EECS 211
Advanced System Software
Winter 2011

# Assignment 4

**Posted:** February 24, 2011
**Due:** March 2, 2011 at 2pm

**Topic:** User programs in Nachos

**Instructions:**

The goal of this assignment is to implement a set of user programs that can execute on a Nachos kernel (and test it!). For simplicity, we will use only very simple user programs that make only a few basic system calls to the Nachos kernel.

This assignment follows Task 1 of "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.

**Preparation: Understand the given framework**

Go into the **`test`** directory. Run the provided user programs on the provided (incomplete) kernel in the **`userprog`** directory. For example, run the provided **`halt`** program (the only one that will work with the unmodified kernel) as follows:
**`../userprog/nachos –x halt`**

In order to allow you to create and run your own user programs, the instructor has prepared a working Nachos kernel that supports the basic system calls that we will use in this assignment. While in your **`test`** directory, you can copy this prepared Nachos kernel to your test directory as follows:
**`cp ~doemer/eecs211/nachos.W11kernelA4 nachos`**
You can then execute the provided (or your own) user programs with the prepared Nachos kernel, as follows:
**`./nachos –x halt`**

The prepared Nachos kernel contains extra debugging support messages that you can enable with the **`–d X`** option. For example,
**`./nachos –d X –x halt`**
will indicate that the shutdown was actually initiated by the user program.

Note that you will only need to change files in the **`test`** directory for this assignment. All other files should be left unmodified!

We will limit this assignment to use only very basic system calls. Specifically, your user programs should rely only on the following 7 system calls being supported in the kernel:

(a) SC_Halt
(b) SC_Exit
(c) SC_Create
(d) SC_Open
(e) SC_Read
(f) SC_Write
(g) SC_Close

For the file I/O system calls (c) through (g), input from the console (**`OpenFileId ConsoleInput`**, alias **`stdin`**), output to the console (**`OpenFileId ConsoleOutput`**, alias **`stdout`**), and input and output to up to two regular files (**`OpenFileId > 1`**) is supported. In total, we will limit every user program to a maximum of 5 open files per user program.

**Task: Implement and test 6 simple user programs**

We will create a set of simple Nachos user programs (as test cases for the kernel in the following Assignment 5) and run them (for now) on the prepared kernel. To get started, you may want to take a look at the few examples that are already provided in the **`test`** directory.

Your own user programs should be the following:

(a) Program **`HelloWorld.c`**:
should print the famous string “Hello World!” to the console and then cleanly exit
(b) Program **`Reverse.c`**:
should let the user enter some text string on the console (e.g. “This is a test”) and then print the string backwards to the console (e.g. “tset a si sihT”)
(c) Program **`ListFile.c`**:
should ask the user for a file name and print the contents of that file to the console

All these “good” test programs should run fine and exit cleanly.

Since you will implement your own kernel in the following Assignment 5, we will also need negative test cases that check whether or not a kernel is “bullet-proof”. For this purpose, create and run the following “bad” examples:

(d) Program **`MemError.c`**:
attempts to store the word 42 into the invalid memory address 1

(e) Program **`FileError.c`**:
attempts to write data to a file that has already been closed
(f) Program **`IOError.c`**:
attempts to read a string from the standard output stream

Obviously, all these "bad" test programs will hopefully be killed by the OS before they do any damage.

**Deliverables:**

a) Six test programs **`HelloWorld.c`**, **`Reverse.c`**, **`ListFile.c`**, **`MemError.c`**, **`FileError.c`**, and **`IOError.c`**

b) Six log files **`HelloWorld.log`**, **`Reverse.log`**, **`ListFile.log`**, **`MemError.log`**, **`FileError.log`**, and **`IOError.log`** that show the running of your user programs

c) A brief description (in the body of your email!) of your implementation

**Submission instructions:**

To submit your homework, send an email with subject "EECS211 HW4" to the course instructor at doemer@uci.edu. Please include the files listed above as attachments, and put your brief (!) description in the body of your email.

To ensure proper credit, be sure to send your email before the deadline:
March 2, 2011, 2pm (sharp!).

--
Rainer Doemer (EH 3217, x4-9007, doemer@uci.edu)