Welcome to the York Group
Version 0.1

Adam T. Moser
May 26, 2009

Contents

1 Introduction to the York Group 4

2 Research Information 4

3 Getting Started 5

  3.1 Your Computer .......................... 5
  3.2 Linux .................................. 5
  3.3 CVS - Concurrent Versions System .... 5
    3.3.1 checkout and update .............. 6
    3.3.2 add and commit .................. 8
  3.4 The Electronic Library ............... 9
    3.4.1 Searching with tkYorkLib .......... 9
    3.4.2 Adding Articles .................. 11
  3.5 Message Board ........................ 14
  3.6 Viewing LaTeX ........................ 15

4 Resources 16

  4.1 In the Lab ............................ 17
    4.1.1 Website .......................... 17
    4.1.2 Backing up your computer ......... 18
    4.1.3 Connecting your computer ........ 19
    4.1.4 Printing .......................... 20
### 8 Computer Code

8.1 Writing Code .................................................. 44  
8.1.1 Fortran ....................................................... 44  
8.1.2 Perl .......................................................... 44  
8.2 Compiling Code .................................................. 45  
8.2.1 Compilers .................................................... 45  
8.2.2 Trouble Shooting ............................................. 45

### 9 Chemistry Software

9.1 Gaussian .......................................................... 45  
9.2 AMBER .......................................................... 45  
9.3 VMD ............................................................. 45
1 Introduction to the York Group

Welcome to the York Group. This document is meant to provide useful information for an incoming graduate student, post-doc, visiting researcher, etc. I write this just as if I was talking to you, though I imagine later on I will have to fix the language to make it sound more professional. Most of what is written will be “borrowed” from other resources, so let’s hope I remember to reference everything.

I plan to include information on a lot of topics. None of it will likely be complete information, rather enough information to answer simple problems or get one started in an area. I find most people run into the same problems, and if as a group we start writing down the information, we could save ourselves a lot of time. Also, I want this document to be systematic, that is to say in the same order as you would need the information if you are just starting out with us.

This document is NOT meant to take the place of you asking questions. There is a lot going on in this lab, we use a lot of different tools, there is a lot to know. This introduction is just a place to get started and a reference as you begin here. You should still feel free to ask questions; we just want to know you have already looked around a bit yourself.

If you are reading this as a hard copy, there are many links to other documents, pictures, internet sites that have more information. Make sure that once you have learned how to get at this document electronically you check for these resources.

Often in this guide I will give examples of things that you can try. For example, if I want you to open up a terminal and type something, I will indicate it like this

```
>type stuff here
```

More about this as you need it.

2 Research Information

The York Group is involved in a variety of projects that require a lot of different tools. For more information go check the website. In a nutshell, we study catalytic RNA, ribozymes, using whatever computational techniques necessary to get at the questions we ask. These techniques include implicit solvation modeling, molecular mechanic simulations, monte carlo simulations, quantum mechanical / molecular mechanical simulations, \textit{ab initio} electronic structure calculations including hartree-fock, density functional, and higher correlation methods. Further, this group also develops methods when current software cannot do what we need. As an incoming person, you should probably be talking to Darrin about what particular project you will be working on and what techniques you will need to be familiar with.
3 Getting Started

Frankly, there are a lot of good things to know, but there are a few things you have to know to get started. You need to get a computer, know a little linux, and be able to use cvs and the electronic library.

3.1 Your Computer

First off, lets get you a computer. To get going you will need one of the lab administrators (probably me or Tim) to make you an account. This machine is going to be yours to work on, so its pretty much up to you to keep it working and customize it as you see fit. First thing you should notice about your machine is it is not running a Windows or Mac operating system. Hardware-wise the machines in the lab range from single processor pentium 3 systems to multicore duo processors. All the machines are capable of handling the tasks you will start out doing, so lets move on to how to do stuff on your computer.

3.2 Linux

The York Lab uses linux as our operating system (OS). In particular, we use the flavor of linux called Fedora Core (FC). If you want to use a different OS and know how to make it work, more power to you, but don’t expect much support from the rest of the lab if some piece of software isn’t working right. The thing about FC, and really all linux, unlike say Windows is that you have to know a lot more about your computer to do things. On the other hand, you have a lot more control on your computer once you learn about it.

With linux there is much less point-&-click and a lot more typing on the command line. If you don’t know anything about linux at all, skip down to Learning Linux. Its pretty likely you already know how to make, move, and edit files and directories using vi (5.5) or emacs (5.6), which is all we need to get started. For an excellent review of a lot of linux information, check out this user guide.

3.3 CVS - Concurrent Versions System

In this lab you will often be working on code or documents at the same time as others, therefore we use a program called Concurrent Versions System (CVS). CVS is sort of the backbone of the lab. You can read about it here, here, or here. Also, this has a great list of commands.

In a nut shell, its a program that stops the following problem: Bob is working on a paper and makes a change in the discussion section. At the same time Darrin is making
another change in the discussion section. Bob save the file then Darrin saves the file. Bob’s changes are lost. How does CVS do it? Well instead of there being one file that everyone is working on directly, CVS keeps a master copy and users just check it out. When you go to commit your work back to CVS, it checks your changes and all the other changes that might have come in by others to make one version that contains all the changes.

All master copies of all these files live on riesling, which if you don’t already know is the noisy computer in the server room in the back of the lab. How do you use CVS? Actually it is pretty easy.

We need to make sure your computer knows to look at riesling whenever you use the \texttt{cvs} command, so open up the environment profile for all users (i.e. cvs should be setup for all users).

```bash
> less /etc/profile
```

Make sure in that file the following is in there someplace

```bash
# CVS
export LOCALCVS=$HOME/CVS
export CVSHOME=$HOME/CVS
export CVSROOT=:\texttt{:ext:riesling:/usr/local/cvsroot}
export CVS_RSH=`\texttt{which ssh}`
export CVSEDITOR=`\texttt{which vim}`
```

(Note: the ` above are backticks: shift + tilde key). The above just says that your cvs directory lives in your \texttt{$HOME}, when cvs needs something look at riesling, use ssh protocol to move files around, and vim to edit cvs messages. The first 4 lines are necessary; the fifth line is optional.

Now if this doesn’t appear there, don’t panic. Check `~/\texttt{.bash_profile}`, which is your personal environmental variables. Add the cvs information in there if it wasn’t in \texttt{/etc/profile}, then tell Adam it wasn’t in \texttt{/etc/profile} and he should fix it. Check out section 5.9 for more information about what information should be in what files when it comes to your environment.

### 3.3.1 checkout and update

First off lets learn how to get something from CVS and how to update something we currently have.

1. Make a new CVS directory in your home directory

   ```bash
   > cd ~
   > mkdir ~/CVS
   ```

   and descend into that directory

   ```bash
   > cd ~/CVS/
   ```
You will always start here to get content from cvs. This means whenever you want to check out something from cvs, move into the 

```
~/.CVS
```

2. Now CVS works just like your local library. If you want, something you have to check it out. You just tell it you want to check it out. Say we want the York Group Electronic Library.

```
cvs checkout ElectronicLibrary/Utilities
```

One thing about checking out files: only checkout what you need. For example, the Software directory contains all the software we write for the lab and copies of the software we download from other groups. It is obviously a lot, lot, lot of stuff. If you’re working on COSOMod for example, just check that out

```
cd ~/.CVS

cvs checkout Software/Programs/COSOMod
```

Your probably asking yourself right now, “How do I know what is in CVS?” Great question! I wish I had a really good answer for you. Right now the only way is to log onto riesling and look around in /usr/local/cvsroot. I really need to find a simple web-based client to search a cvs repository.

Let’s say you already checked out something ... for example this document.

```
cd ~/.CVS/

cvs checkout /Projects/Mosera/GroupIntro/
```

Then you see that someone made a change to the document or added some new pictures. You want to update the directory, but you don’t want to recheck it out every time. Simplify done,

```
cd ~/.CVS/Projects/Mosera/GroupIntro/

cvs update
```

All finished. One thing to be careful of. Lets say you were making changes at the same time I made changes, and you try to update the directory. CVS is going to stop and warn you that your version is not the latest and may ask you to recheck the directory out because of this file. You have a couple choices to fix this: (1) rename your file to something and try again or (2) delete everything the directory and update. This will be more clear I think when it happens. You can get around this problem by making sure you cvs update files before you edit them.

One option that is available when you update or checkout something is the “-d” option. This will do an update or checkout directories as well as the files. Say you want to checkout not just my projects directory, but all the files and directories within it.

```
cd ~/.CVS
```
>cvs checkout -d Projects/Mosera Remember, you must be in the ~/CVS directory to check things out.

### 3.3.2 add and commit

One thing you really need to be able to do with cvs is add things. Again, very simple. There are two things you might have to add: directories and files. To add a directory. Lets make you a projects directory as an example

1. Checkout the Projects directory already in cvs and move into it.
   ```
   >cd ~/CVS
   >cvs checkout Projects
   >cd Projects
   ```

2. Now just make your directory
   ```
   >mkdir YourLastName
   ```

3. Now add this directory to the cvs repository.
   ```
   >cvs add YourLastName
   ```

CVS will add this directory no questions asked. It will prompt you for your password. If you want to add a file, there is just a little more you have to do. Lets make a file in your new directory.

1. Move into your new Projects directory.
   ```
   >cd ~/CVS/Projects/YourLastName
   ```

2. Make a file .. anything will do.
   ```
   >vi testfile.txt and write some words.
   ```

3. Now tell cvs you want to add this file.
   ```
   >cvs add testfile.txt
   ```
   CVS will prompt you for your password then say something like.
   ```
   cvs add: scheduling file `testfile.txt` for addition
   cvs add: use `cvs commit` to add this file permanently
   ```
   What cvs just said was, “I see this new file you want to add, if you’re sure you do, then use cvs commit.”

4. Ok, we are sure we want to commit this file and all the other files in this directory that we added or changed.
   ```
   >cvs commit
Again, you will be prompted for your password. Then a vi window will appear, see section 5.5. It is asking you to type a message, that will be attached to an email and go to everyone in the lab. This is how people know what is going on in CVS ... emails ... lots of emails. Type something like “Learning how to commit files.” Notice how below where you type is a list of the changes you are committing, in this case adding a file. Anyway, write the message and quit vi. CVS spits out some stuff and as long as it doesn’t say ’error’ you’re good.

Some things to know about adding and committing. When you cvs add, nothing is changed in the actual cvs repository on riesling. That is to say if you computer exploded, that change would be lost and anyone checking out that file will not see it. It isn’t until you cvs commit the file addition or changes that it happens for everyone. Also, when you cvs commit it will check every file in that directory and every file in every directory below that. Any files identified as “to be added” and any files already in cvs that are changed from the master copy on riesling will get committed as well.

There are some more tricks with cvs, so if you want to know some more go down to Section 7.5.

3.4 The Electronic Library

The York lab keeps its own set of electronic journals in pdf form. We do this for two reasons. (1) Quick access to journals of interest to the lab and (2) excellent system for making references for papers, written exams, thesis, etc. The Electron Library (EL) is kept in cvs, and if you followed the cvs instructions above you already checked out a copy of it on your machine. EL contains two things for every article: a .bib entry that has the searchable/parceable information about the article and a .pdf that is the actual article. Most importantly you need to be able to do three things with the EL: look up journals, add journals correctly, and reference journals.

3.4.1 Searching with tkYorkLib

Lets start with looking up journals. Go to the utilities directory

> cd ~/CVS/ElectronicLibrary/Utilities

that contains a bunch of useful perl scripts that help manage the EL. You should really consider adding this directory to your path, see section 5.9.2. Mostly you will use just one, tkYorklib.pl, written by Tim Giese. To run this script,

> export LANG=C then >./tkYorkLib.pl

The first command just sets the language, don’t worry so much about this at the moment. A tool bar should appear that looks like this.
This tool does a couple of things for us. The first tab called “LaTeX” we will get to a bit later, for now click on “BibSearch” and then “BibViewer”. tkYorkLib can search two different directories either the one on riesling (Network Search) or the one on your computer (Local Search). Since you just checked out the EL these should be identical, but theoretically they might be different at times. Let’s load the Local Search EL, so click on that button. In the “Search Results” box in the lower left, you should see a lot of journals, all named in a similar fashion:

LastNameFirstAuthor_JournalAbbrev_Year_vVolume_pPage

Go ahead and click on one of the entries once. On the right, you should see the .bib entry for the article. We will talk extensively about what has to be in the .bib file later. Now,
double-click the article in the search results window. This should bring up the pdf of the article you clicked on.

Now let’s search for something in particular. The searching system is based on a branching search, you will understand in a minute. In the text box by “New Branch” type York, then click the “New Branch” button. You will notice the search results have changed. Single click one of the articles and the bibviewer window will show the .bib entry. You will notice that the word “York” is highlighted. Problem is most of these articles got picked out because they have “New York” in the .bib and we really want to find articles by Darrin. So, go back to the new branch text box and add “Darrin”, then click the new branch button. The list gets much shorter.

One last tip about searching go ahead and refine your search with “1999”. That should leave you with one article. Now pretend you meant to actually search “1998”. Click on “Darrin” in the search tree window. This will reset search results window, and now you can make a new branch. Search “1998” now. This should leave 5 articles. Also note that some of them are NOT from 1998, some got picked because 1998 appears in the page number or in the abstract within the .bib file.

You also don’t have to search by name or year. Anything that appears in the .bib file will work. So let’s talk about the .bib file a bit more.

### 3.4.2 Adding Articles

Adding articles to the EL is one of the more common things you will find yourself doing when you start in the lab. It is also includes by far the most mistakes people make. This means Adam must spend his time fixing them because some mistakes require root access on riesling, which you do not have yet. So please for the love of all that is good and holy in the world, pay attention here.

**Instructions** have actually already been written for adding articles, please read this as well at some point. Here I’m going to go step-by-step through what you must do. Please read them all the way through at some point because all the steps are important.

1. Go find an article that you think the lab should have access to. Good places to go are [http://pubs.acs.org/about.html](http://pubs.acs.org/about.html), the ACS journal list or [http://sciweb.lib.umn.edu/](http://sciweb.lib.umn.edu/) is the UofM’s library (4.2.2) front page for finding journals. Open up a pdf of the article.

2. Save your pdf to `~/CVS/ElectronicLibrary/NewEntries` then rename your file in the following way:

   ```
   LastNameFirstAuthor_JournalAbbrev_Year_vVolume_pPage.pdf
   ```

   For example, that pdf of the article from 1999 of Darrin’s we looked up earlier like
this York_JPhysChemA_1999_v103_p11060.pdf. If you are not sure what the appropriate journal abbreviation, check http://library.caltech.edu/reference/abbreviations/

3. Move yourself to the NewEntries directory
   >cd ~/CVS/ElectronicLibrary/NewEntries
   and open up a new file with emacs with the same name as the article, but ending in .bib instead of .pdf.
   LastNameFirstAuthor_JournalAbbrev_Year_vVolume_pPage.bib

4. This .bib file is what the searching and reference programs use (i.e. they don’t read the actual pdf). The reason we opened this file with emacs is that emacs recognizes this is a .bib file and gives us some special options. Also, the .bib file is in LaTeX, which you will get a more thorough introduction to later.

   On the tool bar you should see “Entry-Types”, which for your journal article is “Article in Journal”. This will bring up a template that you get to fill in.

5. The .bib file must include a couple of things, while other things are optional. The file name, the authors, the title, journal, year, volume, page range, and annotate are
absolutely necessary. The key, number, month, and note are optional. Here is a filled out .bib file.

- The file name goes after “@Article{“ and before the “,”. Notice you do not have a file type at the end (e.g. no .pdf or .bib)
- The authors are always written with “and” between them and no commas. Do not put names in all caps, even if the journal does.
- The title needs an extra set of brackets around it. Please use the full title.
- Use the journal abbreviation, which should be almost exactly like in the file name, except you have “” after abbreviated words.
  
  [http://library.caltech.edu/reference/abbreviations/]
- Year, easy
- Key is any key words you think would be useful. Things you might be searching for that would get this article to come up.
- Volume , easy
- Page range with “- -” between the first and last page. Also, JChemPhys only has a first page ... its the only exception I know.
• Annote is usually the abstract. You can add more information if you think it is necessary.

6. Check for nonstandard characters like í, À, ô, etc. To find out how to make these characters, open up tkYorkLib.pl again and hit the “LaTeX” button on the left. This will bring up a menu of a bunch of different symbols LaTeX can make ... “Show Text Accents” is the button you are looking for. Click any symbol in that menu and, then paste into your .bib file and it will show you how to make the symbol. For example, \'{\AA} = À.

7. We need to fill and clean the entry. In the emacs tab (along the top border) is “BiBTeX-Edit”. Click on this menu and look for “Operate on Current Entry”, then slide over to get the sub menu. Choose “Fill Entry”, this will clean up the entry. Then repeat this process and choose “Clean Entry”, which will remove unused lines and fix up the option naming. Make sure to give it one more look over before you save and close the bib file.

8. We have a little perl script made to help check if you made mistakes. Run 

~/CVS/ElectronicLibrary/Utilities/TestBibTeX

in the NewEntries folder where your bib entry is. If it doesn’t create a viewable file, something is wrong. Then check your terminal and read the output and see if it found errors.

Now at this point you have successfully made a bibtex readable entry, but if you try to search for it using TkYorkLib it can’t find it. Why? Well that is because your new entry hasn’t been parsed and entered in to the EL, it is still in the waiting area (NewEntries). If you are 100% sure your bib file and pdf file are ok. Use cvs to add and commit the files. Now ssh over to riesling and type

updateElectronicLibrary.pl

This will tell riesling to check the NewEntries folder and parse and sort up all the files in there. Go back to your computer now and cvs update your ElectronicLibrary folder. All done.

3.5 Message Board

Without a doubt you will run into a problem you don’t know how to fix at some point. The thing is its likely that someone else ran into the same problem at some point. So the York Group keeps a message board up,

It is searchable and has lots of posts about how to compile code, what group meeting will be about, information about the supercomputing institute, etc, etc. The message board sends emails to everyone registered when new topics or posts are made so if you know the answer or have the same question, you can participate.

Now the message board along with some other internal web pages are not viewable outside the lab without a special user name and password (just ask a current member for them). So visit the page and register your first day.

### 3.6 Viewing LaTeX

In this lab we write publications with LaTeX. Think about LaTeX the same way you think about writing FORTRAN code or a Perl script, its a file that you use a compiler on that produces a product. In this case, we write a LaTeX file (usually ending in .tex) and compile them into some more readable format (e.g. pdf, dvi, .ps, .etc). For now we need you to be able to compile a LaTeX file so you can read it. At some point you will probably want to learn how to write LaTeX as well. Section 7.4 will hold a more complete introduction.

Just like for ElectronicLibrary, we need to add some information to your environment so LaTeX knows where things are and how to run them. So again, go check your /etc/profile for the following lines.

```bash
#LaTeX
export TEXINPUTS=./usr/share/texmf/tex/
export TEXINPUTS=$TEXINPUTS:/usr/share/texmf/pdftex/
export TEXINPUTS=$TEXINPUTS:$CVSHOME/LaTeX/Custom/
export TEXINPUTS=$TEXINPUTS:$CVSHOME/LaTeX/RevTeX/
export TEXINPUTS=$TEXINPUTS:$CVSHOME/LaTeX/BibTeX/
export TEXINPUTS=$TEXINPUTS:$CVSHOME/LaTeX/PresentationTools/TEX/
export DVIVIEWER='which xdvi'
export PDFVIEWER='which xpdf'
and
#BibTeX
export BIBINPUTS=./usr/share/texmf/bibtex/bib
export BIBINPUTS=$BIBINPUTS:$HOME/CVS/LaTeX/BibTeX/
export BIBINPUTS=$BIBINPUTS:$CVSHOME/ElectronicLibrary/Bibliography/
export BSTINPUTS=./usr/share/texmf/bibtex/bst/
export BSTINPUTS=$BSTINPUTS:$HOME/CVS/LaTeX/BibTeX/
export BSTINPUTS=$BSTINPUTS:$CVSHOME/ElectronicLibrary/Bibliography/
Also, go ahead and add configure to your path.
PATH="$PATH:$HOME/CVS/Software/Configure"
```
These are setting up environmental variables that LaTeX will use and they may be written as one long line instead of a bunch of separate lines.

Best way to learn is by doing, so go checkout this document, the configure program, and the LaTeX packages.

```
> cd ~/CVS/
> cvs checkout Projects/Mosera/GroupIntro
> cvs checkout Software/Configure
> cvs checkout LaTeX/Custom
> cvs checkout LaTeX/Tools
> cvs checkout LaTeX/BibTeX
> cvs checkout LaTeX/PresentationTools/TEX
```
Make sure that Software/Configure is in your pat; see section 5.9.2 if you don’t know how to do this. Now go into the GroupIntro directory.

```
> cd ~/CVS/Projects/Mosera/GroupIntro
```

If you look around in this directory you should see a couple of things, like the *.eps picture files and the *.tex file “GroupIntro.tex”. The tex file contains this document, so go ahead open it and look around. You see the text but also a lot of other characters you know aren’t shown in the document you read. In any case, close the tex file and lets make the document.

```
> configure -latex GroupIntro.tex
```
This is telling configure to write out a makefile, Section 7.1, that will compile a LaTeX document and that it should start the compiling by reading GroupIntro.tex.

Now try to compile the document.

```
> make
```
This didn’t work because you didn’t tell it what type of output you want, so instead it just gave you a list of the options. Most people like reading tex files in dvi format, so lets try that.

```
> make dvi
```
You will see a ton of stuff roll across your terminal. This is the latex program compiling the document. Finally, it should open up this introduction with your dvi viewer.

### 4 Resources

You’re not walking into a blank room, obviously we have been here a while and there are some resources in place that can save you some time.
4.1 In the Lab

First, what kind of resources are available in the lab? Right off the top of my head, beyond your computer, I’m thinking about our website, message board, QCRNA database, printers, backup, the dart board, etc. Let’s go over a couple of useful ones.

4.1.1 Website

The York Lab website http://theory.chem.umn.edu includes a lot of useful information. Including

1. People - list of former and current members with email addresses and homepage links.
2. Research - broad overview of research done in the lab, though not always current.
3. Publications - list of publications with links. If you click the little picture to the left of each title, it should open a pdf version of the article.
4. Intranet - list of all kinds of resources on using the UofM library, software, chemistry background, linux information, FORTRAN compiling, LaTeX and BibTeX resources, cvs information, perl basics, html basics, MSI information, funding and fellowship links, and miscellaneous information.
5. Schedule - group meeting schedule with dates, presenters, and topics.
6. Links - UofM and Minneapolis website links
7. Contact - contact information for Darrin with map and direction information for someone visiting
8. QCRNA - link to our RNA database
9. News - group highlights, not always current

You may be asked from time to time to make changes to the webpage. All the webpages that are NOT personal webpages, live in CVS/html (for cvs info see Section 3.3). You can check out these pages and make changes just like any normal cvs file. When you commit changes in this directory, cvs runs a script that will mirror those changes on theory.chem.umn.edu so that they actually appear to someone surfing the web. Just a reminder, cvs lives on riesling.chem.umn.edu, but the webpage lives on theory.chem.umn.edu. If the mirror script errors, make sure to tell someone that can fix it.
At some point you may be asked to create your own personal webpage about your research. Our webpage lives on theory.chem.umn.edu, which is the computer that sits by riesling. This machine does not have user accounts, so to get access you need to talk to Adam or Tim about getting the ssh key (see Section 5.11) that allows you to log in. To make your webpage, go down and read Section 6.2

### 4.1.2 Backing up your computer

Currently, we are using external hard drives for backing up our machines. This allows each person to be individually responsible for backing up their computer. Additionally, anything that is properly committed to cvs also is stored on riesling, which is itself again backed up. You need to be able to take an external hard drive and set up it to easily back up the information on your computer.

You’re welcome to go about it any way you like, here is an example. Get yourself (i.e. tell Adam you need) an external hard drive. Currently, we are using Western Digital MyBook of the 500gig variety. This means you can backup everything on your computer, but it is a little too big to carry around with you. For this exercise, I’ll be using the WD 500gig MyBook just because that is what most other people are using.

First off, if you don’t know anything about mounting, formatting, partitioning hard drives and the following instructions confuse you, then go check out Section 5.8.

So take your MyBook out of the box, plug in the power, make sure your computer is booted up, and plug in the usb cable in the back. Fedora Core is pretty decent at detecting and mounting this external disk. You should see the My_Book on your desktop. To see some more information, try out

```bash
>cat /etc/fstab
```

This will tell you all about what devices are mounted, where it is mounted to, what format the disk is, and what are the mounting options. You should probably find an entry kinda like the following:

```
/dev/sdb1 /media/My_Book vfat auto,user,exec 1 0
```

I’m not going to get into disk formatting types, I don’t know enough to talk about it. I do know that vfat is readable and writeable by both windows and linux, so if you plan on using this backup for multiple machines it isn’t a bad choice. If you are planning to just backup a linux partition, ext3 type is the way to go. Anyway, back to getting stuff done.

Go ahead and make yourself a directory on your backup drive, say

```bash
mkdir /media/My_Book/backup
```

Tim made this little script, which works wonders.

```bash
#!/bin/bash
if [ ! -d /media/My_Book ]; then
```

18
echo "/media/usbиск does not exist! Stopping without action."
fi

if [ ! -e /media/My_Book/backup/ ]; then
echo "usbиск does not appear to be mounted. Stopping."
else
    echo "Everything looks ok."
    rsync -avzu --rsh='which ssh' --exclude=".*" --exclude="*~"
        --exclude="*.rwf" --exclude="*.chk" --exclude="*.e"
        --exclude="core.*" --exclude="Desktop*" --delete ~/media/My_Book/backup/
fi

Place this in your bin directory so you can run it from anywhere, I have it named “FROM_HERE_TO_USB”. When you run this it will check to see if your external hard drive is mounted, if it is it will make sure your directory is available, and if all that is true then it will make a mirror of your home directory in backup. If you want to understand more, just learn about rsync.

>man rsync

It is also possible to automate this process, see Section 7.3.

4.1.3 Connecting your computer

There will be times you will need to connect to your computer from somewhere else (i.e. connect remotely). For basic networking information go check Section 5.10. All the computers in the lab have a static ip address. Here is a list with subnet, gateway and DNS:

chardonnay .160
pinot-blanc .162
bordeaux .164
muscat .166
shiraz .167
pinot-noir .170
sylvaner .168
chainti .161
buzet .163
rioja .165
tokay .169
For anything else (i.e. laptops, windows machine, etc) must use either wireless or DHCP to connect. For wireless, we have a access point in the lab, just open an internet browser and it will prompt you for your x500 information.

Now if you want a hardline, you need to register your computer. Go to https://dhcp.umn.edu/. Simple set by step instructions, the only thing you need is the hardware address, which you get by becoming root and using

> ifconfig

Look for the line that has “HWaddr” for “eth0”. It will have something like “00:16:76:B2:12:B6”.

### 4.1.4 Printing

We have four printers in the group.

- **Group color printer** - HP Color LaserJet 3700 dn - This printer is in the lab and used for most printing. It does double sided printing. IP address 128.101.162.176

- **Group bw printer** - HP LaserJet 4000 N - This printer is in the lab and is used when either the color printer is down or you are printing something that doesn’t need color or double sided printing. IP address 128.101.162.175

- Darrin’s color printer - - In Darrin’s office.

- **Group color printer (old)** - - In the extra office in smith. Very old printer that does not do double side. IP address ???.???.???.???

To print to any of these computers, you need to set up a printer driver. This is just a set of options that tell your computer how to connect and print to a particular printer. The problem is different versions of fedora core set up drivers differently, so I can’t give you a single step-by-step set of instructions. A couple of things to remember.
1. printers use network port 9100
2. the color printer uses tray 2 to load paper
3. don’t print non-standard paper sizes
4. use long-edge for double sided printing
5. make a driver for both the color and bw printer
6. make sure you have an opening in your iptables for the printer (see Section 5.10.1).

4.2 Outside the Lab

Good stuff you can find outside the lab.

4.2.1 PDF Scanner

The department keeps a scanner in the main office (Smith 139), that will scan then email you a pdf. Its on the desk across from the secretary (Kathy Ross). There are instructions hanging overtop the scanner. This thing is great when you have an old journal or book chapter that you have only in hard copy.

4.2.2 Library

The library has a boat load of stuff you can get/do. I don’t know all of it, but the one thing you better know is http://sciweb.lib.umn.edu/. This site has links to

1. MNCAT - the library search engine
2. Chemistry specific journal connections
3. Interlibrary Loan (ILL) service

4.2.3 Electronic Shop

The chemistry’s electronic shop is a very useful spot:
http://www.chem.umn.edu/services/eshop/
The EShop is staffed by

- Bruce Moe - Manager - Go to guy for questions on the university network, wireless network, computer problems, etc.
• Mike Casey - Mr. Fixit. Great with laptops.

• Eric Schulz - Webmaster for the department and the person I go to to put in purchase orders to for computers/parts.

Generally speaking these guys don’t know a ton about linux as an operating system, but they know a lot about computers, printers, networking, windows, macs, etc. This is also the office through which we are suppose to make large computer purchases. They also keep a stock of simple parts if you need something (screws, cd-roms, keyboards, etc).

4.2.4 Minnesota Supercomputing Institute

The Minnesota Supercomputing Institute (MSI) is our primary resource for computer time. Most everything I am about to tell you can be found at their [website](#). There is a lot going on at the MSI, but I’m just going to highlight a couple of the things you probably need to know now.

The MSI runs a bunch of [workshops and tutorials](#) throughout the year. Some are more useful than others, but if you are brand new to scientific computing and/or linux, I suggest going to some of the early ones. Also, they post pdf versions of the power point slide presentations [here](#). In particular, check out the “Chemistry and Physical Sciences” and “Hardware Resources and Programming Basics”.

4.2.4.1 MSI Hardware

Next you might want to know what actually [computers](#) are at the MSI. This actually changes with time as older machines are taken away and new ones are added. As of right now, there are 5 main machines: Calhoun, Blade, Altix, Regatta, Elmo. Each are a little different and are better used for different calculations, so let’s talk about each one.

• **Calhoun** - This is an SGI altix cluster. It has 256 computer nodes. Each node has 2 quad-core 2.66 GHz Intel Xeon processors and 16 GB of RAM. Best used for jobs that require lots of processors like big molecular dynamic simulations.

• **Blade** - This is an IBM Bladecenter Linux cluster (ya, like in the commercials). It has 307 computer nodes. Each node has two dual-core 2.6 GHz AMD opteron processors and 8 GB of RAM. Best used for jobs that require lots of processors like big molecular dynamic simulations.

• **Altix** - An older SGI altix cluster that is a collection of different altix systems. It has 10 computer nodes: 1 with 256 1.6 GHz processors and 512 GB RAM, 1 with 48...
1.3 GHz processors and 96 GB of RAM, and 8 nodes with 16 1.5 GHz processors and 20 GB of RAM. I suggest using the altix for quantum mechanic calculations that you want lots of processors and need a lot of RAM.

- Regatta - An IBM Power4 system with 23 computer nodes. Nodes are a mix of 1.3, 1.5, and 1.7 GHz processors with varying amounts of RAM (though 552 GB total). I suggest using the altix for quantum mechanic calculations that you want lots of processors and need a lot of RAM.

- Elmo - New linux cluster. 192 AMD 2.3GHz Opterons. I’ve been using this for quantum calculations and I think others have been using it for CHARMM.

What computer you use will often be determined by the software. Some software is available on all the machines (e.g. Gaussian03), while other software (e.g. our home built CHARMM/MNDO) might only be compiled and usage on a couple machines. Another thing to think about is if you are running the jobs in parallel or serial. As an example, I needed to run something like 400 Gaussian03 jobs. Gaussian03 doesn’t scale that well, so each job was only going to use 4 processors. Therefore I used the altix and regatta mostly because I didn’t need the massively parallel Blade or Calhoun. Tai-sung, on the other hand, runs NAMD to do 256+ processor MD simulation jobs, so he uses the Blade and Calhoun.

### 4.2.4.2 Connecting to the MSI

Finally, how do I get an account on those machines and how to connect to them. Getting an account requires asking Darrin. He will email the MSI and they should send you a password and login name. Make sure they give you an account on all five of the machines above.

Most often you will use ssh, see Section 5.11 for more info, to connect to these machines. Log onto any one of them and change your password immediately by typing

```
> passwd
```

Almost all the commands you use on your computer work here.

### 4.2.4.3 Running Calculations

Each machine is a little different. Check the MSI webpage for the best way to submit and manage jobs.
5 Learning Basic Linux

What I mean when I say basic linux is the ability to navigate between directories, open and edit files, and be able to control processes on your computer.

Here are a couple of more complete and well written introductions to using basic linux.

1. http://linuxreviews.org/beginner/

2. MSI Tutorial

3. Evelyn’s cheat sheet

4. User guide

As I noted above most of what happens in linux is typed out instead of clicked. You do this typing on a command line that appears in a terminal window. You can open a terminal either by clicking the picture that looks like this

![Terminal Icon]

or by clicking on “Applications” at the top of your screen, moving down to “Accessories”, then clicking “Terminal”. Now you have a place to type commands to your computer.

5.1 Users

First off its important to know who you are. Type

>whoami

This will return who the computer thinks you are, which ought to be the login name you signed on as. You are not the only user on this machine, you can list them all using

>cat /etc/passwd

You will notice that there is a lot of users on this machine and most of them aren’t names you recognize. For example, there is user “sshd” in that list. There is also user “daemon”. These are examples of programs that run independently of any user, or more accurately they are their own users. “sshd” is a program that controls the ssh program, which allows you to remotely connect to other computers. We have users so that we can define what operations/programs everyone should have access to.

Toward the bottom of the list, you should see yourself. Here is my entry on my computer.
mosera:x:714:100::/usr/people/mosera:/bin/bash

The information is given as a list with “:” to separate them. Here is an explanation.

- loginname:password:user id:group id:home directory:shell

Note that any user can see /etc/passwd, but only root can modify it (something you don’t want to do unless you know what you are doing). Also notice that my password is “x”. That is because we encrypt the passwords and put them in a different file, the “x” tells the operating system to go look there.

I want to point you to a very special user on this computer, “root”. Root, also called the superuser, is the administrator of the computer and has access to programs and files that make up and control every part of the computer. Having root access means you have complete control of your computer both for good or evil. Remember, because root is so powerful, only become root if you need to do something that absolutely requires it.

Just because you login to the computer as one user doesn’t mean you have to stay that way. The command

```
>su
```

allows you to change who you are and stands for switch users. For example, I log onto the computer as “mosera”, but I need to do something that requires me to be “root”

```
>su - root
```

It will prompt me for the root password, and assuming I type it correctly I am now root. Note that as root you will not be prompted for a password when using su, because as root you have access not only to everyone’s account but also their identity.

### 5.2 The File System

The linux file system is just like a giant, branching tree. It all starts in the “/” directory and moves out from there. Most people visualize “/” living at the top and you branch down. So if you go “up” a directory, you’re going back toward “/”. If you go “down” a directory, you’re moving away from “/”.

Your home directory probably lives in

```
/usr/people/yourname
```
or in

```
/home/yourname
```

This is means your home directory is subdirectory. Unlike windows, for a counter example, you can look around your computer at all the files very easily, including the files that run the operating system itself. As you get more adept in using linux, you will likely end up making modifications. Here is an example of a linux file tree, most likely close to your tree.
You can think of your command line (where your current terminal is) as sitting at a specific spot on the tree. From any particular place in the tree you can jump to any other as long as you know either the absolute path to where you want to go in respect to “/” or the relative path to where you want to go. Some simple commands that allow you to navigate your file system.

1. **pwd** - print working directory
   
   `>pwd` will print the full path to where you are currently at.

2. **cd** - change directory
   
   This command actually has a lot of uses. It can move someone to their home directory just using the command. If you do

   `>cd`
   `>pwd`

   you should be in your home directory.

3. **ls** - list directory contents

26
Simplily lists the files in the directory current directory. A useful option is `ls -l` which provides all file information.

4. tree - print directory tree
   This command can put out a lot of output, so give it a shot with the following options.
   `tree -dx -I CVS` Your output will be the whole directory tree not including anything that has CVS in it. This command can be useful when you want an idea of where things are in the tree. If you remove the `-dx` options, it will pring the files as well as the directories.

Finally, I want to give you a quick way to get an idea of how much space your files are taking up and how much space you have on your disk. First is, `df` which gives a breakdown of how your disk space is arranged and in use. Try out `df -h`

You should get 6 columns of information. For example,
```
/dev/sdc1 459G 29G 407G 7\% /Data1
```

1. Filesystem - the device name of the disk, which frankly can be pretty complicated since certain software can be used to manage disk usage.

2. Size - How much space that chunk of disk has. The “-h” argument makes it get printed in megs and gigs, instead of bytes.

3. Used - How much space on the disk is currently full.

4. Avail - How much space on that disk is available.

5. Used% - What percent of the disk is used.

6. Mounted on - Where this disk is mounted and more importantly how you identify what your looking for. For example, look for “/” here. That will tell you how much disk space is available for things like root and the operating system to write to. Also look for “/usr/people” or “/home/”. This will give you information about your home directory.

What if you wanted to know how much space a specific directory is using up? Well, `du -h` will do this for you. When you run this command you will see a complete it will give a complete breakdown of every directory. If you just want a summary, try `du -sh`.

For a full explanation on the file system, look [here](#).
5.3 Files

Most of your work will be done in files, so knowing a little about files might be useful (read this or this). First off, every file is owned by someone and has certain permissions about who is allowed to do what to the file. Let's look at an example I know you have.

```bash
>ls -lh ~/CVS/ElectronicLibrary/Utilities/tkYorkLib.pl
```

This should produce the following output.

```
-rwxr-xr-x 1 mosera users 274K Aug 28 2007 tkYorkLib.pl
```

Here is how to read this. (1) The first block of letters is the permissions. (2) The “1” designates the number of files contained in this file. (3) The owner of the file. (4) The group of the file. (5) The size of the file. (6) The date the file was made. (7) The name of the file.

Now, about the permissions, this is shown in a block of 10 letters. The first letter is either “-” or “d” for file or directory. The next 9 letters are grouped as three sets of 3 letters. Each group of letters are for read (“r”), write (“w”), or execute (“x”). The first grouping of letters are for the owner of the files permissions, the second group of letters
are for the groups permissions, and for the third group of letters are for anyone else using the file. The permissions are very important as they decide who is able to do what to a file or directory. If something isn't working and you have no idea why, permission issues are one of the first things to check.

Usually the permissions are fine, but if you need to change them the chmod program will help. The usage is straight forward. You can change the permissions of the user (u), group (g), others (o), and everyone (a). Say you wanted to make a file you wrote readable by the “users” group but not anyone else.

>chmod g+r

Or let's say you write an executable perl script that you want others to be able to use.

>chmod a+x

Now everyone can execute your script. Use >ls -l filename to see how the permission block changes.

Sometimes it isn’t that you need to change the permissions, but you need to change who owns the file.

- chown - changes the owner of a directory or file. Usage
  >chown newusername filename

- chgrp - changes the group of a directory or file. Usage
  >chgrp newgroup filename

5.4 Aliases and Symbolic Links

One of the most simple ways to customize and optimize your computer is through aliases and links. When used properly both can make your life a lot easier.

An alias is just a way to assign an abbreviation. These are usually set in your ~/bashrc file. For example, every time I log into riesling, I don’t want to type out

>ssh -XY mosera riesling.chem.umn.edu

So I created an alias in ~/bashrc that reads

alias riesling='ssh -XY mosera riesling.chem.umn.edu'

Now all I have to do to ssh over is type

>riesling

Another example of an alias that I strongly recommend you make is

alias rm='rm -i'

The rm command destroys files, but using the “-i” argument forces you to confirm you want to delete something before it destroys it.

Symbolic links is a special type of file that is actually just a path to another file or directory. For example, let's make a symbolic link for tkYorkLib.pl into our ~/bin.
>cd ~/CVS/ElectronicLibrary/Utilities/
>ln -s tkYorkLib.pl ~/bin/LibSearch

Now in ~/bin/ there is a file called LibSearch (check this using ls -l). If I say LibSearch (and assuming I remembered export lang=C), then tkYorkLib ought to appear.

5.5 Vi

Vi is a screen-oriented file editing program. When you open a file with vi you do not get new window, rather the file is opened directly within your terminal. This is the feature that makes vi useful, because if you are connecting to your computer remotely, you may not be able to create new windows. On the other hand, you don’t have a mouse in vi usually, so vi uses an alternative method to manipulate the text.

Another little bit of information, is that when you say >vi newfile.txt you are most likely getting the program Vim, “Vi IMproved”. Vim is just a more advanced version of vi, I’m just noting it here so you know.

Using vi is fairly straightforward. Vi operates in either “insert” mode or “command” mode. In insert mode, you are basically just typing like you would in any other text editor. In command mode, you are able to manipulate the text similar to ways you might with a mouse. Therefore, the real trick to using vi well is knowing the hot-keys that manipulate text while in command mode.

There are about a hundred good tutorials about how to use vi, try out this one, or this one, or maybe this one. I just want to highlight a couple of commands.

- “i” - enters insert mode. Once you use this command, you can start typing like normal.
- escape key - if you are in insert mode, this will return you to command mode.
- “d” - delete current line
- “d#d” - delete any number (#) of lines
- “:w” - write file, basically the save key
- “:q” - close file, quit without saving
- “:wq” - write then quit
- “:wq!” - write then quit forced
5.6 Emacs

Emacs is likely going to be your standard text editor when you’re working at your computer. This is closer to a standard text editor than vi as it opens a separate window to type in. Just like vi, if you want to really be efficient, you need to know the hot-keys and how to use buffers. Buffers are just split windows that are another way to manipulate your text.

Again there are tons of tutorials out there and explanations, I suggest this one. Usually, you will just want the list of the hot keys.

5.7 Processes

Your computer is just a collection of programs and files. Most of the programs are going on in the background and you would never know. Each process/program is identified by a number, the process ID number (PID). Knowing what processes are controlling what actions on your computer and controlling the processes you start is very useful. Here are a couple of useful programs:

- **top** - This program gives a lot of information on what processes are currently running and who controls those processes. It is updated about every 4 seconds and is sorted by cpu usage. Also memory usage, uptime, process ID number (PID), etc.

- **ps** - Lists current processes as a list, instead of an updating screen.
  
  `<ps auxf` will list every process in tree format with full information. This is a very powerful way to track down a job. For example, lets say your internet browser, firefox locked up and you needed to find the process to kill it.
  
  `<ps auxf | grep firefox`
  
  Will show all the processes that have the word “firefox” in it.

- **jobs** - Lists all active programs started in the terminal you are working in.

- **kill** - Attempts to kill a process. You can identify a process either by process ID number (PID) or by its number from the “jobs” output. For example, lets say your firefox browser locked up and you used “ps” to find that
  
  `mosera 3686 0.0 0.0 4448 1068 ? S Apr01 /usr/bin/firefox`
  
  the PID number is 3686. I want to kill it, so I use
  
  `<kill –9 3686`
  
  and the firefox browser goes away.

- “&” - This symbol after any program call will place the program in the background so you can keep typing. Example,
>emacs test.txt
This will open an emacs window, but your terminal command line will be locked.
>emacs test.txt &
This will do exactly the same thing, but the terminal command line will still be available.

- xkill - This command will change your cursor into a little target, the next window you click on will be closed and the process killed.
- CTRL + C - breaks/cancels an ongoing operation/program/process. For example you run >./myprogram and realize it is an infinite loop. CTRL + C will stop it immediately.
- CTRL + Z - pauses an ongoing operation. For example, you open a file with emacs
>emacs file.txt. This locked out your terminal. Hit CTRL + Z to pause it. You can then use “bg” or fg” to put it into the background or foreground (restart the process).

These are just a few tricks. Check out
www.comptechdoc.org/os/linux/usersguide/linux_ugprocesses.html or http://linuxreviews.org/beginner/shortcuts/ for a really detailed explanation.

5.8 Hard Disk

So mounting, formatting, partitioning, and troubleshooting your local hard disk is a pretty useful and involved process. I really don’t know all the ins and outs, so I’m just going to provide a few useful websites.

1. formatting hard drive
2. partitioning and mounting a hard drive
3. Making a new linux ready disk
4. fuser
5.9 Your Bash Environment

You can control a lot about how your computer acts and looks. Really if your good enough you can change just about anything. What follows is some useful information about how and where some of those changes can go.

If you want to know all of your environmental variables try.
> env
This is a really long list of every environmental variable, including some very important ones like “PATH” and “HOME”.

5.9.1 Where Environment Info Goes

Your bash environment is controlled by a handful of files:

- `~/.bashrc` should contain “User specific aliases and functions”
  I am very afraid of the `rm` command, because by default it doesn’t ask you if you’re sure you want to delete the file (i.e. the force delete option is default on). So I add `alias rm='rm -i'` to my `~/.bashrc` file. Now whenever I delete something it asks me if I’m sure.

- `~/.bash_profile` should contain “User specific environment and startup programs”
  An example of a user specific environmental setting would be `export PATH=/usr/people/name/junkbin:$PATH` which would be specific to that user’s directory tree.

- `/etc/profile` should contain “System wide environment and startup programs”
  The cvs export commands are an excellent example of this. Everyone using the machine needs cvs setup the same way. Here are some things that should be included in in everyone’s `/etc/profile`.

- `/etc/bashrc` should contain “System wide aliases and functions”.
  Similar to `~/.bashrc` except you want it to be true for everyone.

5.9.2 Your Path

When you want to run a program, your computer has to know where it is. Even simple programs like “ls”, which shows what is in your directory is a program. When you try to use a program, your computer looks in two places: (1) the directory you are in and (2) all the directories in your path. Most of the programs you run are not in the directories your
terminal is current in. Your path is a magical variable that helps your computer find the programs your interested in. Really all you care about for the moment is seeing what is in your path and how to add things to your path.

To see what is in your path type

```
> echo $PATH
```

This will be different for every machine, but just as an example my path looks like this:

```
/usr/people/mosera/bin:/opt/intel/fc/9.1.040/bin:/usr/kerberos/bin:/usr/local/bin:
/usr/bin:/bin:/usr/X11R6/bin:/usr/people/mosera/CVS/Software/Configure:
/usr/people/mosera/CVS/DataBase/Semiempirical-SRP/Utilities/EXE:
/usr/people/mosera/CVS/ElectronicLibrary/Utilities
```

What this is saying is that if I try to run a program and my computer can’t find it in the directory I am in, it will start checking each of the directories in my path ... like my bin, usr/local/bin, usr/bin, etc etc etc.

Adding directories to your path is pretty simple. If they are directories every user should have access to (like LaTeX things or Fortran compiler things), then change the path in /etc/profile. If they are things that are just for you (like your private /usr/people/lastname/bin that has your own programs and scripts), then change the path in ~/bash_profile. No matter which place you make the addition to your path, the text is the same. Lets say I want to add /usr/people/lastname/CrazyScripts directory

```
PATH=$PATH:/usr/people/lastname/CrazyScripts
```

export PATH

Really important that I have the “PATH” in there because I want to add to the existing path, not overwrite it. If you don’t export the variable PATH, the computer won’t remember it.

Also, when you set your path you can use other variables. For example, there is a set variable called $HOME. which holds your home directory. Go ahead and

```
echo $HOME
```

It should print your home directory. You can use this to set paths. For example,

```
PATH=$PATH:/$HOME/CVS/Software/Configure
```

will add the configure directory in cvs assuming you checked it out.

Now that these directories are in your path, any programs can be found there. So if you put some cool perl script that helps spell check a tex file for example in the $HOME/CrazyScripts directory, your computer can find it for you no matter what directory you are currently in.

A good practice example is to add CVS/ElectronicLibrary/Utilities to your path so you can use tkYorkLib.pl from anywhere.
### 5.10 Network

So as you know each computer is hooked into the university’s local computer network. This is what allows you to connect to the internet, use cvs, connect to your computer from home, etc. For a full tutorial on networking concepts and fundamentals, you should probably head to the library because it is very involved. Here is a little pdf slideshow with some introduction on networking. Here is some linux specific network information.

I find 50% of the problem is knowing the terminology, so here are the basics. Your computer is connected through an ethernet cable, also called “Cat 5” for category 5 cable. This wire runs into a network room that holds a network switch. The switch identifies you with a number, called your ip address. The “ip” is four sets of three digits, for example: 128.101.162.175

Your computer’s ip is assigned this number in one of two ways.

- **Static** - every time you connect to the network your computer identifies itself to the switch by the same ip address every time. On old switches this is defined by the hardware, but on newer networks (like ours) it is set on your computer. N.B. if more than one computer tries to identify itself with the same static ip address, bad things happen.

- **DHCP** - dynamically assigned ip address every time you connect. The switch usually has a set block of ip numbers that it can assign using DHCP. If those numbers are all used up no one new can connect with DHCP.

One note about about DHCP. To stop just anyone from walking up to a plug and jumping on the network, when you connect to the switch using DHCP the switch asks your computer for its MAC address. This is an identification of the hardware on your network card, you will read below how to identify it. Every network card has one and they are all different. The switch checks to see if this hardware is on its list, and if so you get to connect. How do you get some hardware on this list? You register it here.

The switch also does something else for you. Every time you want to search something on google, you don’t type 64.233.167.99 instead you type www.google.com. How did your computer know that www.google.com = the correct ip address that leads your computer to the right google computer. This is because the switch knows the internet phone book, called the Domain Name System (DNS). The DNS translates hostnames (e.g. www.google.com) into ip addresses. For the University of Minnesota, we have two computers that keep the DNS at ip’s 128.101.101.101 and 134.84.84.84.
5.10.1 Iptables - your firewall

An important program if your internet is going to work properly is your iptables. This program is the linux version of a firewall. Even for linux, keeping a firewall up is very very important as evil people on the internet are attacking your computer basically all the time. (This isn’t a joke I can show you the logs.) For a lot of detailed information I suggest reading this or this. For now let me just show you the things I found useful to know.

The file that controls your firewall is in /etc/sysconfig/iptables. The only user that can edit this file is root, so you need root access. Basically what it is doing is setting up a list of rules to follow anytime something tries to connect to your computer from the internet. Now different version of fedora core and red hat write their iptables a bit differently, so there might be some differences from my example, but the general idea is the same. A typical line may look something like this

```
# ALLOW NEW SSH CONNECTIONS FROM THESE MACHINES
-A INPUT -m state --state NEW -m tcp -p tcp --dport 22
-s 128.101.162.0/255.255.255.0 -j ACCEPT
```

This line is allowing new connections to be made on port 22 using the tcp network protocol; in more simple terms it is allowing ssh connections. This access is restricted though for only machines that have ip numbers from the 128.101.162.XXX subnet (i.e. only computers in the lab).

Most of the machines should have their iptables set to be very paranoid and blocking most all connections. If you need to connect from a particular computer (say at your apartment), then you will need to add a line in the iptable allowing access to that. Say your ip is 150.101.202.45 at your house. You would want to add a line to your iptables that looks like

```
-A INPUT -m state --state NEW -m tcp -p tcp --dport 22
-s 150.101.202.45 -j ACCEPT
```

Now just changing the file isn’t enough, you have to restart the iptable program to put the changes into effect.

```
>service iptables restart
```

Now you should be all set to work from home.

From time to time a program or process that needs the net will and you will have no idea why it isn’t working. A good thing to check is if the iptables are blocking it. To test just shutdown the iptables for a long enough to test it.

```
>service iptables stop
```

Do your test.

```
>service iptables start
```

And make sure they go back up. This little check might save you a lot of hours wondering
why something that should be working isn’t.

5.10.2 Trouble Shooting

At some point you will likely have problems with your network/internet connection. It happens to everyone. Here are some simple commands that might help you diagnose the problem. Some of these will require root access.

- `mii-tool` This program tries to detect if your computer’s ethernet cable is plugged in and if it is getting any signal from the other end. If it doesn’t say something like

- `ping` This will try to test to see if you can connect to another computer. Try out

```
> ping 128.101.162.178
```

You should get something like this as output

```
64 bytes from 128.101.162.178: icmp_seq=1 ttl=64
time=0.143 ms
```

That ip address is riesling.chem.umn.edu. The output is saying how much information was sent out and how long it took to get back. Hit “ctrl + c” to stop the program, and it will give you a summary of what it did, something like

```
12 packets transmitted, 12 received, 0% packet loss, time 12546ms
rtt min/avg/max/mdev = 0.076/0.140/0.721/0.058 ms
```

This shows how many packets of information you sent and that 0% was lost, which is probably a good thing. Note you can also use `ping` with a hostname, like

```
> ping riesling.chem.umn.edu
```

```
eth0: negotiated 100baseTx-FD, link ok then you have a problem.
```

- `nslookup` Sometimes your network may work just fine, but the DNS is down. This will attempt to use the DNS to change a name into an ip address or the reverse.

- `ifconfig` This lists all internet connections your computer is currently running. This program requires root access. If everything is ok, you should see a block of test for “eth0”, this is your standard network connection. Note that the block of text includes your MAC address, which is right after “HWaddr”. Also listed is your ip address, directly after “inet addr”.

- `ifdown & ifup` These commands will turn off or on an internet connection. For example, lets say when you ran `ifconfig` you did not see a block for “eth0”. You could try
>ifup eth0

to see if it will start up. Alternatively, if you made any changes to “eth0” and you want to reset the connection, use
ifdown eth0 then ifup eth0 to restart it.

- service network start/stop/restart/status Your network just a service being run by a deamon. As with all services you can either start, stop, restart, or check its status using the ’service’ command. Very often if your network isn’t working doing a service network restart will get you fixed.

So if your internet connection isn’t working, use mii-tool to see if the cable is plugged in correctly. Try ifconfig to see if your computer has a known connection. Use ping to see if you can connect to another computer. Use nslookup to make sure your computer can connect to the DNS. This is a great start at troubleshooting. Once you have an idea of the problem use service network to fix it.

5.11 SSH

When you connect to a computer remotely, most of the time you will be using Secure SHell. This program allows you to make encrypted remote connections and is often used by other programs when they need to do something remotely (i.e. scp or rsync). If you want a detailed explanation of how this network protocol works, click the link above.

Basic usage of very simple. Lets say you want to connect to the altix machine over at the MSI.

ssh username altix.msi.umn.edu

This will prompt you for your password and then log you in with a command line. You can also write out the same command like this.

ssh username@altix.msi.umn.edu

or

ssh -l username altix.msi.umn.edu

The only other option I find use for is -X. This allows X11 forwarding, which in simple terms means you can get graphics as well as the command line. Say you wanted to use gaussview or emacs to look at some files on the altix. When you connect if you need to say

ssh -X username@altix.msi.umn.edu

otherwise when you try >emacs myfile.txt it will fail to send the graphics of the emacs window to your machine.
One final trouble shooting point about ssh. Every time you connect to a machine ssh writes down that machines information in `/ssh/known_hosts`. Sometimes you will try to connect somewhere and it will give you and error

`@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED! @

IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!

Someone could be eavesdropping on you right now (man-in-the-middle attack)!

It is also possible that the RSA host key has just been changed.

The fingerprint for the RSA key sent by the remote host is

`23:00:20:83:de:02:95:f1:e3:34:be:57:3f:cf:2c:e7`

Please contact your system administrator.

Add correct host key in `/home/xahria/.ssh/known_hosts` to get rid of this message.

Offending key in `/home/xahria/.ssh/known_hosts:8`

RSA host key for localhost has changed and you have requested strict checking.

Host key verification failed.

This is just SSH being paranoid because the information your computer wrote down in

`'/ssh/known_hosts'` is different than what that remote computer is sending back now.

If you are sure there is nothing shady going on, just open up your `known_hosts` file and delete the entry for this remote computer (in the above case it is entry 8).

### 5.11.1 Keys

If you find yourself constantly connecting to a particular computer, typing your password over and over can really suck. So SSH provides a way to make a 'key' that you put in a file that can act like a password for you. This is how I make my ssh keys (read more [here](#)).

So you have your computer 'wine' and you want to stop typing your password every time you connect or transfer files from computer 'blade'. Here is what you do.

1. Be on 'wine' and type
   ```
   > cd ~/.ssh/
   > ssh-keygen -t dsa
   ```
   This will prompt you for a passphrase, type something pretty long. This will create two files 'id_dsa' and 'id_dsa.pub'

2. Move the public key to the correct places.
   ```
   scp ~/.ssh/id_dsa.pub blade.msi.umn.edu:~/.ssh/authorized_keys2
   ```

3. Now tell your computer to use this new key you just made.
   ```
   > ssh-agent sh -c 'ssh-add < /dev/null && bash'
   ```
This should prompt you for your passphrase one last time.

4. Now try it out.
   `ssh blade.msi.umn.edu`

   Now you should be all set and whenever you use ssh (or any program that relies on ssh), it will not prompt you for a password.

6 HTML

HyperText Markup Language (HTML) is the primary language for writing webpages. If you want some history on HTML read this. For this lab you will care about knowing HTML because you may have to edit our webpage at some point and likely you will be asked to make your own webpage.

6.1 Tutorials

I learned simple HTML from here, which is a VERY basic introduction. I really don’t know HTML that well so I will leave it to others to point people in the way of good, useful information.

6.2 Your webpage on our website

At some point Darrin is going to want you to put up a webpage on our server so he can send people interested in your work somewhere. Our webserver is theory.chem.umn.edu. While making your personal webpage is up to you, how it is linked into the groups webpage is standard.

   You do not have an account on theory to make changes. The only user is ’httpd’. To be able to log into theory as that user you need an ssh key_gen that Adam or Tim will get you (see Section 5.11.1 for information on ssh key_gens). This key will be installed on riesling only, so to get over to theory you must do the following.

   > ssh -X riesling.chem.umn.edu
   > ssh -X httpd@theory.chem.umn.edu

   Now you are user ’httpd’ in directory ’/usr/people/httpd/’. Go up a directory then into the People directory so you can find where to put your personal webpage.

   > cd ../People/
   > ls -l

   Notice that each persons directory is owned by the httpd user. If you do not have a
directory here, make one with your username (i.e. \texttt{mkdir mosera}). This folder is where you will put your webpage. Just to test if things are working, make a file called \texttt{'index.html'} with the following in it.

\begin{verbatim}
<HTML>
<TABLE WIDTH="100\%">
<TR>
  <TD>
    <IMG SRC="http://theory.chem.umn.edu/Group/Images/UMN_logo2.gif">
  </TD>
  <TD style="vertical-align: top;">
    <H2>My Name Here</H2>
    Welcome, friend, to my webpage.<BR><BR>
    <i><b>Latest news:</b></i> I now have a webpage!<BR><BR>
    <FONT SIZE="+2"><CENTER>_--><a href="http://theory.chem.umn.edu/">EXIT</a><!-->_</CENTER></FONT>
  </TD>
</TR>
</TABLE>
</HTML>
\end{verbatim}

To check this open your web browser and go to \texttt{http://theory.chem.umn.edu/username/} where \texttt{'username'} is whatever you named the folder your webpage is living in. It ought to look similar to \texttt{http://theory.chem.umn.edu/lo78/} except the picture is different. You now have the beginning of a webpage.

7 Programs

There are a variety of useful programs you will likely run into. I’m not out to write a user manual or tutorial for each of them, just give you an idea of what they do and how to get information on them.

7.1 Make and Configure

\texttt{Make} is a piece of software designed to help compile code. In particular code that has lots of dependencies or where you have specific compiler flags you are testing. It has a lot of application, but in this lab I have only seen it used to compile two kinds of things: \LaTeX
documents and Fortran code. When you do more with LaTeX (see Section 7.4 for more info) or writing code you will get more information about this, but I am going to show you one example with LaTeX.

Let's say you used cvs to check out this document (Section: 3.3). If you open the file “GroupIntro.tex”, you will see this document except that it includes all the LaTeX compiling codes (e.g. \ref, \section, etc). This TeX file must be turned into something user readable and “make” will help us.

First, before we can “make” the document we need to tell it what it will try to make it into. This where the “configure” script comes in. Make sure you have /CVS/Software/Configure checked out of cvs and updated. Now

```
> configure -latex GroupIntro.tex
```

This will tell the make program that “GroupIntro.tex” contains the code you want to compile and you are planning on using LaTeX to compile it. It will write out a new file for you called “Makefile”, which contains all the options and flags needed to compile the document. Now we just need to compile the document. Try

```
> make
```

This will give you a list of options because LaTeX documents can be compiled in different ways. Let's make a pdf.

```
> make ps2pdf
```

This should produce a pdf of this document.

### 7.2 Yum

The Yellow dog Updater, Modified (YUM) is a free RPM manager. Basically, it searches a variety of servers (called repositories) that hold software you might want. What is cool is it will figure out before you download and install something exactly what version you should be using and if it has any other software dependencies and gets those as well. It can only be run as root, as it will be installing programs for you. Here is an example of how it works.

I am trying to install the newest version of AMBER, but when it tries to compile it fails because it can't find a particular library called libXt.so. So I am going to use YUM to find this package. First become root.

```
> su -
```

Now lets have YUM search all the repositories it knows about to see if it can find a package with that library.

```
> yum provides libXt
```

First, YUM will update the repository information, then it will begin listing the packages
that have this file. An example listing looks like this.

```
libXt-devel.i386  1.0.2-3.1.fc6    core
Matched from:
/usr/lib/libXt.so
```

We see the name of the package 'libXt-devel.i386', what version it is '1.0.2-3.1', and what operating system it is for 'fc6'. Notice that it only choose packages that are compatible with my operating system (Fedora Core 6). Then it lists the file that matches what you asked for, 'libXt.so'. I decide that I want that package, so I tell yum to install it.

```
yum install libXt-devel
```

It will first check to see if it is already installed, and if so will just say 'Nothing to do'. Then it will check if having this package requires having any other packages already, and if so will ask you if it can install those also. Just say 'yes' and it should install it all for you. All done.

There is a configuration file for YUM that tells it how to act, /etc/yum.conf. Normally, there is no reason to change this around.

### 7.2.1 Repositories

How did YUM know where to look on the internet to find the software you want? Well if you look in /etc/yum.repos.d/ directory you will find a variety of 'repo' files that tell YUM exactly where to look for what. A 'repo' file contains the name and internet address of a server (a repository) that has software you might want.

Most FC distributions come with a bunch of repositories already setup, but you can add files here to give YUM more places to look. Some examples of other repositories are Fresh RPMs, DAG, RPM Fusion, and Dries. On the front page of all of these websites is instructions on how to enable this repository for you. In general it will be a two step process. First you need a 'repo' file in /etc/yum.repos.d/ that looks something like this.

```
[dries]
name=Extra Fedora rpms dries - $releasever - $basearch
(Note I cut off the last line a bit, don’t try to copy this example.) Second you will new a GPK (gnome package kit) key . On the site will be an example of how to use rpm to get it.
7.3 Cron

“cron” is a program that allows you to schedule things to happen at precise times. This is best explained with an example. Let's say you want to automate the backing up your computer each night like we talked about earlier (Section 4.1.2). We made a bash script to do this and let's assume we put it in `~/bin/`.

The cron program is controlled by “crontab”. Let's see what cron events are already schedules.

```bash
> crontab -l
```

Most likely this will return “no crontab for” you. So let's add one by opening up your personal crontab.

```bash
> crontab -e
```

This will open a vi terminal (5.5). The command usage works like this.

```text
minute hour DoM month DoW user cmd
```

DoM = day of month and DoW = day of week. So let's say we want to back up our drive every night at 11:05pm Monday - Friday every month. This would look like

```text
05 23 * * 1-5 mosera ~/bin/backupscript.sh
```

7.4 LaTeX

configure

7.5 CVS

remove

    import
cvs lock files

8 Computer Code

8.1 Writing Code

8.1.1 Fortran

8.1.2 Perl

reference perl book
8.2 Compiling Code

8.2.1 Compilers

8.2.2 Trouble Shooting

9 Chemistry Software

9.1 Gaussian

9.2 AMBER

9.3 VMD