

CMSI 387

OPERATING SYSTEMS

Spring 2007

Project Submission Guidelines

As you prepare the finishing touches on your kernel projects, you should now have a great appreciation for when something is well-packaged and well-documented. This can make all the difference between smooth sailing on the project and being stuck in a rut for a long time. For this reason, these guidelines are being given out so that your own project can be installed and used as quickly as possible (and to keep *me* from being stuck in my own rut when looking at your projects!).

Key Artifacts

Your `/projects/cmsi387` CVS repository constitutes the project submission. As a bare minimum, this repository must contain:

- The `readme.txt` file that you first committed for Assignment 0301 — Edit this file in place so that it contains the information listed to the right.
- The *source code* for your project — You only need to commit the code that you actually wrote; if the code needs to be built with the rest of the kernel, then your `readme.txt` file should contain instructions on where these files should be within the kernel source tree.

Brownie points for you if you include any of the following items *in addition to* your raw source:

- A *unidiff patch file* that automatically applies your changes to a pristine kernel source tree (as you were asked to do in Assignment 0125).
 - A *makefile* that automates the compilation and installation of your project.
- Your source code should consist of two primary components; make sure to indicate them clearly:
- The kernel change or extension itself
 - Test programs or scripts that help illustrate the change that you made
- Any other artifacts/files/libraries that are needed to build, run, and test your project. For items created by third parties, provide information on how to acquire them (i.e., download link, `apt-get` instructions, etc.).

Key Information for `readme.txt`

Your final, committed `readme.txt` file must contain at least the following information:

- *A concise description of your project* — What it is, what it does, and how it does it.
 - *Requirements/prerequisites* — A list of what is needed in order to get your project up and running (in addition to a pristine 2.6.19.2 kernel source tree and a bootable disk image). The general expectation is that your project can be run from a UML build that integrates your work, booted off a Debian 3.1 disk image; if it's anything different (i.e., additional hardware required, boot CD, different kernel version, etc.), you should document it clearly.
 - *Build instructions* — How to compile your project based on what is committed to the repository; include any instructions such as where files should be placed within a kernel source tree (if needed), how to apply your unidiff patch (if available), and how to install any additional or nonstandard libraries.
 - *Testing/demonstration instructions* — Steps that clearly demonstrate the change that you made. Describe the expected behavior on an unmodified/unextended kernel when following those steps, then describe the new behavior if those same steps are performed once your project has been booted, and/or installed.
- Be as direct and unambiguous as possible with these instructions; come up with something that makes it *absolutely clear* that your change is having the desired effect.