapping Pages
Random Page Selected for replacement is =123Swap out from Virtual Page:17Thread id :161859536
Swap in at Virtual Page:25Thread id :161926096
Page Fault Exception Occured
---------------PageFaultExecpection in Exeception.cc -------------
------------Available Frames in RAM--------- =-1
Swapping Pages
Random Page Selected for replacement is =41Swap out from Virtual Page:24Thread id :161926096
Swap in at Virtual Page:45Thread id :161926096
Page Fault Exception Occured
---------------PageFaultExecpection in Exeception.cc -------------
------------Available Frames in RAM--------- =-1
Swapping Pages
Random Page Selected for replacement is =81Swap out from Virtual Page:22Thread id :161926096
Swap in at Virtual Page:24Thread id :161926096
Page Fault Exception Occured
---------------PageFaultExecpection in Exeception.cc -------------
------------Available Frames in RAM--------- =-1
Swapping Pages
Random Page Selected for replacement is =102Swap out from Virtual Page:43Thread id :161859536
Swap in at Virtual Page:22Thread id :161926096

 After Matrix Multiplication Value is7220sshail@lcs-vc-cis486:~/Assign2_nachos/code/build.linux$
```