USCC
Programming Assignment 0 – Installation (Windows)

By: Sanjay Madhav
University of Southern California

Prerequisites

If at all possible, it's easier to setup a Mac to work on the USCC projects. However, Windows will also work, though there are a few more steps involved.

- Windows 7 or higher
- Visual Studio 2013 (optional – just to help edit code)
- ~5 GB free space (when built, LLVM will take about almost 4 GB)

Installing Cygwin

Building in Windows requires Cygwin. Follow these steps:

1. Download installer for Cygwin (x86): http://cygwin.com/setup-x86.exe
2. Select the default options until you get to the page asking for which packages you want to install. You need to add the following optional packages: `flex`, `make`, `gcc-g++`, `python`, `libncurses-devel`, `git`, `gdb`, and `curl`.

Installing and Building LLVM

In case it's unclear, in the instructions below `$` is not something you should type, but it just represents the command prompt.

1. Open the Cygwin terminal and `cd` to the directory you want to work in. In Cygwin, if you want to access your normal drives, it's mapped to `/cygdrive`. So `C:\dir` is accessible from `/cygdrive/c/dir`. You will be making multiple directories and links, so it's not recommended to do this in ~.
2. Download the LLVM 3.5 source:
```
$ curl -O http://llvm.org/releases/3.5.0/llvm-3.5.0.src.tar.xz
```
3. Extract the archive:
```
$ tar xJf llvm-3.5.0.src.tar.xz
```
4. Rename the directory to `llvm`:
```
$ mv llvm-3.5.0.src llvm
```
5. Enter the llvm directory:
```
$ cd llvm
```
6. Configure LLVM to build a debug build:
```
$ ./configure CC=gcc CXX=g++ --disable-optimized --enable-terminfo=no
```
7. Build LLVM (this may take over an hour depending on your machine, but you only have to do this once):
```
$ make
```

8. Once you've built LLVM, you need to make a couple of links in the parent directory, so:
```
$ cd ..
$ ln -s llvm/Debug+Asserts/lib/ lib
$ ln -s llvm/Debug+Asserts/bin/ bin
```
9. LLVM is now setup for use with USCC

Forking and Cloning the Starting Code

1. Open the following URL in your browser: https://bitbucket.org/uscc/uscc-projects
2. Fork the repository *making sure to make your fork private*. Please do not make a public fork as it will allow anyone, including other students, to see your code and potentially copy off of it. For more details, refer to the following Bitbucket documentation page: https://confluence.atlassian.com/display/BITBUCKET/Forking+a+Repository
3. Now go back to the terminal and make sure you are in the directory that contains the `llvm` directory as well as the two symbolic links to `bin` and `lib`
4. Clone your fork into a local directory called `uscc`:
```
$ git clone https://username@bitbucket.org/username/forkname.git uscc
```
(You can get the exact URL in the top right corner of your repo's page on Bitbucket. Alternatively, you can checkout using a visual git client such as SourceTree.)

Building via Command Line

1. Enter the `uscc` directory:
```
$ cd uscc
```
2. Build uscc:
```
$ make
```
3. To verify it worked, enter the tests directory:
```
$ cd tests
```
4. In the tests directory, run the following:
```
$ ../bin/uscc -a test002.usc
```

You should then get the following output:
```
test002.usc:16:1: error: Function implementation missing
{
^
1 Error(s)
```
5. Then try running the test suite with the following command:
```
$ python testParse.py
```

You should have 22 out of 23 tests fails.

Editing Code via Visual Studio

Once you verify that you can build via command line, you can open the uscc.sln file in Visual Studio 2013. You can edit your code directly in Visual Studio. The build command in Visual Studio still builds via Cygwin, but it's a shortcut to typing in the command yourself. You still will have to run uscc and the test suites via the command line, and if you need to debug you will have to debug in `gdb`.