BSC4933/5936: Introduction to Bioinformatics

Laboratory Section: Tuesdays from 3:45 to 5:45 PM.

BioComputing Basics

Week 1, Tuesday, August 26, 2003

Author and Instructor: Steven M. Thompson

A very basic introduction to biocomputing facilities at Florida State University:

including background information on all computers, fundamentals of the UNIX/Linux operating system and the X environment, client/server computing connections, and simple text editing, as well as a brief introduction to the GCG® Wisconsin Package® and its graphical user interface (GUI) SeqLab®.
Introduction

I write these tutorials from a ‘lowest-common-denominator’ biologist’s perspective. That is, I only assume that you have fundamental molecular biology knowledge, but are relatively inexperienced regarding computers. As a consequence of this they are written quite explicitly. Therefore, if you do exactly what is written, it will work. However, this requires two things: 1) you must read very carefully and not skim over vital steps, and 2) you mustn’t take offense if you already know what I’m discussing. I’m not insulting your intelligence. This also makes the tutorials longer than otherwise necessary. Sorry.

I use three writing conventions in the tutorials, besides my casual style. I use bold type for those commands and keystrokes that you are to type in at your keyboard or for buttons or menus that you are to click in a GUI. I also use bold type for section headings. Screen traces are shown in a ‘typewriter’ style Courier font and “/ / / / / / / / / /” indicates abridged data. The ‘greater-than’ symbol (> ) indicates the system prompt and should not be typed as a part of commands. Really important statements may be underlined.

A computer is an electronic machine that performs rapid, complex calculations, and compiles and correlates data. It is composed of at least five basic parts: a central processor unit (CPU) that performs calculations, a data input device (such as a keyboard and mouse), a data output device (such as a monitor screen and printer), data storage devices (such as hard drives, floppy disks, and compact disks), and random access memory (RAM) where computing processes occur. Other necessary components include networking and graphics modules (boards) as well as the main architecture that it’s all plugged into (the mother board). The quality, size, number, and speed of these components determine the type of computer: personal, workstation, server, mainframe, or super, though the terms are quite ambiguous. Computers have a set of utility programs, called commands, known as an operating system (OS) that enable them to interact with human beings and other programs. OSs come in different ‘flavors’ with the main distinctions related to the company that originally developed the particular OS. Three primary OSs exist today with each having multitudes of variants: Microsoft Windows, Apple Mac OS, and UNIX. While all the various OSs have similar functions, the functions’ names and their execution methods vary from one to another. Most systems have a Graphical User Interface (GUI) to their OS providing mouse driven buttons and menus, and many provide a command line interface as well.

This course will extensively use Linux, a version of the UNIX OS, on a central FSU biocomputing server named Mendel. UNIX was developed in the USA, originally by BELL, then licensed to AT&T and now used in various implementations on many different types of computers the world over. UNIX is a line-oriented system similar to the old MS-DOS OS, though many GUIs exist to help drive it. The UNIX command line interface is often characterized as being very unfriendly compared to other OSs. Actually UNIX is quite straightforward, especially regarding its file systems. UNIX is the precursor of most tree structured file systems including those used by MS-DOS, MS Windows, and the Macintosh OS. These file systems all consist of a tree of directories and subdirectories. The OS allows you to move about within and to manipulate this file system.
An often useful analogy is the file cabinet metaphor — your account is analogous to the entire file cabinet. Your directories are like the drawers of the cabinet, and subdirectories are like hanging folders of files within those drawers. Each hanging folder could have a number of manila folders within it, and so on, on down to individual files. Hopefully all arranged with some sort of logical organizational plan.

Computers these days are most often connected to other computers in a network, particularly in an academic or industrial setting. These networks consist of computers and a high-speed combination of copper and fiber optic cabling. Sometimes more than one computer is networked together into a configuration known as a cluster where computing power can be spread across the individual members of the cluster (nodes). An extreme example of this is called grid computing where the nodes may be spread all over the world. Individual computers are most often networked to larger computers called servers as well as to each other. The worldwide system of interconnected, networked computers is called the Internet. Various software programs enable computers to communicate with one another across the Internet. Graphics-based browsers, such as Microsoft Explorer or Netscape Navigator, designed to access the World Wide Web (WWW), one part of the Internet, are an example of this type of program, but only one of several that you will see today.

Computers only do what they have been programmed to do. Their accuracy entirely depends on the software being used, the data being analyzed, and the manner in which it is used. In scientific biocomputing research, this means that the accuracy and relevancy of your results depends on your understanding of the strengths, weaknesses, and intricacies of both the software employed, and of the biological system being studied.

**My Example Protein System**

I use members of the same dataset throughout the course’s lab tutorial examples to make them more interesting and to provide a common focused objective. You will be doing the same starting next week with your choice of one of four course ‘project’ molecules. It is somewhat analogous to what you would do in an actual laboratory setting and will provide a basic framework on which you can build. My example molecule is the very well characterized and vitally important protein Elongation Factor 1[^1].

The Elongation Factors are a vital protein family crucial to protein biosynthesis. They are ubiquitous to all of cellular life and, in concert with the ribosome, they must have been one of the very earliest enzymatic factories to evolve. Three distinct subtypes of elongation factors all work together to help perform the vital function of protein biosynthesis. In [Eu]Bacteria and Eukaryota nuclear genomes they have the following names (the nomenclature in Archaea has not been completely worked out and is often contradictory):

<table>
<thead>
<tr>
<th>Eukaryota</th>
<th>[Eu]Bacteria</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>EF-1[^1]</td>
<td>EF-Tu</td>
<td>Binds GTP and an aminoacyl-tRNA; delivers the latter to the A site of ribosomes.</td>
</tr>
<tr>
<td>EF-1[^1]</td>
<td>EF-Ts</td>
<td>Interacts with EF-1[^1]/Tu to displace GDP and thus allows the regeneration of GTP-EF-1[^1]/Tu</td>
</tr>
<tr>
<td>EF-2</td>
<td>EF-G</td>
<td>Binds GTP and peptidyl-tRNA and translocates the latter from the A site to the P site.</td>
</tr>
</tbody>
</table>
The Elongation Factor subunit 1-Alpha (EF-1\[a\]) in Eukaryota and most Archaea (called Elongation Factor Tu in [Eu]Bacteria [and Euk’ and Arch’ plastids]) has guanine nucleotide, ribosome, and aminoacyl-tRNA binding sites, and is essential to the universal process of protein biosynthesis, promoting the GTP-dependent binding of aminoacyl-tRNA to the A-site of the intact ribosome. The hydrolysis of GTP to GDP mediates a conformational change in a specific region of the molecule. This region is conserved in both EF-1\[a\]/Tu and EF-2/G and seems to be typical of GTP-dependent proteins which bind non-initiator tRNAs to the ribosome.

In *E. coli* EF-Tu is encoded by a duplicated loci, *tufA* and *tufB* located about 15 minutes apart on the chromosome at positions 74.92 and 90.02 (ECDC). In humans at least twenty loci on seven different chromosomes demonstrate homology to the gene. However, only two of them are potentially active; the remainder appear to be retropseudogenes (Madsen, et al., 1990). It is encoded in both the nucleus and mitochondria and chloroplast genomes in eukaryotes and is a globular, cytoplasmic enzyme in all life forms.

The three-dimensional structure of Elongation Factor 1\[a\]/Tu has been solved in more than fifteen cases. Partial and complete *E. coli* structures have been resolved and deposited in the Protein Data Bank (1EFM, 1ETU, 1DG1, 1EFU, and 1EFC), the complete *Thermus aquaticus* and *Thermus thermophilus* structures have been determined (1TTT, 1EFT, and 1AIP), and even cow EF-1\[a\] has had its structure determined (1D2E). Most of the structures show the protein in complex with its nucleotide ligand, some show the ternary complex. The *Thermus aquaticus* structure is shown below as drawn by NCBI’s Cn3D molecular visualization tool:
Notice that half of the protein has well defined alpha helices and the rest is rather unordered coils partly defined by beta strands. GTP fits right down in amongst all the helices in the pocket. The *Thermus aquaticus* structure has six well-defined helices that occur from residue 24 through 38, 86 through 98, 114 through 126, 144 through 161, 175 through 184, and 194 through 207. There are also two short helices at residues 47 to 51 and 54 to 59. The guanine nucleotide binding site involves residues 18 to 25, residues 81 to 85, and residues 136 to 139. Residue 8 is associated with aminoacyl-tRNA binding.

Because of strong evolutionary pressure resulting in very slow divergence and because of its ubiquity, EF-1α is an appropriate gene on which to estimate early life phylogenies and with which to ask early branching order questions in ‘deep’ eukaryotic evolution. In fact, a series of papers in the early-90’s, notably those by Iwabe, et al. (1989), Rivera and Lake (1992), and Hasegawa, et al. (1993) all base ‘universal’ trees of life on this gene. Iwabe, et al. used the trick of aligning the EF-1α gene parologue EF-1β to their EF-1α dataset to root the tree. Laboratory examples with this protein will illustrate how different biocomputing analyses can lead to different functional and structural insights. I will restrict my example dataset to a subset of ‘primitive’ eukaryotic EF-1α sequences. These will include many protists and algae but will exclude much of the “Crown” group, including all of the higher plants, true fungi, and metazoans.

**Biocomputing Facilities at Florida State University**

Conradi Computer Teaching Laboratories are equipped with Macintosh and Microsoft (MS) Windows personal computers. The CSIT Teaching Lab has UNIX (Linux) workstations. Any system can be used for this Laboratory Course as long as a few key communication programs are installed. Most all systems will have some type of a [WWW browser](http://www.mozilla.org) available, be it Explorer, Navigator, Mozilla, Konqueror, Safari, or Opera; it doesn’t matter. You can use whatever is on the machine. Unfortunately a Web browser alone is not enough for serious biocomputing. More often than not you will need to directly connect to a server computer using a command line, “terminal,” window where you can directly interact with the server computer’s OS. The ‘old way’ to do this was with a common program called telnet. However, telnet is an un-secure program from which smart hackers can ‘sniff’ connection account names and passwords. Therefore, in this age of the hacker, most server computers no longer allow telnet connections. A newer program named ssh, for secure shell, encrypts all connections and is now required for command line access to most servers. Ssh comes preinstalled as a part of all modern UNIX OSs but doesn’t come with pre OS X Macs or any MS Windows machines and, therefore, will have to be installed on those platforms in order to be used for the course. Ssh is available on both the Macs and the MS Windows machines in the Conradi Labs and on all CSIT machines.

Along the lines of secure connections, there are often times when you’ll need to move files back and forth between your own computer and a server computer located somewhere else. The ‘old’ un-secure way of doing this was a program named ftp, for file transfer protocol. Just like telnet it has the unfortunate attribute of allowing hackers to ‘sniff’ account names and passwords. Therefore, an encrypted file transfer counterpart to ssh is now required by most servers. That counterpart is called [sftp](http://www.openssh.org) and [scp](http://tools.ietf.org/html/rfc4251), for secure file transfer protocol.
and secure copy. It's also included in all modern UNIX OSs but not in pre OS X Macs nor in MS Windows. An implementation of these programs is also available on all the Conradi and CSIT computers.

Furthermore, since ssh is strictly a non-graphical terminal program, and since all Web browsers’ graphics capability is inadequate for the truly interactive graphics that much biocomputing software requires, another type of graphical system needs to be present on the computer that you use for this course. That graphical interface is called the X Window System. It was developed at MIT (the Massachusetts Institute of Technology) in the 1980’s, back in the early days of UNIX, as a distributed, hardware independent way of exchanging graphical information between different UNIX computers. Unfortunately the X worldview is a bit backwards from the standard client/server computing model. In the standard model a local client, for instance a Web browser, displays information from a remote server, for instance a particular WWW site, also called a Uniform Resource Locator (URL). In the world of X, an X-server program on the machine that you are sitting at (the local machine) displays the graphics from an X-client program that could be located on either your own machine or on a remote server machine that you are connected to. Confused yet?

Nearly all UNIX computers, including Linux, but not including Mac OS X, include a genuine X Window System in their default configuration. MS Windows computers, including the ones in the Conradi Lab, are often loaded with X-server emulation software, such as the commercial program XWin32 or eXceed, to provide X-server functionality. Macintosh computers prior to OS X required a commercial X solution; often the program MacX or eXodus was used. However, since OS X Macs are true UNIX machines, they can use a variety of free public domain packages to provide true X Windowing. This is being done on the Conradi Lab Macs.

Florida State University’s main biocomputing server for sequence analysis is a Dell PowerEdge 6650 named Mendel bought with Howard Hughes Medical Institute monies from the recently successful Biology Department undergraduate education proposal. Mendel has four 1.6 GH Intel Xeon CPUs, eight GB of RAM, and over 700 GB of storage. This machine (mendel.csit.fsu.edu) is managed by, and located in, the School of Computational Science and Information Technology (CSIT), and runs RedHat Linux version 7.2. RedHat Linux is a commercial distribution of the free UNIX derived, Open Source Linux OS. Linux was invented in the early 1990’s by a student at the University of Helsinki in Finland named Linus Torvalds as a ‘hobby.’ Mendel only allows ssh, scp, and sftp connections. In order to display X Windows on your local computer you will need to allow ssh X tunneling. While in this course you will learn what this means and how to use much of the biocomputing software installed on Mendel. You have been issued an account on this machine by merit of your enrollment in this course. Other local servers that we will connect to today include the URLs http://bio.fsu.edu/ and http://www.csit.fsu.edu/. Becoming familiar with these three machines is the main objective of today’s lab.

**Week One Tutorial: Exploring the FSU Biocomputing Environment**

**WWW Browsers and other Local Programs**
So how do you do bioinformatics? Often bioinformatics is done on the Internet through the WWW. This is possible and easy and fun, but, beside being a bit too easy too get sidetracked upon, the Web can not readily handle large datasets or large multiple sequence alignments. These sort of datasets quickly become intractable in the Web environment. You’ll know you’re there when you try. In spite of that, let’s begin with a Web resource designed specifically for this course.

Activate the computer that you are sitting at by moving its mouse or by pressing the return key on its keyboard. You may then have to log onto it with an appropriate user ID and password. This is not your Mendel account; it is whatever account gives you access to that lab’s computers. If you don’t know this information, contact a lab instructor as soon as possible. Now launch a Web browser by selecting the appropriate icon. If the icon is on the desktop, a <double-click> will launch it, if it’s in a Mac, MS Window, or UNIX menu, a single <click> will do it. When the Web browser has opened go to the Course Web page, http://bio.fsu.edu/~stevet/BSC5936.html, and select the link that connects to the BSC596 Laboratory, http://bio.fsu.edu/~stevet/BSC5936/Lab.syllabus.html page. You’ll see something like the following Safari window snapshot:
This page provides links to each week’s exercise tutorial and lab report throughout the semester. Explore the navigation links in the sidebar. Notice the different remote Web server computers that different URLs connect to. The computer’s Internet name is that portion of the URL just after the http:// and before the next slash (/), for instance, bio.fsu.edu for my Course pages. Go to my Home page, http://bio.fsu.edu/~stevet/cv.html, and check out the biocomputing Bookmark list there. Don’t spend too much time exploring Web resources right now though, just realize that a huge spectrum is available. Next week we will explore several of these sites in much greater detail while learning about many of the biocomputing databases available.

So what are alternatives to Web based biocomputing? Desktop biocomputing software solutions can be installed on your own personal computer. A client-server protocol different than a Web browser/server, Network Entrez from the National Center for Biotechnology Information (NCBI), provides good text search based access to the NCBI databases, though it is no longer supported. Other free, public domain sequence analysis programs are also available, but they can be complicated to install, configure, and maintain. The user must be pretty computer savvy, especially to get them to all cooperate with one another. So, commercial software packages, such as MacVector, DNAsis, DNASTar, etc., are available, but license hassles, a big expense per machine, and database access all complicate matters. Therefore, non-Web, server-based solutions are often employed. These require network access to UNIX server computers. A big advantage of this solution is access to fast, powerful programs with convenient database access, all on the same server. Connections can be made from any networked computer anywhere! Again free public domain solutions are available, but now a very cooperative systems administrator must build and maintain the system for its users. However, with a commercial biocomputing server package there are minimum systems management concerns, and only one license fee for the entire institution, rather than individual licenses for every machine running the package. From an economics point of view, it’s a ‘no-brainer’ for an institution to support a commercial server based biocomputing solution.

**Mendel — FSU’s Biocomputing Server**

Therefore, let’s begin to explore the UNIX server biocomputing world. On pre-OS X Macs or MS Windows machines find and use the appropriate icon to launch **ssh**, either on the desktop or in menus. After ssh opens, use the appropriate menu command to connect to mendel.csit.fsu.edu. You’ll be asked for a user name and a password. Note that the password is not displayed on the screen as you type it. This user name and password is different than the one previously used to get on the teaching lab computer system. As stated in the Introduction, these Mendel accounts were newly established for you when you registered for the course. See your instructor for the logon information. Write down your user name somewhere where you won’t forget! (We’ll be changing the password below, so you’ll be writing it down a bit later.)

On a UNIX (or Linux) or Mac OS X machine launch an X-terminal program window with the appropriate icon from the desktop or menus and issue the command **“ssh -X user@mendel.csit.fsu.edu”** not typing the quotes and replacing “user” with your account name on Mendel. You’ll be asked for your Mendel
password. The “-x” (note that the -X is capitalized!) option is necessary to allow ‘X tunneling’ and set up your X environment. This is the only encrypted, secure way to make X connections, and is required by Mendel, if you want to use any resources that require X windows. If you are using a GUI version of ssh, then the X tunneling option should be turned on by default. Further details of X on Mendel will not be fully covered in this tutorial. There are too many variables depending on your local machine — we’ll just go over the key concepts. If this isn’t enough, ask me for further assistance. I’m also available for personal help in your own laboratories. If you are having trouble using Mendel from there, just contact me at stevet@bio.fsu.edu.

Regardless of the ssh method used to launch and connect to Mendel, you should now have an interactive command line terminal session running on Mendel in a separate window on your local machine’s desktop. Mendel’s OS checked your username and password, and if correct, ran your default shell program and any startup scripts and then returned the system prompt. The shell program is your interface to the UNIX OS. It interprets and executes the commands that you type. Common UNIX shells include bash, Korn, the C shell, and a popular C shell derivative that Mendel users run by default called tcsh. Tcsh, like bash, enables command history recall using the keyboard arrow keys, accepts tab word completion, and allows command line editing. Upon logging in, you end up in your ‘home directory,’ that portion of Mendel’s hard drive disk space reserved just for you and designated by you from anywhere on the system with the character string “$HOME”. This is called an “environment variable.” You should see a screen trace similar to the following upon logging in:

Welcome to the WISCONSIN PACKAGE
Version 10.3-UNIX
Installed on linux

Copyright (c) 1982 – 2001, Accelrys Inc.
A wholly owned subsidiary of Pharmacopeia, Inc. All rights reserved.

Published research assisted by this software should cite:
Wisconsin Package Version 10.3, Accelrys Inc., San Diego, CA

Databases available:
GenBank/GenBank Tags Release 135.0 (4/2003)
GenPept translations Release 135.0 (4/2003)
Genome (H. sapiens) Build 33.0 (4/2003)
NRL_3D PDB sequences Release 28.0 (9/2000)
PIR-Protein Release 76.0 (3/2003)
Swiss-Prot Release 17.2 (7/2002)
Restriction Enzymes (REBASE) 10.0 (10/2000)

Technical support see: http://www.accelrys.com/support/

Online help: % genhelp or genmanual or
http://www.csit.fsu.edu/gcg/ with off-campus
restricted access: user - gcg, password - stevet

thompson@mendel >

The screen trace shows the version numbers for the Wisconsin Package (much more on this topic later today) and of all its online databases on Mendel. The system prompt displays the user and the machine name and waits to receive a command. On different UNIX systems the prompt may appear different ways depending on
how the system administrator has set up the user environment. Here I will just use the 'greater than' sign (>) to represent the system prompt. It should not be typed as part of the command.

In command line mode each command is terminated by the 'return' or 'enter' key. UNIX uses the ASCII character set and unlike some OSs, it supports both upper and lower case. A disadvantage of using both upper and lower case is that commands and file names must be typed in the correct case. Most UNIX commands and file names are in lower case. Commands and file names should not include spaces nor any punctuation other than periods (.), hyphens (–), or underscores (_). UNIX command options are specified by a required space and the hyphen character (–). UNIX does not use or directly support function keys. Special functions are generally invoked using the 'Control' key. For example a running command can be aborted by pressing the "Control" key [sometimes labeled “CTRL” or denoted with the karat symbol (^)] and the letter "c." The short form for this is generally written CTRL-C or ^C. Using control keys instead of special function keys for special commands is sometimes difficult to remember, the advantage is that nearly every terminal program supports the control key, allowing UNIX to be used from a wide variety of different platforms that might connect to the server.

The general command syntax for UNIX is a command followed by some options, and then some parameters. If a command reads input, the default input for the command will generally come from the interactive terminal. The output from a system level command (if any) will generally be printed to your terminal. The command syntax allows the input and outputs for a program to be redirected into a file or the output of one program can be passed as the input to another program. General command syntax follows:

```
cmd
    cmd -options
    cmd -options parameters
```

To cause a command to read from a file rather than from the terminal, the “<” sign is used on the command line and the “>” sign causes the program to write its output to a file (for those programs that do not do this by default):

```
cmd -options parameters < input
    cmd -options parameters > output
    cmd -options parameters < input > output
```

To cause the output from one program to be passed to another program as input a vertical bar (|), known as the “pipe,” is used.

```
cmd1 -options parameters | cmd2
```

This feature is called “piping” the output of one program into the input of another.
Certain printing (non-control) characters, called “shell metacharacters,” have special meanings to the UNIX shell. You rarely type shell metacharacters on the command line because they are punctuation characters. However, if you need to specify a filename accidentally containing one, turn off its special meaning by preceding the metacharacter with a “\” (backslash) character or enclose them in “’” (single quotes). The metacharacters “*” (asterisk), “?” (question mark), and “~” (tilde) are used for the shell file name “globbing” facility. It’s wrong to use a globbing character in the absence of any matching files. When the shell encounters a command line with a leading “~”, or with “*” or “?” anywhere on the command line, it attempts to expand that word to a list of matching file names using the following rules: A leading “~” expands to the home directory of a particular user. Each “*” is interpreted as a specification for zero or more of any character. Each “?” is interpreted as a specification for exactly one of any character. For example, the pattern “dog*” will access any file that begins with the word dog, regardless of what follows. It will find matches for, among others, files named “dog,” “doggone,” and “doggy.” The pattern “d?g” matches dog, dig, and dug but not ding, dang, or dogs; “dog?” finds files named “dogs” but not “dog” or “doggy.” This is known as using a “wild card.” Generally when a UNIX command expects a file name, “cmd  filename,” it’s possible to specify a group of files using a wild card expression. Two globbing shell metacharacters cause wild card ‘expansion’:

* matches any string of characters zero or longer,

? matches any single character.

A couple of examples using wild card characters along with the pipe and output redirection follow:

```bash
cmd */*.data | cmd2
(cmd */my.data? > filename
```

The first example will access all files ending in “.data” in all subdirectories one level below the current directory and pass that output on to the second command. The second example will access all files named “my.data” that have any single character after the word data in all subdirectories one level below your current directory and output that result to a file named filename. Wild cards are very flexible in UNIX and this makes them very powerful, but you must be extremely careful when using them with destructive commands like “rm” (remove file).

Getting help in any OS can be very important. UNIX provides a text-based help system called man pages. You use man pages by typing the command “man” followed by the name of the command that you want help on. Before moving any further into UNIX, let’s change our passwords from the initial ones you were given. Give the command “man passwd” to see how the man pages work and read about passwd. Press the space bar to page through the man pages: type the letter “q” for quit to get back to your command prompt. Now issue the “passwd” command to change your password. As before, you will not see the passwords as you type them, and you may have to choose a different password if Mendel doesn’t like your choice. Follow the onscreen directions:
Following are three examples of the *ls* command in my account:

```
    thompson@mendel > passwd
    Changing password for thompson.
    Old password:
    New password:
    Retype new password:
```

Write down your new password along beside your Mendel account name. Don’t forget!

When an account is created, your home directory environment variable, “`$HOME`,” is created and associated with that account. In any tree structured file system the concept of where you are in the tree is very important. There are two ways of specifying where things are. You can refer to things relative to your current directory or by its complete `path` name. When the complete path name is given by beginning the specification with a slash, the current position in the directory tree is ignored. To find the complete path in Mendel’s file system to your home directory type the command “`pwd`.”

```
    thompson@mendel > pwd
    /home/thompson
```

This UNIX command shows you where you are presently located on the server. It displays the complete UNIX path specification (this always starts with a slash) for the directory structure of your account. Also notice that UNIX uses forward slashes (/) to differentiate between subdirectories, not backward slashes (\) like MS-DOS. The *pwd* command can be used at any point to keep track of your location. Several commands for working with your directory structure follow:

```
pwd       print working directory  
ls         list the contents of the directory  
mkdir      make a new directory  
cd         change directory
```

To list the files in your home directory, use the “`ls`” command. There are many options to the *ls* command. Check them out by typing “`man ls`”. The most useful options are the “`-l`” option and the “`-a`” options. The command “`ls -l`” will list the files and directories in your current directory in the ‘long’ form with extended information. A UNIX convention is that files with a period as the first character in their name are not listed by the *ls* command unless the “`-a`” ‘all’ option is given.

This convention has lead to a number of special configuration files having periods as the first character in their name. Some of these files are executed automatically when a user logs in, much like “`AUTOEXEC.BAT`” and “`CONFIG.SYS`” are executed in MS-DOS upon log in. On many UNIX systems there is a file executed upon every login called “`.login`” and another one that sets up the shell environment called “`.cshrc`”. In general you do not want to mess with these files in your account until you are very comfortable with the OS. Following are three examples of the *ls* command in my account:

```
thompson@mendel > ls
    bin   gcg   mail   patterns   seqlab   temp.epsf   tutorials
```
In the output from "ls –l" additional information regarding the file permissions, owner of the file, size, modification date, and file name is shown. In the output from "ls –a" those 'dot' system files are now seen. Nearly all OSs have some way to customize your login environment with editable configuration files; these are them. The experienced user can place commands in these special files to customize their individual login environment.

Subdirectories are generally used to group files associated with one particular project or files of a particular type. For example, you might store all of your memorandums in a directory called "memo." The "mkdir" command is used to create directories and the "cd" command is used to move into directories. The special placeholder file ".." allows you to move back up the directory tree. Note its use below with the cd command to go back up to the parent of the current directory:
After the “cd ..” command pwd shows that we are ‘back’ in the home directory. Next we’ll look at several commands that deal with files, rather than directories:

- **rm**: remove (delete) a file,
- **mv**: move (rename) a file,
- **cp**: copy a file to another file or a file or set of files into a directory,
- **more or less**: page through a file, moving from one page to another with the space bar.

Below are some examples of these commands, and of command redirection and piping with ls and more to allow paging through directory listings. Issue the following commands:

```bash
> ls -la | more
> ls -la /usr/X11R6/bin > tmp
> more tmp
> cp tmp memo/tmp.out
> mv tmp tmp.txt
> rm memo/tmp.out
```

A very useful command that allows searching through files for a pattern is called grep. The first parameter to `grep` is a search pattern; the second is the file or files that you want searched. For example if you had a bunch of different data files whose file names all ended with the word data in several different subdirectories and wanted to find the one that mentioned zebra, you could use the following command:

```
grep zebra */*data
```

After doing the above steps (remember — do anything in bold) please read over and experiment with the commands in bold in the following tables.

**Important UNIX Commands and Keystroke Conventions:**

- `< .>`: Current working directory.
- `< ..>`: Parent directory of current working directory.
- `< ~>`: User’s home directory (C shell and tcsh only, also `$HOME`).
- `< & >`: Execute the specified command in another process.

Most commands have on-line documentation available through the man pages:

- `man tcsh`: Pages through the manual page on the tcsh shell.
- `man -k batch`: Gets you the title lines for every command with the word batch in the title.

Command to change your password:

```
passwd
```

Change your login password
Commands for looking at the system, other users, your login sessions, jobs you are running, and command execution:

**uptime**
Shows the time since the system was last rebooted. Also shows the "load average". Load average indicates the number of jobs in the system ready to run. The higher the load average the slower the system will run.

**w or who**
Shows who is logged in to the system doing what.

**top**
Shows the most active processes on the entire machine and the portion of CPU cycles assigned to running processes. Press "q" to quit.

**ps**
Shows your current processes and their status (running, sleeping, idle, terminated, etc.); (use the man ps pages as options widely vary, see especially the a, e, l, f, u, and U options).

**at**
Submit script to the at queue for execution later.

**bg**
Resumes a suspended job in background mode.

**fg**
Brings a background job back into interactive mode.

The following commands affect the file system and access files. The basic file commands:

**cat tmp.txt**
Shows the contents of the file "tmp.txt" on your screen, also concatenates files, for example: "cat file1 file2 > file3."

**more tmp.txt**
Shows the contents of the file "tmp.txt" at the terminal one page at a time; press the space bar to continue. Type a "?" when the scrolling stops for viewing options (less often available; it is more powerful than more).

**head tmp.txt**
Shows the first few lines of the file "tmp.txt."

**tail tmp.txt**
Show the last few lines of the file "tmp.txt."

**grep xterm tmp.txt**
Show the lines in the "tmp.txt" that contain the specified pattern, here the word "xterm."

**wc tmp.txt**
Counts the number of characters, words, and lines in the file "tmp.txt."

**cp tmp.txt tmp**
Copies the file "tmp.txt" to the file "tmp." Any previous contents of the file "tmp" are lost.

**mv tmp.txt tmp**
Renames (moves) the file "tmp.txt" to the file "tmp." Any previous contents of the file "tmp" are lost.

**mv tmp memo**
Since "memo" is a directory name not a file name, this command moves the specified file, "tmp," into the specified directory, "memo," keeping the original file name intact.

**rm memo/tmp**
Deletes (removes) the file "tmp" in the directory "memo." It is unrecoverable!

**chmod perm**
Changes the permissions of a file. See "man chmod" and also "man chown" for further details.
lpr file  
Prints the specified file on the default system printer. Will need to specify a particular print queue with the “-P” option to send it elsewhere.

Directory commands:

pwd  
Print Working Directory. Shows you where you are at in the file system. Very useful when you get confused. (Also see "whoami" if really confused!)

ls  
Shows (lists) your files' names.

ls -l  
Shows your files’ names in extended (long) format including file size, ownership, and permissions.

ls -al  
Shows all files including the system files (.files) in your directory in the long format.

mkdir newdir  
Makes a new directory in your current directory.

rmdir newdir  
Removes a subdirectory from your current directory. Directory must be empty to remove the directory.

rm -r dir  
Removes all the files, and subdirectories of a directory and then removes the directory. Very convenient, useful and dangerous.

cd  
Move back into your home directory from anywhere.

cd memo  
Move down into a directory named “memo” from your current directory.

Usually it is best to leave programs using the quit or exit commands; however, occasionally it is necessary to terminate a running program. Here are some useful commands for doing this. Commands for bailing out of programs:

<Ctrl c>  
Aborts a running process (program); no option for restarting it later.

<Ctrl d>  
Terminates a UNIX shell, i.e. exit present control level and close the file. Use “logout” or “exit” to exit from your top level login shell.

<Ctrl z>  
Pauses (suspends) a running process and returns the user to the system prompt. The suspended program can be restarted by typing “fg” (foreground). If you type “bg” (background) the job will also be started again, but in background mode.

kill –9 psid  
Kills a process with the given process ID using the “sure kill” option. This number is obtained using some variation of the ps command.

I discussed Mendel's specific networking requirements earlier. The following commands provide simple access to a subset of UNIX networking capabilities (host refers to a computer's fully qualified Internet name or number, e.g. zen.art.motorcycle.com or 999.999.99.99):
ftp host

File transfer protocol. Allows a limited set of commands (dir, cd, put, get, help, etc.) for moving files between machines. Note: un-secure method, so often restricted to “anonymous ftp” only. See sftp and scp as an alternative.

scp

Secure copy file, syntax: “scp file user@host:path” or “scp user@host:path file.” Good for moving a few files.

sftp

Secure file transfer protocol. Allows same subset of commands as ftp, but through an encrypted connection. Good for moving lots of files.

telnet host

Provides an un-secure terminal connection to another Internet connected host (discouraged and often disabled!). See ssh for a secure alternative.

ssh user@host

Connect to a host computer using a secure, encrypted protocol.

Text Editing

Text editing is often a necessary part of computing. This is never that much fun, but always very important. You can use your own favorite word processor like MS Word, if you would like, but be sure to “Save As” “Text Only” with “Line Breaks!” Native word processing format contains much binary control data in it specifying format and so forth; the UNIX OS can’t read it. Saving as text avoids this problem. Editing this way is a two-step process though. After the editing is done, the file needs to be transferred to the UNIX server with scp or sftp. Therefore, it makes sense to get comfortable with at least one UNIX text editor. That will avoid the file transfer step, saving some hassle. There are several around, including many driven though a GUI, but minimally I recommend learning pico. It’s description, along with two alternatives follow:

pico newfile

A text editor provided with the pine mailer; appropriate for general text editing but not present on all UNIX systems. This is a simple to use editor with a command banner presenting a menu of Ctrl Key command options. Type in your text and then press Ctrl-X to exit and save “newfile.”

vi file

The default UNIX text editor. This comes with all versions of UNIX and is extremely powerful, but it is quite difficult to master. I recommend avoiding it entirely unless you are interested in becoming a UNIX expert.

emacs file

This is a very nice alternative text editor available on many UNIX machines. This editor is also quite powerful but not nearly as difficult to learn as vi.

UNIX is not the easiest computer OS to learn. Have patience, ask questions, and don’t get down on yourself just because it doesn’t seem as easy as some OSs. The power and flexibility of UNIX is worth the extra effort. Plus, UNIX is the de facto standard OS for most scientific computing, so the effort will not be wasted.

After going through the above commands, especially trying out the ones in bold, it’s time to see the primary sequence analysis biocomputing package used in this course. That package is called the Accelrys GCG Wisconsin Package. Its birth and growth are a real success story among the death of so many dot.coms and
biotech endeavors of the last several years. You saw their introductory welcome screen seen when you first logged on to Mendel. That process also initialized the Package's user environment.

The Genetics Computer Group

The Wisconsin Package for Sequence Analysis began as a service project in 1982 in Oliver Smithies’ lab in the Genetics Department at the University of Wisconsin, Madison. It spun off that effort into a University Research Park location becoming an independent private company, the Genetics Computer Group (GCG), in 1990. Then in 1997 the Oxford Molecular Group of Great Britain, a chemical informatics company, acquired GCG. The drug discovery and development firm Pharmacopeia next purchased GCG, and the other Oxford Molecular holdings, in late 2000. Finally, in summer 2001, it, along with Pharmacopeia’s other software holdings, were all placed under the new corporate name Accelrys, Inc., which became a subsidiary of Pharmacopeia. For more information on Pharmacopeia, Accelrys, and the Wisconsin Package’s history see the links at http://www.accelrys.com/about/gcg.html.

The Wisconsin Package has arguably become the global ‘industry-standard’ in sequence analysis software. It provides a comprehensive suite of nearly 150 integrated DNA and protein analysis programs, from database, pattern, and motif searching; fragment assembly; mapping; and sequence comparison; to gene finding; protein and evolutionary analysis; primer selection; and DNA and RNA secondary structure prediction. The package’s programs work together in a "toolbox" fashion. Much like a carpenter’s toolbox where using the right tool correctly in the right order can build a house, several relatively simple programs properly used in succession can lead to sophisticated sequence analysis results with the Wisconsin Package. Furthermore, the programs are 'internally compatible.' By this I mean that once you learn how to use one program, how the programs ‘look and feel,’ then you pretty much know how to run all of the programs, since all 'act' similarly, and, most importantly, the output from many programs can be used as input for other programs. This is how you use the programs in a logical succession. The present version of the Wisconsin Package only runs on computers running the UNIX operating system, but it can be accessed from any networked terminal. More than 30,000 scientists worldwide at over 500 institutions in more than 30 countries use the package, so learning it here will most likely be useful at any other research institution you may end up at.

SeqLab is included as a part of the GCG Wisconsin Package standard distribution. This powerful X Windows based GUI is a ‘front-end’ to the package. It provides an intuitive alternative to the UNIX command line by allowing menu-driven access to most of GCG’s programs. SeqLab makes running the Wisconsin Package much easier by providing a common editing interface from which most programs can be launched and alignments can be manipulated. SeqLab originated way before it had anything to do with GCG. Steve Smith was working on bacterial ribosomal RNA phylogenies with Walter Gilbert and Carl Woese. Steve realized the vital need for a comprehensive multiple sequence editor. Nothing existed at the time that satisfied him, so he invented one. In addition to providing the vital editing function, it also served as a menuing system to external functions such as the PHYLIP molecular evolution programs and Clustal alignments. He called it the “Genetic
Data Environment: GDE" (1994). Many people were very impressed and he made it freely available. Coincidentally GCG realized the need for some sort of a ‘point-and-click’ environment for their system. They were losing lots of business, only being able to provide a command line interface. Therefore, they started trying to develop a GUI for the Wisconsin Package and released it in 1994. They called it the “Wisconsin Package Interface,” WPI for short. Few were impressed. It only provided a menu to their programs, hardly anything more than the “-check” command line option they've always had. So they did a natural and very smart thing. They hired Steve Smith away from Millipore, where he had recently moved, into their company, so that he could merge his GDE with their WPI. The late 1996 offspring was SeqLab, and, thank goodness, they threw away the acronyms (GDE + WPI = SeqLab). As ‘they’ say “The rest is history” and once more GCG’s customers are (generally) happy. Even though it’s not perfect, once you gain an appreciation for SeqLab’s power and ease of use, I don’t think you’ll be satisfied with any other sequence analysis system. I know I’m not.

**Using the Wisconsin Package — Specifying Sequences and Logical Terms!**

Before launching into SeqLab, let’s go over an important, central ‘idea’ of the Wisconsin Package. One of the most difficult aspects of the Package for new users to get used to is how to tell the programs what sequences you want to work with. GCG calls this “specifying sequences” and it’s crucial to understanding the way their programs work. Once you’ve become comfortable with these concepts, so many of the frustrations commonly encountered with the Package will disappear. So, to answer the always perplexing GCG question “What sequence(s)? . . .” the four ways of specifying sequences, in order of increasing power and complexity:

1. The sequence is in a local GCG format single sequence file in your UNIX account. This sequence file can be anywhere in your account as long as you supply an appropriate path so that the program can find the file. The sequence file can have any name but it is best to use extensions that tell you what type of molecule it is, e.g. “.seq” and “.pep” (e.g. “my.pep” or “~/subdir/my.seg”). Use the program reformat to convert ‘raw’ text format files to GCG format. Several GCG From and To programs are also available for specific data format conversions, and SeqLab’s Editor Mode can directly “Import” native GenBank and ABI style trace format files without the need to reformat.

   ```
   ACTGAGCTCATAGCAACTACGAGTTTACCAAGATATACAGATTTAATAGCTGATCCCATGGGA
   .
   `
   ```

   Next, the clean GCG format single sequence file after reformat:

   ```
   ACTGAGCTCATACTACGAGTTTACCAAGATATACAGATTTAATAGCTGATCCCATGGGA
   ..
   ```

   This is a small example of GCG single sequence format. Always put some documentation on top, so in the future you can figure out what it is you're dealing with! The line with the two periods is converted to the checksum line.
2. The sequence is in a local GCG database (much more about molecular biology databases next week) in which case you ‘point’ to it by using any of the GCG database logical names. These names make sense and are either the name of the database or an abbreviation thereof. Subcategory logical names based on GenBank divisions can be used for nucleotide databases, such as Bacterial. FSU GCG logical database names are listed below. Take a look but don’t worry too much about the specifics right now, we’ll go over them in detail next week. A colon, (:) always sets the logical name apart from either an accession code or a proper identifier name or a wildcard expression, and unlike the rest of UNIX, the whole thing is case insensitive. Several examples follow: GenBank:EctufBT, gb:x57091, SwissProt:EFTu_Ecoli, sw:p02990, PIR:EFicTa, and p:a91475 all refer to the elongation factor Tu in E. coli, DNA in the first two, protein in the last four. If you know that the database uses consistent naming conventions, then you can use a wild card to specify all of a particular type of sequence. This works particularly well in SwissProt because of its consistent naming conventions; e.g. SW:EFTu.* specifies all of the EF-Tu sequences in SwissProt, SW:*Ecoli specifies all of the E. coli sequences in SwissProt.

Because all the sequences are available in the local GCG databases, it is seldom necessary to put individual sequences in your account. In fact, the only time that you should put individual database sequences in your account is if you are somehow modifying them, such as making an alignment or engineering a vector. There’s no need to fill up your own account with data available on the same server outside your account.

Logical Terms for the Wisconsin Package at FSU

Sequence databases, nucleic acids:

<table>
<thead>
<tr>
<th>Logical Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GENBANKPLUS:*</td>
<td>all of GenBank plus EST and GSS</td>
</tr>
<tr>
<td>GBP:*</td>
<td>all of GenBank plus EST and GSS</td>
</tr>
<tr>
<td>GENBANK:*</td>
<td>all of GenBank except EST and GSS</td>
</tr>
<tr>
<td>GB:*</td>
<td>all of GenBank except EST and GSS</td>
</tr>
<tr>
<td>BA:*</td>
<td>GenBank bacterial</td>
</tr>
<tr>
<td>BACTERIAL:*</td>
<td>GenBank bacterial</td>
</tr>
<tr>
<td>EST:*</td>
<td>GenBank EST (Expressed Sequence Tags)</td>
</tr>
<tr>
<td>GSS:*</td>
<td>GenBank GSS (Genome Survey Sequences)</td>
</tr>
<tr>
<td>HTC:*</td>
<td>GenBank High Throughput cDNA</td>
</tr>
<tr>
<td>HTG:*</td>
<td>GenBank High Throughput Genomic</td>
</tr>
<tr>
<td>IN:*</td>
<td>GenBank invertebrate</td>
</tr>
<tr>
<td>INVERTEBRATE:*</td>
<td>GenBank invertebrate</td>
</tr>
<tr>
<td>OM:*</td>
<td>GenBank other mammalian</td>
</tr>
<tr>
<td>OTHER_MAMMALIAN:*</td>
<td>GenBank other mammalian</td>
</tr>
<tr>
<td>OV:*</td>
<td>GenBank other vertebrate</td>
</tr>
<tr>
<td>OTHER_VERTEBRATE:*</td>
<td>GenBank other vertebrate</td>
</tr>
<tr>
<td>PAT:*</td>
<td>GenBank patent</td>
</tr>
<tr>
<td>PATENT:*</td>
<td>GenBank patent</td>
</tr>
<tr>
<td>UN:*</td>
<td>GenBank unannotated</td>
</tr>
<tr>
<td>UNANOTATED:*</td>
<td>GenBank unannotated</td>
</tr>
<tr>
<td>VI:*</td>
<td>GenBank viral subdivision</td>
</tr>
<tr>
<td>VIRAL:*</td>
<td>GenBank viral subdivision</td>
</tr>
<tr>
<td>GENOME:*</td>
<td>NCBI RefSeq working draft</td>
</tr>
</tbody>
</table>

Human Genome sequence, nucleic acids

Sequence databases, amino acids:

<table>
<thead>
<tr>
<th>Logical Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SW:*</td>
<td>all of Swiss-Prot (fully annotated)</td>
</tr>
<tr>
<td>SPTREML:*</td>
<td>Swiss-Prot preliminary EMBL translations</td>
</tr>
<tr>
<td>SPT:*</td>
<td>Swiss-Prot preliminary EMBL translations</td>
</tr>
<tr>
<td>P:*</td>
<td>all of PIR Protein</td>
</tr>
<tr>
<td>PIR:*</td>
<td>all of PIR Protein</td>
</tr>
<tr>
<td>PROTEIN:*</td>
<td>PIR fully annotated</td>
</tr>
<tr>
<td>PIR1:*</td>
<td>PIR fully annotated</td>
</tr>
</tbody>
</table>
3. The sequence is in a GCG format multiple sequence file, either an MSF (multiple sequence format) file or an RSF (rich sequence format) file. The difference is that MSF files contain only the sequence names and sequence characters, whereas RSF files contain sequence names and data, plus sequence annotation; i.e. they are “richer.” As in GCG single sequence format, it is always best to retain the suggested GCG extensions, msf or rsf, in order for you to easily recognize what type of file they are without having to look, though it is not required and they could just as well be named Joe.Blow. To specify sequences contained in a GCG multiple sequence file, supply the file name followed by a pair of braces, “{},” containing the desired sequence specification. For example, to specify all of the sequences in an alignment of elongation 1 and Tu factors, you could use a naming system like the following: ef1a-tu.msf(*). Furthermore, one can point to individual members of the alignment or subgroups by specifying their name within the braces, e.g. EF1a-Tu.rsf(eftu_ecoli) to point just to the E coli sequence or EF1a-Tu.rsf(eftu_*) to point at all of the EfTu’s as long as you use a sequence naming convention that retains this convention.

4. Finally, the most powerful method of specifying sequences, and the ‘way’ that SeqLab works, is in a GCG “list” file. This file can have any name though it is convenient to use the GCG extension “.list” to help identify them in your directory. It is merely a list of any other sequence specifications and can even contain other list files within it. List files can be created by hand with an editor and they are produced by many of the GCG programs, such as all the search programs. This is how the output from one program can become input to another. The convention to use a GCG list file in a program is to precede it with an at sign (@). Furthermore, you can supply attribute information within list files to specify something special about the sequence. This is especially helpful with length attributes that can restrict an analysis to specific portions of a sequence and are shown in the example below:

| PH:* | GenBank phage | PIR2:* | PIR preliminary |
| PHAGE:* | GenBank phage | PIR3:* | PIR unverified |
| PL:* | GenBank plant | PIR4:* | PIR unencoded |
| PLANT:* | GenBank plant | NRL_3D:* | PDB 3D protein sequences |
| PR:* | GenBank primate | NRL:* | PDB 3D protein sequences |
| PRIMATE:* | GenBank primate | RSF | GenBank synthetic |
| RO:* | GenBank rodent | GENRUNDATA | path to GCG default data files |
| RODENT:* | GenBank rodent | GENMOREDATA | path to GCG optional data files |
| STS:* | GenBank (sequence tagged sites) | GENBASE | path to GCG user's default databases |
| SY:* | GenBank synthetic | GENRUNDATA | path to GCG default data files |
| SYNTETIC:* | GenBank synthetic | TERM | user's terminal port (dev/tty) |
| TAGS:* | GenBank EST and GSS | PLOTPORT | user's terminal graphics port |

An example GCG list file of many elongation 1 and Tu factors follows. As with all GCG data files, two periods separate documentation from data.

```
my-special.pep begin:24 end:134
SwissProt:Eftu_Ecoli
Ef1a–Tu.msf{*}
/usr/accounts/test/another.rsf{ef1a_*}
@another.list
```
Using SeqLab

Now that you understand some of how the Wisconsin Package ‘thinks’ let’s take a look at the SeqLab GUI. But first display the file entitled “sample.list” in your account:

> more sample.list

!!SEQUENCE_LIST 1.0

..

SwissProt:EF1a_Giala Begin:1 End:396 Wgt:1.00 Type: P
!


@/usr/local/gcg/gcgcore/data/moredata/EF1a-primitive.SW.list

/usr/local/gcg/gcgcore/data/moredata/EF1a-primitive.rsf{*}

Since we’re not going to learn about all of the biocomputing databases until next week, I’ve made some of my data available in this sample list file for your use today. Had I not done this, you wouldn’t have any data to use today without learning more. I’ve placed three different GCG file specifications for EF-1[] sequences in this sample list file. The first points to Giardia lamblia EF-1[] from the local GCG SwissProt database; the second is a list file of EF-1[] sequences from several ‘primitive’ eukaryotes in the SwissProt database made by the GCG reference searching program LookUp; the third points to a Rich Sequence Format (rsf) file of a multiple sequence alignment of the same molecular system.

As discussed in the Introduction, specialized X-server graphics communications software is required to use GCG’s SeqLab. A few X tips should be mentioned at this point. X Windows are only active when the mouse cursor is in that window, and always close X Windows when you are through with them to conserve system memory. Furthermore, rather than holding mouse buttons down, to activate X items, just <click> on them. Also, X buttons are turned on when they are pushed in and shaded. Finally, don’t close X Windows with the X-server software’s close icon in the upper right- or left-hand window corner, rather, always, if available, use the window’s own “File” menu “Exit” choice, or “Close,” or “Cancel,” or “OK” button.

Now issue the command “seqlab &” (again without the quotes) in your terminal window to fire up the SeqLab interface. The ampersand, “&,” is not necessary but really helps out by launching SeqLab as a background process so that you retain control of your initial terminal window where you can issue OS commands. This should produce two new windows, the first an introduction with an “OK” box; check “OK.” You should now be in SeqLab’s “Main List” mode with the empty default list file, “working.list,” open in the main window. Go to the “File” menu and select “Open List . . .” and then use the “Open List File” window to select the “sample.list” file from your account. Press “OK” to see the next screen snapshot:
Besides having your instructor put data in a sample list file for you, other ways to get sequences into SeqLab include the “Add sequences from” “Sequence Files. . .” choice under the “File” menu. Only GCG format compatible sequences or list files are accessible through this route though. You can also directly load sequences from the online GCG databases with the “Databases. . .” choice under the “Add sequences” menu if you know their proper identifier name or accession code. When using the “Add Sequences from” “Sequence Files. . .” function the “Filter” box is very important! By default files are filtered such that only those that end with the extension ”.seq” are displayed. This often won’t do you any good as the sequences that you may want to add may have other extensions. Therefore, delete the ”.seq” extension in the “Filter” box (including the period) if necessary, but be sure to leave the ”*” wild card. Press the “Filter” button to display all of the files in your working directory (or ‘filter’ to any desired extension). Select the file that you want from the “Files” box, and then check the “Add” and then “Close” buttons at the bottom of the window to put the desired file into your current list, if you’re in List Mode, or directly into the Editor, if you’re in “Editor Mode.” Furthermore, in “Editor Mode” two additional choices are available. As mentioned earlier, you can “Import” sequences from GenBank or FastA format files or from ABI style binary trace files. And you can use the “File” menu “New Sequence” choice to create empty slots to hold brand new entries, either “DNA,” “RNA,” “Protein,” or “Text,” where you can either type in data or copy and paste it from a different window.

**SeqLab Preferences and Help**

Before going any further, go to the “Options” menu and select “Preferences . . .” A few of the options should be checked there to insure that SeqLab runs its most intuitive manner. The defaults are usually fine, but I want you to see what’s available to change. Remember, buttons are turned on when they’re pushed in and shaded.

First notice that there are three different “Preferences” settings that can be changed: “General,” “Output,” and “Fonts,” start with “General.” The “Working Dir . . .” setting will be the directory from which SeqLab was initially launched. This is where all SeqLab’s working files will be stored; it can be changed in your accounts if desired, however, it is appropriate to leave it as is for now. Be sure that the “Start SeqLab in:” choice has “Main List” selected and that “Close the window” is selected under the “After I push the “Run” button:”
choice. Next select the “Output” Preference. Be sure “Automatically display new output” is selected. Finally, take a look at the “Fonts” menu. If you are dealing with very large alignments, then picking a smaller Editor font point size may be desirable in order to see more of your alignment on the screen at once. <Click> “OK” to accept any changes.

SeqLab’s Help system is always available. Every program window and the Main window has a “Help” button. Press the “Help” button in the upper right-hand corner of the “Main List” window. This will produce a “Help” window where you can pick and choose topics as desired from the “Subtopics:” section. <Double-click> topics to visit; press the “Go Back” button to return. “Overview” moves you back to the root of the Help system. Take a few minutes to investigate the SeqLab online Help system at this point. Be sure to flag down your instructor if you have any problems.

Two other Help systems are also available on the Wisconsin Package. Issuing the commands “genhelp” or “genmanual” in a command line terminal window launches a text driven Help system and the URL http://www.csit.fsu.edu/gcg/ links to a Web version of the Help system. The Web version will request a username and password if connecting from an off-campus location, to comply with GCG’s license restrictions. Give the username “gcg” and the password “stevet,” if this is the case.

Exploring SeqLab

Select the top entry in your sample list file, “SwissProt:EFla_Giala” and switch “Mode:” to “Editor.” SeqLab now displays the EF-1α protein sequence from Giardia lamblia in its Main Editor Window. Your display should look similar to mine following below:

```
This individual sequence comes from the GCG SwissProt database (remember logical_term:ID). It’s named by its official SwissProt entry name (ID identifier) and appears in the Editor window with the amino acid residues color-coded. The nine color groups are based on a UPGMA clustering of the BLOSUM62 amino acid scoring matrix, and approximate physical property categories for the different amino acids. Turning off “Invert” causes the letters to assume the colors and the background to go white. Turning “Wrap” on causes the sequence to wrap vertically in the display. Use whichever combination of settings you prefer:
```
I prefer the default non-wrapped, inverted display. Quickly <double-click> on the entry’s name (or single <click> the “INFO” icon with the sequence entry name selected) to see the database reference documentation for it. (This is the same information that you can get with the GCG command “typedata -ref” at the command line.) You can also change sequences’ names and add any documentation that you want in this window. “Close” the “Sequence Information” window after reading it. Switch “Mode:” back to “Main List” to see the other sequences specified in your sample list file.

Next select the second entry in your list, “EF1a-primitive.SW.list.” This is a list file within a list file. Notice the at sign (@) in front of it in the working list. Remember the at sign is necessary in the Wisconsin Package for specifying a list file. <Double-clicking> the entry ‘opens’ the file to show you all of its members; <double-click> it again to ‘close’ the file. Insure that “EF1a-primitive.SW.list” is selected and then switch “Mode:” back to “Editor.” You should see an unaligned dataset similar to the figure below:
The file EF1a-primitive.SW.list specifies a number of EF-1 proteins from primitive organisms, all from the SwissProt database. Those are the entries seen in the Editor display now. The previous Giardia sequence is a member of this dataset. This is an example of the type of dataset that can be specified by the list file output of a Wisconsin Package program, here the LookUp program. Change “Mode:” back to “Main List” so that we can see the last GCG sequence specification in your working list.

Select the third entry there, the “EF1a-primitive.rsf{∗}” multiple sequence alignment. Remember the {∗} specifies all of the sequences within the Rich Sequence Format (rsf) file. Be sure “EF1a-primitive.rsf{∗}” is selected and then switch “Mode:” back to “Editor” again. Expand the window to an appropriate size by ‘grabbing’ the bottom-left corner of its ‘frame’ and ‘pulling’ it out as far as desired. Any portion of, or the entire alignment loaded, is now available for analysis by the GCG programs. The display should look similar to the following graphic:

![Sequence Alignment Graphic]

With this alignment loaded in the Editor explore the interface a bit. Nearly all GCG programs are accessible through the “Functions” menu. Select various entries’ names and then go to the “Functions” menu to perform different analyses on them, but don’t take the time to do that today. You can select sequences in their entirety by <clicking> on their names or you can select any position(s) within sequences by ‘capturing’ them with the mouse or by using the “Edit” menu “Select Range. . . ” function. You can select a range of sequence names by <shift><clicking> the top-most and bottom-most name desired, or <ctrl><click> sequence entry
names to select noncontiguous entries. (A bug in the Linux version of SeqLab prevents this from working correctly. The ‘work-around’ is to ‘cut-and-paste’ the sequences desired so that they end up side by side. GCG version 11 is supposed to have this fixed.) The “pos:” and “col:” indicators show you where the cursor is located on a sequence without including and with including gaps respectively.

Notice this alignment contains both protein and DNA sequence data. SeqLab doesn’t restrict you to using one or the other data types at a time. Each protein sequence is named with the Genus and species abbreviation for the organism it came from and the DNA sequences are named by their GenBank Accession codes. It also contains on-screen text annotation, the line entitled “functional_annotation.” Use the scroll bars to move around within the alignment. The scroll bar at the bottom allows you to move through the sequences linearly (unless “wrap” is set); the one at the side allows you to scroll through all of the entries vertically.

Place your cursor anywhere within the alignment data. Press the <space bar> to insert gaps and move the sequence to the right. Periods (.) appear in the sequence to represent alignment gaps. Press <delete> to remove the gaps. A very powerful manual alignment function can be thought of as the ‘abacus’ function. To do this select a region that you want to slide flanked by gaps and then press the <shift> key as you move the region with the right or left arrow key. You can slide residues greater distances by prefacing the command keystrokes with the number of spaces that you want them to slide. Notice that as you move a DNA sequence its corresponding protein translation comes along. This is called “grouping” and is indicated by the group numbers associated with the entry names. The “GROUP” function allows you to manipulate ‘families’ of sequences as a whole — any change in one will be propagated throughout them all. Here I have each DNA sequence grouped to its corresponding protein translation. To “GROUP” sequences, select those that you want to behave collectively and then <click> on the “GROUP” icon above your alignment. You can have as many groups as you want. You are not allowed to delete sequence characters unless you change their “Protections.” This prevents you from accidentally changing your sequence data. <Click> on the padlock icon to produce a “Protections” window. Notice that the default protection allows you to modify “Gap Characters” and “Reversals” only. Check “All other characters” to allow you to “CUT” regions out of an alignment and/or delete individual residues and then <click> “OK” to close the window.

Change the “Display:” box from “Residue Coloring” to “Feature Coloring.” The display now shows a color schematic of each entry’s feature information based on its database Feature Table, and on various GCG analyses that can produce RSF feature files. Quickly <double-click> on one of the various colored regions of the sequences (or use the “Features” choice under the “Windows” menu). This will produce a new window that describes the features located at the cursor. Select the feature to show more details and to select that feature in its entirety. All the features are fully editable through the “Edit” check box in this panel and new features can be added with several desired shapes and colors through the “Add” check box. “Close” the “Feature Editor” and “Sequence Features” windows, if you’ve opened them. Your display should look similar to the following window snapshot:
Next change "Display:" to "Graphic Features." "Graphic Features" represents features using the same colors as above, but in a 'cartoon' fashion. The "1:1" scroll bar near the upper right-hand corner allows you to 'zoom' in or out on the sequences. Move it to "2:1" and beyond and notice the difference in the display. Here's a region of my alignment at an "8:1" zoom factor and using "Graphic Features" "Display:"
This should be just about enough for today. Go to the “File” menu and use the “Exit” SeqLab choice to leave the GUI. You will be asked to “Save” the RSF file and the working list if you’ve made any changes in either. Check “OK” and the windows will go away.

Lab Report

Mendel has an e-mail system on it called Pine. Many of you may have used the Pine mailer in the past. It is not at all fancy and I certainly don’t recommend using it as your standard e-mail system. In fact Mendel doesn’t even allow incoming messages to be delivered. However, Mendel does allow outgoing Pine messages and we will be using it here as a way for you to turn in lab reports. Therefore, launch Pine by typing the command “pine” at the command line. If this is the first time that Pine has been started in your account you’ll get a message about a mail subdirectory being created and you’ll see a Welcome dialogue. Press “e” to exit the Welcome screen. Next press “c” from Pine’s “MAIN MENU” to compose a message to me. Type “stevet@bio.fsu.edu” into the “To . . :” space and “Lab Report #1” into the subject line. In the “Message Text” portion of the window tell me who you are, what your major or graduate program is, and who your faculty advisor is. Also tell me what computing experience you have had before enrolling in this course and what your expectations for the course are. Finally, press <control>“x” and then “y” to send me the message and then “q” and “y” to quit Pine. Next, log off of Mendel by typing “exit” at the command line in the ssh terminal window. Finally log off the Mac or Windows PC or Linux computer that you have been using.

Conclusion

Today’s tutorial should have provided you with the basics necessary to get about in the UNIX OS. You should now feel somewhat comfortable at the UNIX command line, at least enough so as to maintain your file and directory structure in your new Mendel account. Account maintenance is your responsibility; if you can’t find your files or figure out what is what, you only have yourself to blame. Furthermore, you should now have a basic understanding of how the GCG Wisconsin Package for sequence analysis works and of how to use their SeqLab GUI. We will be using SeqLab all semester long — get used to it. Next week you will learn how to access and use the various molecular biology databases available online, both locally within the GCG system, and remotely at Web servers throughout the world. You will also be asked to pick a ‘project’ molecule choice from a list of four, that the next nine lab tutorials will all use.

Acknowledgements

I offer special thanks to Charles Severance for providing much of the material on which the basic UNIX guide portion of this tutorial was based (http://www.msu.edu/~systems/ucgdocuments/misc/unixtut.html). I also wish to acknowledge Susan Jean Johns, my former colleague and supervisor at the Center for Visualization, Analysis and Design at Washington State University, now at the University of California San Francisco, for providing some of the introductory material for this tutorial — thank you Susan for teaching me so very much over my early years in bioinformatics.
References


