CS350 Operating Systems Fall 2022

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 30,000 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 Preliminaries

There are two options for setting up OS/161 and Sys/161. The first uses Docker containers and the second does not. You can use either approach.

3.1 Using Docker Containers

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. However, students often find that working in the student environment is cumbersome and inconvenient. We have created a Docker container that provides Sys/161 and the toolchain required to build, run, debug and test your programs.

Install a copy of OS/161 in your linux.student.cs account. Using the step-by-step installation instructions at https://www.student.cs.uwaterloo.ca/~cs350/common/Install161.html

Build and run OS/161 in your linux.student.cs account. Once you have installed OS/161, can use the

guide at https://www.student.cs.uwaterloo.ca/~cs350/common/WorkingWith161.html to learn how to modify, build, run, and debug OS/161 code. Read and understand this *before* you start working on the programming assignment.

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. That is, initialize the os161-1.99 directory with git. You will find the The Git Primer Slides mentioned in the previous section helpful for setting up Git specifically for the CS350 assignments.

Create remote GitLab repository for your OS/161 source code. You should host your CS350 programming assignments on a **private** remote repository on GitLab. You can use this repository to **push** updates from your local environment to the remote repository and **pull** the updates from the remote repository into your account in the linux.student.cs environment, or vice versa. Refer to the Git Primer Slides for the instructions.

Install cs350-container on your local machine. We have a Docker container for CS350, which includes Sys/161 and the toolchain required to build, run, or debug OS/161 code. The cs350-container is available as a public repository on UW GitLab. You should clone the cs350-container repository and follow the instructions in the README.md file to install the cs350-container and run code through the cs350-container. You will find that the cs350-container gives you a similar environment to the linux student environment with the convenience of working on your local machine. You are encouraged, not required, to learn about Docker.

Once you have completed these steps, you will have setup the environment for working on the programming assignments for CS350. You can now continue to work on the rest of the components of this pogramming assignment and test them locally using the testing and evaluation scriptsin the cs350-container.

To submit your programming assignments, you can push updates from your local environment to the remote repository on Gitlab and pull the updates into your linux.student.cs environment from the same remote repository. You must submit your programming assignments from the linux.student.cs environment using the cs350_submit script.The cs350_submit command and submission instructions are described in detail in Section 5.

3.2 Without Docker Containers

Not all systems can use Docker containers (for example, version of Windows prior to Window 10). Without Docker on your local machine you cannot use the cs350-container. However, you are still strongly encouraged to use Git for version control and Gitlab for hosting your programming assignments, so that you can work conveniently on your local environment.

In this case, **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 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 tutoral mentioned in Section 2.

4 Assignment Requirements

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

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, 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, your 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, 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]: tt2Starting
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]: tt2Starting
thread test 2...
Forking thread: threadtest0


F0orking t0hread: threadtest1
F1orking t1hread: threadtest2
F2orking t2hread: threadtest3
F3orking th3read: threadtest4
F4orking th4read: threadtest5
F5orking t5hread: threadtest6
F6orking t6hread: threadtest777
Thread test 2 done.
Operation took 0.717793640 seconds
```

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

5 Testing Your Programming Assignment

The expected flow of control when working on an assignment in CS350 is as follows:

1. You are working on your local machine and using `cs350-container` to build, run, test and debugOS/161 and linux programs.
2. You push updates to a remote repository
3. You pull updates from remote repository into your linux.student.cs environment.
4. You submit your programming assignments from the linux.student.cs environment.

You can test your code and verify that is works correctly before you submit it for grading in the `cs350-container`, we have provided the `/assignments` folder that will hold all the public testing and evaluation scripts used for each assignment. You can use these scripts to run and verify that your code works as expected before you submit it. Instructions about the script are found in the `README.md` file in the `cs350-container` repository.

We are running auto-grading scripts using public and private tests, so once you submit your code, you will receive feedback and scores for the different components of the assignment. You can submit multiple times, however, each submission completely replaces any previous submissions that you have made for the same assignment.

You are encouraged to submit early and often. Nearing a deadline, the submit server can become busyor at worst be unavailable due to technical issues.

6 Submitting Your Assignment

To submit your work, **you must use the `cs350_submit` command** in the linux.student.cs computing environment (i.e. not `submit`).

The usage is as follows

```
% usage: cs350_submit <assign_dir> <assign_num_type>
```

For Assignment 0:

- the `assign_dir` is the path to your `os161-1.99` folder, and the
- `assign_num_type` is `ASST0`.

When you execute the `cs350_submit` command, you should see output that looks something like this:

```
% cs350_submit cs350-student/cs350-os161/os161-1.99/ ASST0
Please wait as we grade your assignment
Section Name                Marks Received    Out Of          Comments
Added Name                  0.0               2               Name not changed
Added Kernel Menu           0.0               3               dth command not found or failed
A cumulative log can be found here -  ASST0.log
We will always take the highest grade from all submissions as the final grade
Note: All submissions are stored and persisted and checked for plagiarism after the due date
```

The argument `assign_dir` in the `cs350_submit` command, packages up your OS/161 kernel code and submits it to the course account using the regular `submit` command. This assignment only briefly summarizes what `cs350_submit` does. You can (and should) learn more on-line by reading https://www.student.cs.uwaterloo.ca/~cs350/common/SubmitAndCheck.html. Look carefully at the output from `cs350_submit`. It is a good idea to run the `cs350_submit` command like this:

```
cs350_submit cs350-student/a0 ASST0 | tee submitlog.txt
```

This format will run the command and also save a copy of all of the output into a file called `submitlog.txt`, which you can inspect if there are problems. This is handy when there is more than a screen full of output.

You may submit multiple times. **Each submission completely replaces any previous submissions that you may have made for this assignment.**

7 Checking Your Grade

You can run the following command from your linux student environment:

```
cs350_grade <assign_num_type>
```

Where the `assign_num_type` is the same `assign_num_type`, you used when submitting your assignment. For example, for checking your grades for Assignment 0, the command and output should be similar to:

```
% cs350_grade ASST0
Grade for student - kzillehu - and assignment - ASST0: 0.00 at Sun Jan 16 20:42:16 2022
```