# Information on the Mac Mini Continuous Build Machines

4/28/11

## Machine Names and Setups

There are five Mac minis currently set up to do continuous builds of the OpenStudio project. They are all connected to one monitor and keyboard via a KVM switch. The computer names, OS, and KVM port are as follows:

OS – KVM port – Machine name

Mac OS-X 10.6 – port 1 – nlong-18786s

Fedora Core 14 – port 2 - nlong-18783s

Ubuntu 10.04 – port 3 – nlong-18787s

Windows 7 – port 4 – nlong-18782s

Windows XP – port 5 – nlong18785s

These machines are each individually plugged into the NREL local network using the LAN ports at their office location and the one next door. If there is ever a power outage, KVM switch will reset and you will need to plug the keyboard into the switch’s keyboard dongle in order to get past the setup screen. There is no username or password for the KVM switch.

You can log into the Windows machines using your NREL username and password. The Linux machines and the Mac have a common username and password. The username is: nreladmin. The nreladmin user has administrative privileges on Linux and Mac, so that you can make changes to the system.

### Compilers and Tools

Each of the machines has been set up using the instructions on the developer portion of the OpenStudio website. Following the links on the left side of <http://openstudio.nrel.gov/developers> should lead to instructions for each platform. It is important that when setting up development machines, the instructions are followed closely, or problems may arise. The windows machines in particular seem to be especially tricky when setting up for development for the first time, and if the order of installations is not followed, key files may be overwritten. One consequence is that it will be important to determine a set of instructions for using Visual Studio 2010 in the future, as we currently do not have a process for this.

## The Continuous Integration Repository

The continuous build and testing process is controlled by scripts located in the repository: <https://cbr.nrel.gov/ContinuousIntegration/svn>. This repository has three subdirectories.

### CronScripts Directory

The CronScripts directory contains shell scripts and batch files that control how the ruby build scripts are launched.

On the Linux machines the shell scripts set up environment variables that are necessary for the other scripts to have access to password storage for svn and others. If the Linux machines are restarted, it is possible for these environment variables to be incorrect because some of them depend on specific process ID’s. These shell scripts are run through a cron job that is started after midnight each night.

On the Mac I have set up the process using launchctl and launchd because Apple claims that cron will eventually be deprecated on the Mac. Launchd uses plist files to control jobs which are essentially a specific form of xml. The file gov.nrel.nightly.plist is the relevant file for the Mac. The content of this file looks very similar to an xml form of a cron job with some extra options. The plist files are set up in the form of several key, value pairs that define the properties that launchd needs to execute a process. Similar to cron on the Linux machines, the launchd task simply calls the nreladmin-OSX-10.6.sh script which in turn runs the build process.

On Windows, the batch files are fairly simple and straightforward. The line at the top of each file calls the vcvarsall.bat batch file. This file is provided in Visual Studio and sets up the command line environment so that it has access to the Windows compiler tools. The batch file is run by a task set up through Windows’ Scheduled Tasks program which is located at Start -> Programs -> Accessories -> System Tools -> Scheduled Tasks, from the Start menu.

In the case of all the machines, the shell scripts and batch files do a svn update command on the Continuous Integration repository so that any changes to the other scripts in the repository will be applied. Currently, I am updating these working copies using my own username and password, but in the future a common or generic user should be created so that this process isn’t reliant on any one user. The scripts then call ruby to run the ruby scripts in the RubyScripts directory.

### RubyScripts Directory

The RubyScripts Directory contains scripts that control the build and testing process. One of these scripts is required for each machine that the nightly and/or continuous builds will be running on. In addition there is a common.rb script that contains some function definitions that are used by all of the scripts. Some basic changes to the build process or CTest script may require that each of the individual scripts be updated. Examples might include additional variables that are configured by the scripts. The scripts use ruby’s erb package to do text substitution on the main CTest script before it is run.

The beginning part of each script configures some path information and defines some variables. The variable ‘threads’ is created and each build type is run as its own separate thread so that a couple of things may be happening in parallel. The rest of the script sets up each build type as a separate thread and then calls one of the common script functions to run that thread. The common script functions either run the build once or in the case of run\_continuous, check to see if the repository has been updated and run the build throughout the day.

There are several types of builds that can be run through the current scripts. Only the Linux machines are running all of them, because they are the only ones that have all the tools installed. In addition to the Nightly and Continuous builds, they are each running a Coverage build and a Memcheck build. Coverage uses ‘gcov’ to measure how much of the code is touched by the unit test suite. Memcheck uses ‘valgrind’ to look for memory usage and leaks. The Mac could be set up to do these builds, but the build time on the Mac seems too large to make running these checks practical. It is likely that these builds would overlap with the Continuous builds, and further increase build times.

The following is specifics about some of the variables. On all the machines the ‘submit\_package’ variable is currently turned off, but it could be used to collect the nightly package builds in a common location. The ‘win\_version’ variable is important on the windows machines to make sure the correct compiler is used, and should be set to either ‘XP’ or ‘7’. The ‘username’ variable is set to the user that will be used to check out the code from the OpenStudio repository. It is currently set to ‘cbrbuild’ and the repository location ‘https://cbr.nrel.gov/openstudio-mirror/svn/trunk’ has been set up so that there is a consistent user for all the continuous build checkouts. The ‘cbrbuild’ user has a password that has been set up by Nick Long. Unfortunately, the same type of user has not yet been set up for the ContinuousIntegration repository.

Finally, the CTestScripts directory contains the main CTest file that performs the configuration, build, testing, and submission steps. This script has the extension rcmake, because it is not ready to run as is, and must first be configured by ruby’s ERB. I have made an attempt to add additional comments to this script to hopefully clear up some of the more confusing parts. Other than that, it will be necessary to know some CMake and CTest scripting to fully understand this script. Each different type of build can be performed using this script, and the necessary steps will be performed by CTest.

## Specific Machine Considerations

The following notes describe various details pertaining to the individual setups on the build machines.

### Windows XP

The Windows XP machine has been set up with certain privileges so that it can run scheduled tasks without a user being logged into the machine. At the time of this writing, Matt Goodnow is the network admin who is in charge of these group settings. You may need to work with him if there are any problems in the future.

The Windows machines are very finicky as far as the stability of the continuous build processes are concerned. I often find that I need to reset the password in the Scheduled Tasks menu or possible delete the repository checkout and check it out again under the “cbrbuild” username to get things working again.

Both the Windows machines are building OpenStudio using the express version of Visual Studio 2008.

### Ubuntu

Because of some strangeness on the Mac Mini, the Ubuntu machine is using a custom compiled driver for its network interface. The source code for the driver is located on the Broadcom website and simple instructions for installing it are located at: <https://help.ubuntu.com/community/Macmini4-1/Lucid>

We are currently building OpenStudio on version 10.04 Lucid Lynx which is the current Long Term Support (LTS) version of Ubuntu. Another LTS version is expected to be released sometime in 2012.

### Mac OSX

The build times on the Mac are currently very large, and this is mostly due to the necessity of building Universal Binaries. The reason we are building universal binaries, meaning both 32-bit and 64-bit versions is that the default Ruby installation on the Mac is written in 64-bit while the Ruby included and used by Google Sketchup is 32-bit. Therefore, in order to use both the OpenStudio Sketchup plug-in and the system Ruby bindings the universal version must be built. It may be that it is not necessary to build universal binaries on the nightly and continuous builds in order to save some build time.