Development Research Starter Guide

Roitberg Lab

May 15, 2017

Introduction

This guide will cover many things that we believe will be extremely helpful to anyone interested in pursuing computation chemistry deveelopment in the Roitberg Lab. It was made with the guidance and recommendations of many of the more senior students. In order to avoid reinventing the wheel, I've added links to sites I've felt gave a very good explanations to the things I've mentioned. If you run across something you don't understand, most likely the explanation can be found in one of these hyperlinked sites. The hyperlinks may not work on the github site, you may need to download the pdf first and your local pdf viewer. We will be covering the following topics:

- Unix System
- Bash commands
- Text Editors
- IDEs
- Programming Languages
- Github
- Graphing
- Electronic Structure Theory

Unix System

Before we get started, however, I'd like to give a recommendation that you use linux or OSX for your work done in this lab. Many of the programs we work with only run in unix like operating systems. So if you ever wanted to work on something on you personal computer, you may not be able to do so with the Windows OS. Also, many of the text file are incompatible between the two systems and conversions will have to be made. I recommend you use a virtual machine to test out some of the most common distros before you make a leap to dualbooting. This **site** provides a useful guide on how to do a VM install of Ubuntu. Other recommended distros would be **Opensuse**, **FEDORA**, and **Arch** for those people who like to jump right in. (These are some of the most commonly supported free distros)

A very good introduction to the unix system can be found here. If you migrate from windows, the first major change you may notice will be how the directory structure is different. The famous *C://Programs* directory no longer exists. Where then are all the programs? Linux systems follow the Filesystem Hierarchy Standard. Mac OSX uses a slightly different version. Both hierarchies can be found with the command **man hier**. Many permission problems can be avoided if you follow this hierarchy.

Now, as the unix guide given mentions, a shell is the interface between the user and the kernel. You most likely have bash installed as your default, and this guide will assume you do, though there are quite a few alternatives. Here are two of what I consider your best options. Please watch these videos before you decide. Bash very well may not be your best option, zsh probably is.

zsh

cshell

For compatibility, write mobile scripts in bash or cshell. Some High performance computers (HPCs) will have cshell as the default, you can ask them to change the default shell if you prefer.

Shell Commands

You may already know how to use many of the most common commands such as

pwd print the current directory

cd change the directory

ls list the current files and directories in the current directory

rm/cp/mv the delete, copy, and move commands

If you don't know what these things are, theres an excellent tutorial found **here** This tutorial is written for bash, but most, if not all the commands will work for the other recommend shells. However, when you begin development, or even if you simply want to install Amber, these commands will not be enough, further commands must be learnt. You may also need to make modifications to your configuration files.

The commands you should learn with links to the sites with explanations are

locate outputs the locations of a file

which locate a program file in the user's path

export adjust bash directory variables, add directories to user paths

alias create a shortcut for a command

grep search files for a string

sed find and replace words in a file

awk general string manipulation

Remote Computing

Running programs can be very computationally expensive. Often, these programs are too powerful for your personal computer. You will need to use a remote center. SSH, is a protocol for doing so. The university provides a high performance computer (hpc) for us to use. They have a guide for you to get started. Bluewaters is another center we will often use. You can also ask for access our labs local computers.

When you connect over ssh, in order to run certain programs for the server, you will need an Xserver. Linux machines come with this built in. Mac's will need to install xquartz For both Mac and Linux, when you loggin, make sure to use the -Y flag to enable Xserver.

For transferring data between systems we recommend you use a service called Globus. To create a personal endpoint you can follow these directions. Often, mounting a remote directory as a drive on your personal computer can quite useful. Directions exist for both Mac and Linux Note that I believe Ubuntu has a right click option in their directory explorer.

Programming Languages

For development work in the Roitberg Lab, you will need to know a few programming languages. Each one has its own utilities. These programming languages are

FORTRAN Amber is written in this, so of course this is important

Python Python is a fantastic tool for analysis, containing incredible libraries such as matplotlib, scipy, pandas, etc.. I also use it for text parsing, though I guess you could use grep, awk, or sed instead. As for deciding between python2 or python3. It doesn't really matter right now, 2015 (they're not that different), but python3 is the way of the future.

C/C++ These languages will be most useful if you are working with nVidias' CUDA api, which is written for C/C++. Anakin-me is written in these languages.

Further tutorials can be found at **Lynda.com**. After you've finished the tutorials, you can continue practicing on problems provided by numerous sites. The first one is a great introduction into computational chemistry problems..

1. Top Coder
2. USACO
3. VT Crawdad Problems
4. EPI
5. Project Euler
6. Kaggle

Writing Programs

Once you have decided which programming language you will use to write your project, you then need to decide which program you will use to write it. There are quite a few options. First there are two large categories of programming development tools, text editors and integrated development environments (IDE's). A discussion of the advantages-disadvantages can be found here.

The most widely used text editors are

- vim
- emacs
- sublime text
- atom

Many people in this lab use vim, which is the editor I would currently recommend. However, it does have a fairly tough learning curve. To help learn it, people have created a free tutorial as well as a fun, very effective game. Its also very customizable, and there are many plug-ins that improve the program. Emacs is the other primary contender, and is the one I generally use. People customize both emacs and vim to their liking, and you can find configuration files for these editors all throughout the Internet. A good example for a vim configuration file can be found here, where an excellent start for emacs can be found with spacemacs. I have my own config files as well, though my documentation is not as well maintained. The Editor War has been going on for ages arguing which is best. Take a look and choose the one you think is right for you.

Some of the most widely used IDE's are

- Code::Blocks
- Eclipse
- NetBeans
- VisualStudios
- Xcode
- JetBrains Suite

I've tried all of these. My favorite's lie in the JetBrain suite. Justin's is Codblocks. Care should be taken to note the supported compilation archetecture. For example, currently Clion (part of Jetbrains) doesn't support make files.

IDEs are not necessary, in fact, many developers prefer to use text editors such as VIM or Emacs for most of their work. Try a few and find something you like.

Github

You can think of Github as the Dropbox of programmers, if Dropbax allowed you create multiple versions and kept a repository of previous changes. It also allows you to merge versions. This service is extremely useful, in fact, it's so useful that OSX now incudes support for it by default. I recommend you look at this guide for a walk through tutorial. You should also check out this tutorial

In my vim customization I included a plugin that deals directly with Github. Its called vim-fugitive, and I recommend you watch their screencasts found here.

Graphing

Regardless of what you do in this lab, you will eventually need to plot some data. Here are some of the recommended programs.

gnu-plot no GUI, but can find documentation

grace hard to find documentation but contains a GUI

matplotlib python module, lots of documentation ** Favorite

Examples

example 1

Suppose you want to install **Amber**. You are given are either given the tar file from one of your group members or you download from git. You then look at the manual for directions on how to install. We will assume you have all the necessary libraries installed. (We'll cover what libraries are shortly) It tells you to extract into your installation directory and gives you */home/myname* as an example. You are unsure if this is what you want to do, so you open the FSH with **man hier**. You realize that if you're not the computer administrator this is a perfectly fine place to put it. However, you are, and according to FHS, the program should go to */opt*. So you extract into */opt*. The manual then directs you to **export AMBERHOME=/home/myname/amber17**. You quickly translate that to **export AMBERHOME=/opt/amber17**. But what does this do? Well you can think of an export as creating a shortcut to be used by the shell or anything running inside it. So instead of typing in **/opt/amber17** you can use **$AMBERHOME**. That saves some time. But its true utility lies in the ability for other programs to use these exports. If a program needs to use a file from a directory within **AMBERHOME**, it no longer needs the user to tell it where to look. It simply lets the shell redirect it.

example 2

We have just installed CUDA, nVIDIA's API for utilizing their gpus. We've installed it in the directory */usr/local/cuda*. In this CUDA directory, there are a few directories, including *bin* and *lib64*. *bin* contains the compiler that we will want to be able to call from the shell, **nvcc**. *lib64* contains a bunch of shared library files.

We try to compile with

nvcc

We receive an output

nvcc: command not found

How can this be? We just installed it. Maybe **nvcc** isn't in the path. We type the command

which nvcc

Nothing. We then open our shell config and add the line

export PATH=/usr/local/cuda/bin:$PATH

After we source our new configuration, **nvcc** returns */usr/local/cuda/bin*.

We've solved our problem. But we quickly stumble upon another. When we try to compile, we keep getting a message that some libraries aren't found. We now know how to fix this easily. We open our config file, and add the line

export LD_LIBRARY_PATH=/user/local/cuda/lib64:$LD_LIBRARY_PATH

And our problems are solve.

Latex

Latex is an editor that is very widely used in peer reviewed journals. You can think of Latex as the HTML of word processors. A quick tutorial can be found **here** Bibliographies can sometimes be difficult to work with. I've been using vim as my editor while using BibLatex with biber as a backend. To generate the pdf, I enter four commands: pdflatex biber pdflatex pdflatex

Electronic Structure Theory

For those interested in doing work in electronic structure theory, but have little knowledge of the topic, I recommend beginning with Attila Szabo's excellent book Modern Quantum Chemistry. Solutions to the example problems can be found here.