COMP3231/9201/3891/9283 Operating Systems 2019/T1

UNSW

Administration
- Notices
- Course Outline
- UNSW Timetable
- Consultations
- Group Nomination
- Survey Results!!

Work
- Lectures
- Tutorials
- Extended Lectures

Support
- Piazza Forums
- Wiki

Assignments
- Assignment 0 Warm-up
- Assignment 1
- Assignment 2
- Assignment 3

Resources
OS/161
- General
- Man Pages
- Sys161 Pages
C coding
- Info Sheet
Debugging
- GDB and OS/161
General
- "Hardware" Guide
- R3000 Reference Manual
- Intro. to Prog. Threads

Previous years
- 2018 S1
- 2017 S1
- 2016 S1
- 2015 S1
- 2014 S1
- 2013 S1
- 2012 S1
- 2011 S1
- 2010 S1
- 2009 S1
- 2008 S1
- 2007 S1
- 2006 S1
- 2005 S2
- 2005 S1
- 2004 S2
- 2004 S1

The OS/161 System

Background

OS/161 is an educational operating system developed by the Systems Research at Harvard group, at Harvard University. It aims to strike a balance between giving students experience working on a real operating system, and potentially overwhelming students with the complexity that exists in a fully fledged operating system, such as Linux. Compared to most deployed operating systems, OS/161 is quite small (approximately 20,000 lines of code), and therefore it is much easier to develop an understanding of the entire code base.

The complete OS/161 environment consists of three major parts:

- OS/161 itself;
- System/161, the simulated machine that OS/161 runs on;
- development tools required to compile and debug code for the simulated machine.

In your assignments, you will be working exclusively on the OS/161 operating-system source code.

System/161 and the development tools are already installed on CSE machines — you will not need to modify nor understand their source code.

OS/161

The OS/161 source code distribution contains a full operating system source tree, including the kernel, libraries, various utilities (`ls`, `cat`, etc.), and some test programs. OS/161 boots on the simulated machine in the same manner as a real system might boot on real hardware.

The assignment specifications will contain instructions on how to get a copy of the sources for each particular assignment.

Warning: Only use OS/161 sources obtained from the assignment specifications.
We modify some parts of the environment for local use. Hence, there are no guarantees that any local assignments will be achievable or *assessed* correctly if you use components sourced from elsewhere.

System/161

System/161 simulates a "real" machine to run OS/161 on. The machine features a MIPS R2000/R3000 CPU including an MMU, but no floating point unit or cache. The number of CPUs available in the machine is configurable, but we will only be using a uniprocessor configuration at UNSW. It also features simplified hardware devices hooked up to *lamebus*. These devices are much simpler than real hardware, and thus make it feasible for you to get your hands dirty, without having to deal with the typical level of complexity of physical hardware.

Using a simulator has several advantages. Unlike software you have written thus far, buggy software may result in completely locking up the machine, making to difficult to debug and requiring a reboot. A simulator allows debuggers access to the machine below the software architecture level as if debugging was built into the CPU chip. In some senses, the simulator is similar to an *in circuit emulator* (ICE) that you might find in industry, only it's done in software. The other major advantage is speed of reboot. Rebooting real hardware takes minutes, and hence the development cycle can be frustratingly slow on real hardware.

Development tools

The OS/161 environment uses modified versions of the usual GNU development tools (compiler, linker, etc.), named with the `os161-` prefix: `os161-gcc`, `os161-gdb`, `os161-ld`, etc. The OS/161 build system hides most of the complexity of using these tools directly. However, you will still need to learn to use GDB.

You should already be familiar the GDB, the GNU debugger. GDB allows you to set *breakpoints* to stop your program under certain conditions, inspect the state of your program when it stops, modify its state, and continue where it left off. It is a powerful aid to the debugging process that is worth investing the time needed to learn it. GDB allows you to quickly find bugs that are very difficult to find with the typical `printf` style debugging.

Our **GDB tutorial** is a brief and focused introduction to using GDB with OS/161.

Details **beyond** the level you need to know can be found at http://www.gnu.org/software/gdb/gdb.html.

Working from home

See the wiki for instructions for installing System/161 and the development toolchain on non-CSE machines. Feel free to build and install them at home. However, while we make every endeavour to ensure the tools build and function for you, we cannot be responsible for them working (or not) on your machine.

We absolutely will NOT take "It worked for me at home" as an excuse for submissions not working as expected during our automated assessment process. If you are not confident of your submission working at CSE, ensure that your submission correctly builds and functions on CSE machines!

Page last modified: 12:56pm on Tuesday, 23rd of February, 2016

Print Version