Nachos Porject1

生醫電資所碩一 R07945029 王思敏

Why the result is not congruent with expected:

What resulted in the unexpected result is that the virtual memory is only turned on but not be managed for multiple programs. Therefore, when several programs execute at the same time, program will always allocate to the same place and overlap on the part which other programs are using. To avoid this situation, mapping between the virtual memory of the program and the unused memory and allocate the physical address to program are needed.

The plan you take to fix the problem in Nachos & some analysis & code segments:

1. Eliminate the inessential mapping

Take away the unnecessary mapping.

Code:

code/userprog/addrspace.cc → AddrSpace::AddrSpace()

```
AddrSpace::AddrSpace()
{

    // zero out the entire address space
    //bzero(kernel->machine->mainMemory, MemorySize);
}
```

2. Allocate the free pages

Check page by page to find the first not-used physical page and allocate the appropriate size of physical page.

Code:

code/userprog/addrspace.cc → bool AddrSpace::Load(char *fileName)

```
    pageTable = new TranslationEntry[numPages];
    for (unsigned int i = 0, j = 0; i < numPages; i++) {

        pageTable[i].virtualPage = i;   // for now, virt page # = phys page #

        while(usedPhyPages[j++] != FALSE)  // find first not-used phys. page
            { /* empty */ }
        usedPhyPages[j-1] = TRUE;
        pageTable[i].physicalPage = j - 1;

        //  pageTable[i].physicalPage = 0;
        pageTable[i].valid = TRUE;
        //  pageTable[i].valid = FALSE;
        pageTable[i].use = FALSE;
        pageTable[i].dirty = FALSE;
        pageTable[i].readOnly = FALSE;
    }

    size = numPages * PageSize;

    ASSERT(numPages <= NumPhysPages);       // check we're not trying
                                            // to run anything too big --
                                            // at least until we have
                                            // virtual memory

    DEBUG(dbgAddr, "Initializing address space: " << numPages << ", " << size);
```

3. Find the physical address after mapping and data segments into memory

To find the physical address, no. page is calculated by virtualAddr divided by PageSize first. pageTable is indexed to find the corresponding physical page, then the physical memory is obtained by multiplying by PageSize. Finally, the physical address is the physical page added with the offset which is virtualAddr % PageSize

Code:

code/userprog/addrspace.cc → bool AddrSpace::Load(char *fileName)

```
    DEBUG(dbgAddr, "Initializing address space: " << numPages << ", " << size);


    // copy in the code and data segments into memory
    if (noffH.code.size > 0) {
        DEBUG(dbgAddr, "Initializing code segment.");
        DEBUG(dbgAddr, noffH.code.virtualAddr << ", " << noffH.code.size);

        // find the physical address (= physical page number + offset)
        physicalAddr = pageTable[noffH.code.virtualAddr/PageSize].physicalPage * PageSize
            + noffH.code.virtualAddr % PageSize;

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

        // find the physical address (= physical page number + offset)
        physicalAddr = pageTable[noffH.initData.virtualAddr/PageSize].physicalPage * PageSize
            + noffH.initData.virtualAddr % PageSize;

        executable->ReadAt(
                &(kernel->machine->mainMemory[physicalAddr]),
                noffH.initData.size, noffH.initData.inFileAddr);
    }

    delete executable;          // close file
    return TRUE;            // success
}
```

4. Free memory space

After execution, change the usedPhyPages back to FALSE to release the using physical pages for other programs' using.

Code:

```
// free memory space
AddrSpace::~AddrSpace()
{
    for(int i = 0; i < numPages; ++i) {
        usedPhyPages[pageTable[i].physicalPage] = FALSE;
    }

    delete pageTable;
}
```

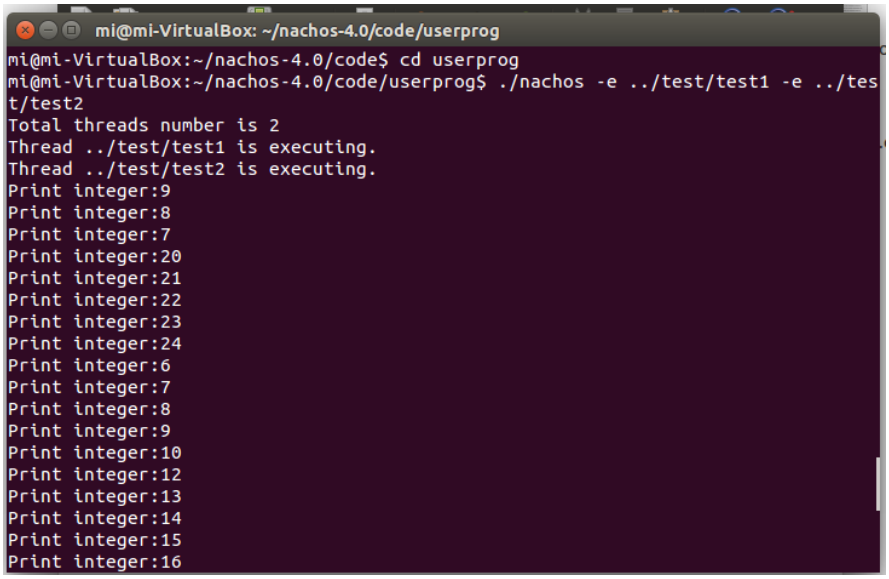
5. Globally record physical page usage

code/userprog/addrspace.h → class AddrSpace{ public: }

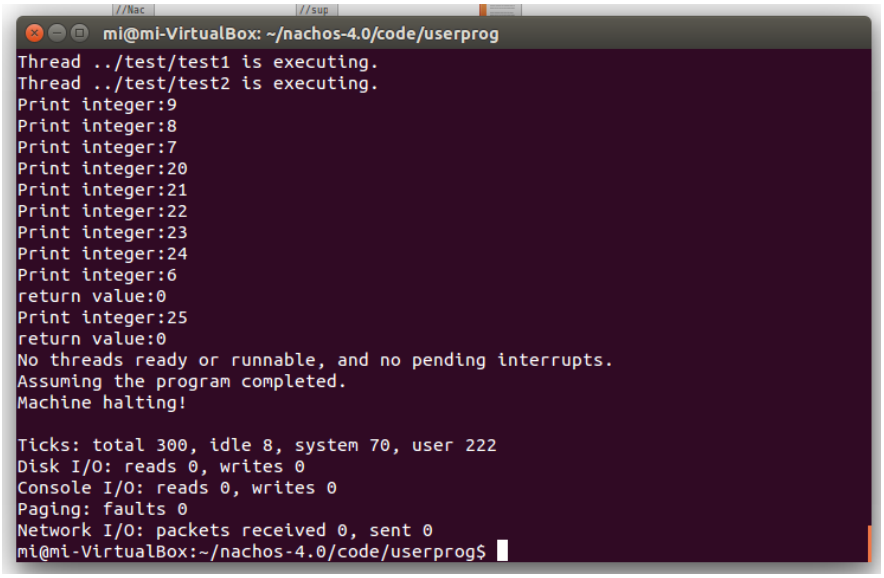
static bool usedPhyPages[NumPhysPages];

Experiment result:

Incorrect result



Correct result



(Optional)Design your own test program:

In this part, I didn't know that an modification of the Makefile inside the test file is needed when creating a new test.c in test file. Thus, an error happened when I rebuild nachos. After checking all the files inside test file and modifying the Makefile, the problem is solved and the test file run without error on code1. However, when changing the int n to double n and divided n by a value(code2) or run with over 3 test programs, unexpected user mode exceptions happened(result3). They may be caused by using the double type on n-- for former and not enough physical pages for

allocating for latter, while the causation is still unsure for me. Modified Makefile, results(result1 & result2 & result3), and code(code1 & code2) are as below.

Modified Makefile:

```
CFLAGS = -G 0 -c $(INCDIR)

all: halt shell matmult sort test1 test2 test3 test4
```

```
test3: test3.o start.o
	$(LD) $(LDFLAGS) start.o test3.o -o test3.coff
	../bin/coff2noff test3.coff test3

test4: test4.o start.o
	$(LD) $(LDFLAGS) start.o test4.o -o test4.coff
	../bin/coff2noff test4.coff test4
```

Code1(test3):

```
#include "syscall.h"
main()
	{
		int	n;
		for (n=35;n>15;n--)
			PrintInt(n);
	}
```

Result1:

```
mi@mi-VirtualBox: ~/nachos-4.0/code/userprog
make[1]: Leaving directory `/home/mi/nachos-4.0/code/bin'
cd test; make all
make[1]: Entering directory `/home/mi/nachos-4.0/code/test'
/usr/local/nachos/decstation-ultrix/bin/ld -T script -N start.o test3.o -o test3
.coff
../bin/coff2noff test3.coff test3
numsections 3
Loading 3 sections:
	".text", filepos 0xd0, mempos 0x0, size 0x150
	".data", filepos 0x220, mempos 0x150, size 0x0
	".bss", filepos 0x0, mempos 0x150, size 0x0
make[1]: Leaving directory `/home/mi/nachos-4.0/code/test'
mi@mi-VirtualBox:~/nachos-4.0/code$ cd userprog
mi@mi-VirtualBox:~/nachos-4.0/code/userprog$ ./nachos -e ../test/test3
Total threads number is 1
Thread ../test/test3 is executing.
Print integer:35
Print integer:34
Print integer:33
Print integer:32
Print integer:31
Print integer:30
Print integer:29
Print integer:28
Print integer:27
Print integer:26
Print integer:25
Print integer:24
Print integer:23
Print integer:22
Print integer:21
Print integer:20
Print integer:19
Print integer:18
Print integer:17
Print integer:16
return value:0
No threads ready or runnable, and no pending interrupts.
Assuming the program completed.
Machine halting!
```

Code2(test4):

```
#include "syscall.h"
main()
        {
                double  n;
                for (n=35;n>15;n--)
                        PrintInt(n/17);
        }
```

Result2:

```
make[1]: Leaving directory `/home/mi/nachos-4.0/code/test'
mi@mi-VirtualBox:~/nachos-4.0/code$ cd userprog
mi@mi-VirtualBox:~/nachos-4.0/code/userprog$ ./nachos -e ../test/test4
Total threads number is 1
Thread ../test/test4 is executing.
Unexpected user mode exception7
Assertion failed: line 91 file ../userprog/exception.cc
Aborted (core dumped)
mi@mi-VirtualBox:~/nachos-4.0/code/userprog$
```

Result3:

```
mi@mi-VirtualBox: ~/nachos-4.0/code/userprog
Machine halting!

Ticks: total 500, idle 64, system 70, user 366
Disk I/O: reads 0, writes 0
Console I/O: reads 0, writes 0
Paging: faults 0
Network I/O: packets received 0, sent 0
mi@mi-VirtualBox:~/nachos-4.0/code/userprog$ ./nachos -e ../test/test1 -e ../tes
t/test2 -e ../test/test3
Total threads number is 3
Thread ../test/test1 is executing.
Thread ../test/test2 is executing.
Thread ../test/test3 is executing.
Print integer:9
Print integer:8
Print integer:20
Print integer:21
Print integer:22
Print integer:23
Print integer:24
Unexpected user mode exception4
Assertion failed: line 91 file ../userprog/exception.cc
Aborted (core dumped)
mi@mi-VirtualBox:~/nachos-4.0/code/userprog$
```