CMSI 387/587

OPERATING SYSTEMS

Spring 2010

Assignment 0304

This assignment is all about getting your hands dirty with building and deploying an operating system kernel.

Not for Submission

1. You'll need the unified diff (*unidiff*) format for this assignment — look it up, and learn how to use *diff* to generate such a file. To round things out, read about *patch* so that you can get an idea of how I'll be using your *unidiff* file.
2. *Understanding the Linux Kernel* may be of particular help for this assignment.

For Submission

Perform the kernel building and modification project described at the end of SGG Chapter 2.

- To be uniform (and for your convenience), the kernel source archive that you should use has been downloaded for you. You'll find it at *http://www.cs.lmu.edu/~dondi/os-images* (there may be more than one version; just grab the latest).
- Commit your work to CVS in the form of a *unidiff* patch and a sample program.

NOTE: Remember that open source software changes faster than textbooks do — accomplishing this work is guaranteed to *not* be exactly as specified in the book. Note, however, that while the specific steps (the "policy") may change, the overall process (the "mechanism") does not. Enjoy!

User-Mode Linux (UML) Option

Testing your kernel via the UML virtual machine allows you to do your work on a Keck lab workstation without having to overwrite the installed OS.

- In addition to what was discussed in class, most of the information that you'll need is available on the UML home page, located at *http://user-mode-linux.sourceforge.net*.
- For your convenience, bootable disk images have been downloaded for you; you can boot UML directly off these files. You will also find them under *http://www.cs.lmu.edu/~dondi/os-images*. The latest versions of these disk images, and more, can be found at *http://uml.nagafix.co.uk*.
- **IMPORTANT:** When booting off a disk image, you'll need a mechanism for storing your local changes. This is done using a *copy-on-write* (COW) layer, which is a separate file that you specify. Thus, your invocation of UML should look something like this:

  path-to-your-UML/linux devfs=mount mem=32M ubda=path-to-your-COW-file,path-to-your-disk-image

  This will boot UML within a virtual machine with 32M of RAM and a disk consisting of your disk image, with the given COW file to capture any changes that you make to the virtual disk.
- You'll need to get files into the virtual machine (i.e., your test program that demonstrates the system call that you added to the kernel). Of all the available options, *hostfs* is the most convenient. Look up how to use this option within your UML virtual machine.

Workstation Option

If you'd like to boot your kernel "for real" from a real machine, you may borrow a "mule" workstation that you can wipe-and-install as needed. Talk to me if this is how you'd like to work.

How to Turn it In

Commit your work under */homework/cmsi387/hellokernel* or */homework/cmsi587/hellokernel*, as applicable. The deliverables consist of two files:

1. A unidiff-format patch file that applies your changes to the original kernel source tree, named *your-last-name-hellokernel.patch*. Before creating the patch file, be sure to *make mrproper* with the appropriate architecture in order to eliminate as many derived artifacts as possible.
2. The source code to a sample program that demonstrates your new system call. Of course, this program should only work correctly when executed under your modified kernel.