CS350 Operating Systems Spring 2023

Assignment 0

1 Introduction

The objective of this assignment is for you to familiarize yourself with OS/161 and Sys/161.

OS/161: OS/161 is a simple operating system kernel, which is made available to you along with a small set of user-level libraries and programs that can be used for testing. The baseline OS/161 that we distribute to you has very limited functionality. Each of the CS350 programming assignments will ask you to improve OS/161 in some way to add additional functionality to the baseline.

Sys/161: Sys/161 is a machine simulator. It emulates the physical hardware on which OS/161 runs. Apart from floating point support and certain issues relating to cache management, it provides an accurate emulation of a server with a MIPS R3000 processor. You will use Sys/161 each time you want to run OS/161. However, you are neither expected nor permitted to make any changes to Sys/161. Additional information about the Sys/161 simulator can be found at https://student.cs.uwaterloo.ca/~cs350/common/sys161manual/.

Before you can complete this assignment, you will need to obtain and build a copy of OS/161. Sys/161 and the toolchain needed to build and debug OS/161 and its applications are pre-installed and ready to use in the linux.student.cs environment. If you are planning to work on your own machine, rather than in the linux.student.cs environment, you will also need to obtain and install both Sys/161 and the toolchain before you will be able to build, run, or debug OS/161 code.

2 Revision Control

The OS/161 kernel alone consists of more than 20,000 lines of C code. It is non-trivial to build and manage OS/161. You will find that revision control will help you manage your OS/161 code and enable tracking changes and reverting code efficiently.

In this course, we encourage you to use Git for revision control. The course web page (see "Assignment Information") includes references to links to various resources on Git. We have a tutorial on Git, https://student.cs.uwaterloo.ca/~cs350/common/gitprimer/gitprimer_F17.pdf. Git is available on the linux student environment and as University of Waterloo students you have access to enterprise edition of gitlab at https://gitlab.uwaterloo.ca/users/sign_in.

3 Setting up OS/161 and Sys/161

Before you start working on CS350 programming assignments, you must complete the following steps:

1. Install a copy of OS/161 in your linux.student.cs account using the step-by-step instructions at https://www.student.cs.uwaterloo.ca/~cs350/common/Install161.html.
   For those working on their own machines, there are instructions at https://www.student.cs.uwaterloo.ca/~cs350/common/Install161NonCS.html.
   However, there is limited support from CS350 staff for local machines.
2. Once you have installed OS/161, you will need to learn how to modify it, build it, run it, and debug it. There is a detailed guide at https://www.student.cs.uwaterloo.ca/~cs350/common/WorkingWith161.html.
   Read and understand this *before* you start working on this assignment.
3. Place OS/161 source code under revision control using Git. Use Git to place the fresh copy of OS/161

in the linux student environment under revision control. See the tutorial mentioned in Section 2.

4 Assignment Requirements

For the assignment, you are required to make two minor changes to the OS/161 kernel:

1. Customize the OS/161 kernel boot output
2. Add a new command to OS/161 kernel menu

The following subsections will be your guide for understanding and completing the OS/161 kernel programming component. You are encouraged to read the code that is discussed in these subsections to begin to understand how OS/161 works.

4.1 Customize the OS/161 Kernel Boot Output

When OS/161 boots, it produces output that looks similar to the following:

```
sys161: System/161 release 1.99.06, compiled Sep 9 2013 23:13:03

OS/161 base system version 1.99.05
Copyright (c) 2000, 2001, 2002, 2003, 2004, 2005, 2008, 2009
   President and Fellows of Harvard College. All rights reserved.

Put-your-group-name-here's system version 0 (ASST0 #17)

4916k physical memory available
Device probe...
lamebus0 (system main bus)
emu0 at lamebus0
ltrace0 at lamebus0
ltimer0 at lamebus0
beep0 at ltimer0
rtclock0 at ltimer0
lrandom0 at lamebus0
random0 at lrandom0
lhd0 at lamebus0
lhd1 at lamebus0
lser0 at lamebus0
con0 at lser0


cpu0: MIPS r3000
OS/161 kernel [? for menu]:
```

Note the line that says “Put-your-group-name-here's system …”. Your first task is to change OS/161 so that the kernel identifies itself as your kernel when it boots. For example, if your name was Liberty Valance, your kernel should say “Liberty Valance's system …”.

Once you have done this change, make sure that you can re-build and run OS/161 with your customized boot output.

Hint: You should read the file kern/startup/main.c. In the future, you can search for relevant strings using grep in a specific os161-1.99 directory or subdirectory.

4.2 Add a Kernel Menu Command in OS/161

The OS/161 kernel includes a simple system that allows debugging messages to be displayed when the kernel runs. There can be different types of debug messages, and the kernel can be told to display only messages of certain types. For example, the file kern/thread/thread.c includes the statement

```
DEBUG(DB_THREADS,"Forking thread: %s\n",name);
```

This defines a debugging message of type DB_THREADS.

The debugging mechanism is implemented in the file kern/include/lib.h. This file also includes definitions of all of the pre-defined debugging message types, such as DB_THREADS. There is a kernel global variable, dbflags, which defines which types of debugging messages should be displayed when the kernel runs (see kern/lib/kprintf.c). In the baseline code, dbflags is set to zero, meaning that no debugging messages are displayed.

After the OS/161 kernel boots, it displays a prompt and waits for an input:

```
OS/161 kernel [? for menu]:
```

If you type ?, you should get a list of available commands and sub-menus, one of which is the operations sub-menu. For this assignment, you are required to add a new command to OS/161 kernel's operations sub-menu. The new command, which **must** be called dth, should enable the output of debugging messages of type DB_THREADS. If such messages are already enabled, the command should have no effect. Thus, any kernel commands that are run after dth should run with DB_THREADS debugging messages enabled.

To do this modification, you will need to understand how the debug message mechanism works and how the kernel menu system works. The latter is implemented in the file kern/startup/menu.c. You should be able to complete this assignment by changing only this single file.

To test your new kernel option, we will use it to run one or more of the kernel's built-in thread tests with DB_THREADS debugging enabled using your new dth command. The kernel has several simple thread tests (e.g., tt1, tt2, tt3) that can be run from the kernel menu prompt.

For example, without DB_THREADS debugging enabled, thread test 2 (tt2) produces output like this:

```
OS/161 kernel [? for menu]: tt2

Starting thread test 2...
0123456701235674
Thread test 2 done.
Operation took 0.662769000 seconds
```

However, if DB_THREADS debugging has been enabled by running your new dth command, this test should instead produce output similar to this:

```
OS/161 kernel [? for menu]: tt2

Starting thread test 2...
Forking thread: threadtest0


FOorking t0hread: threadtest1
F1orking t1hread: threadtest2
F2orking t2hread: threadtest3
F3orking th3read: threadtest4
F4orking th4read: threadtest5
F5orking t5hread: threadtest6
```

```
F6orking  t6hread:  threadtest7
77
Thread test 2 done.
Operation took 0.717793640 seconds
```

The debug messages are produced by the `DEBUG` statements in `kern/thread/thread.c`.

5 Submitting Your Assignment and Checking Your Grade

Once the submission system is set up and able to accept submissions, we will create a pinned post in Piazza describing how to submit your assignments, check your grade, and use slip days.

Note: we strongly discourage you from using slip days for Assignment 0. Save them for future assignments. If you are trying an elaborate set-up and it is not working consistently, then you are strongly encouraged to use the default set-up described on the course webpage, where you ssh into the undergrad environment to edit and submit your code.