EECS 211
Advanced System Software
Winter 2005

# Assignment 3

**Posted:** February 3, 2005
**Due:** February 20, 2005

**Topic:** Multi-programming in Nachos

**Instructions:**

The goal of this third assignment is to develop, implement and test support for multi-programming in the Nachos system. This assignment mostly follows the "Nachos Assignment 2" described in the file **`doc/userprog.ps`** of the Nachos installation. The instructions below assume that you read **`doc/userprog.ps`** in parallel.

**Task 1: Understand the given framework**
Go into the **`userprog`** directory. Run the given program **`nachos`** with the given user-program **`../test/halt`** to test the given code. Trace the execution path by reading through the given sources. Use also the built-in debugging facilities and the debugger **`gdb`** to run the program step by step. Make sure you understand what is going on when the user program is loaded, executes, and issues a system call.

Next, read through the sources provided in the **`userprog`** directory, as well as the ones in other directories (e.g. **`machine`**) that are listed in the **`doc/userprog.ps`** document. Note that, for this assignment, you should only need to change files in the **`userprog`** directory. All other files should be left unmodified.

**Task 2: Implement the missing system calls and exception handling**
See item 1 in **`doc/userprog.ps`**.
Modify and complete the code in file **`exception.cc`** to support the system calls listed in **`syscall.h`** and the exception types listed in **`../machine/machine.h`**. As indicated in the Nachos instructions, you will also need to implement a synchronous console class, **`synchconsole.cc`** and **`synchconsole.h`**, (which is similar to the **`SynchDisk`** class provided in the **`filesys`** directory).

To test your system calls, create a Nachos user program **`reverse.c`** that reads strings from an input file (stdin) and outputs them in reverse spelling (stdout), i.e. a file with two names **`Jane`** and **`John`** would output **`enaJ`** and **`nhoJ`**. The program should exit if **`q`** is entered. To demonstrate that this program (and the system calls in the kernel) is working correctly, provide a file **`reverse.log`** with a log of the program being run in Nachos.

Also, to show that your operating system is “bullet-proof”, write a user program **`bullet.c`** which tries to access memory outside its allocated space. When run under Nachos, it should be properly terminated before accessing any memory outside its bounds. Provide a log file **`bullet.log`** for this demonstration.

**Task 3: Implement multi-programming with time-slicing**
See item 2 in **`doc/userprog.ps`**.
Extend the given code to allow for multiple user-programs being run concurrently. You will need to modify/extend the file **`addrspace.cc`** (and possibly others). To test your implementation of multi-programming, run the **`reverse`** and **`bullet`** programs implemented earlier through the **`../test/shell.c`** shell interface. To demonstrate that this is all working correctly, provide a file **`shell.log`** with a log of both programs being run in the Nachos shell. Note that the **`bullet`** program should be terminated for its illegal access, but not the shell!

Note that for EECS 211 we will not do the part of the assignment that covers response time measuring and analysis (ignore the second paragraph of item 2).

**Task 4: Extra credit: Pass arguments through the exec system call**
See items 3 and 4 in **`doc/userprog.ps`**.
To demonstrate this task, modify your **`reverse`** program such that it takes two arguments, namely an input and an output filename. Also, extend the provided shell to separate arguments from the program name in the command line, and pass the arguments through the extended exec system call to the program.
Demonstrate this working by submitting a log file **`reverse2.log`**.

**Deliverables:**

1. ASCII text file **`assignment3.txt`** explaining your experiments; this should be no longer than one page (of paper, or Unix memory)
2. Files **`exception.cc`**, **`synchconsole.h`**, **`syncconsole.cc,`** **`reverse.c`**, **`reverse.log`**, **`bullet.c`**, **`bullet.log`**
3. Files **`addrspace.h`**, **`addrspace.h`**, **`shell.log`**
4. File **`reverse2.log`**

We will use the same electronic submission procedure as in the previous assignment. Please refer to the instructions listed with Assignment 2 for details.

--
Rainer Doemer (ET 444C, x4-9007, doemer@uci.edu)