BioInformatics: A SeqLab Introduction

It's a brand new field in the last twenty years or so, called various names — computational molecular biology, biocomputing, sequence analysis, bioinformatics, and the latest, genomics and proteomics. But what does it mean? One way to think about it is "reverse biochemistry" — going from a DNA sequence to structure and function, rather than the other way round, no protein purification and characterization necessary. This is only possible because of modern computational speed and power and molecular database growth (GenBank doubles in size almost every year!).

So how does one do bioinformatics?

I. On the Internet through the World Wide Web — possible and easy and fun, but . . . can not readily handle large datasets or large multiple alignments, quickly becomes intractable.

In spite of this: See InterNet page following. So what are the alternatives . . .

II. Desktop software solutions — public domain programs are available, but . . . complicated to install, configure, and maintain. User must be pretty computer savvy. So,

commercial software packages are available, e.g. MacVector, DNAsis, DNAStar, etc.,

but . . . license hassles, expense per machine, and database access all complicate matters!

III. Therefore, server-based solutions (e.g. the Wisconsin Package) — UNIX server computers.

One license fee for an entire institution and very fast, convenient database access on local server disks. Connections from any networked terminal or workstation anywhere!

Operating system: command line operation hassles, communications software — telnet versus ssh and terminal emulation, X graphics, file transfer — sftp and scp, and editors — vi, emacs, pico (or desktop word processing followed by file transfer [save as "text only!"]).

See UNIX guide handout and Using X handout.

The Genetics Computer Group — the Wisconsin Package for Sequence Analysis. Begun in 1982 in the Genetics Dept. at the University of Wisconsin, Madison, then a private company for over 10 years, then acquired by the Oxford Molecular Group, and now owned by Pharmacopeia under the new name Accelrys, Inc., the suite contains almost 150 programs designed to work in a "toolbox" fashion. Several simple programs used in succession can lead to sophisticated results. Also 'internal compatibility,' i.e. once you learn to use one program, all programs can be run similarly, and, the output from many programs can be used as input for other programs. Used all over the world by more than 30,000 scientists at over 530 institutions in 35 countries, so learning it here will most likely be useful anywhere else you may end up.

See Specifying sequences and Logical terms handouts!

IV. SeqLab, a brief history — WPI, Steve Smith and GDE — and some illustrative examples: Glutathione Reductase, G-protein coupled TM7 receptors (a paralogous dataset), L1 major coat protein, Major Histocompatibility Class II, Vicilin seed storage proteins, and Elongation Factor 1α.

V. The tutorial: Elongation Factor Tu/1α.

BioInformatics and the InterNet: the World Wide Web

Some of my favorite World Wide Web sites for molecular biology information, sequence analysis, and general bioinformatics:

Site	URL (Uniform Resource Locator)	Content
National Center Biotech' Info'	http://www.ncbi.nlm.nih.gov/	databases/analysis/software
PIR/NBRF	http://pir.georgetown.edu/	protein sequence database
ProteinDataBank	http://www.rcsb.org/pdb/	3D mol' structure database
Molecules R Us	http://molbio.info.nih.gov/cgi-bin/pdb	3D protein/nuc' visualization
Johns Hopkins Med' Library	http://www.welch.jhu.edu/index.cfm	databases, journals, books
Harvard Bio' Laboratories	http://golgi.harvard.edu/	Mol'/Cell Bio' and links list
IUBIO Biology Archive	http://iubio.bio.indiana.edu/	database/software archive
Univ. of Montreal MegaSun	http://megasun.bch.umontreal.ca/	database/software archive
Japan's GenomeNet Server	http://www.genome.ad.jp/	databases/analysis/software
European Mol' Bio' Lab'	http://www.embl-heidelberg.de/	databases/analysis/software
European Bioinformatics Lab'	http://www.ebi.ac.uk/	databases/analysis/software
The Sanger Institute	http://www.sanger.ac.uk/	databases/analysis/software
Univ. of Geneva BioWeb	http://www.expasy.ch/	databases/analysis/software
The Genome DataBase	http://www.gdb.org/	Human Genome Project
Stanford Genomic Resource	http://genome-www.stanford.edu/	various genome projects
Inst. for Genomic Research	http://www.tigr.org/	microbial genome projects
HIV Sequence Database	http://hiv-web.lanl.gov/	HIV epidemeology seq' DB
The Baylor Search Launcher	http://searchlauncher.bcm.tmc.edu/	sequence search launcher
Pedro's BioMol Res' Tools	http://www.biophys.uni-duesseldorf.de/BioNet/Pedro/research_tools.html	extensive bookmark list
BioToolKit	http://www.biosupplynet.com/cfdocs/btk/btk.cfm	annotated molbio tool links
Felsenstein's PHYLIP site	http://evolution.genetics.washington.edu/phylip.html	phylogenetic inference
The Tree of Life	http://tolweb.org/tree/phylogeny.html	overview of all phylogeny
Ribosomal Database Project	http://rdp.cme.msu.edu/html/	databases/analysis/software
WIT Metabolism	http://wit.mcs.anl.gov/WIT2	metabolic reconstructions
BIOSCI/BIONET	http://net.bio.net	biologists' news groups
Access Excellence	http://www.accessexcellence.org/	biology education
CELLS alive!	http://www.cellsalive.com/	animated microphotography
Genetics Computer Group	http://www.accelrys.com/products/gcg_wisconsin_package	sequence analysis package

Basic Unix for Neophytes

An introduction and cheat sheet graciously stolen from the Internet and modified for bioinformatics use. I am indebted to the countless, but unnamed, contributors to this summary — I apologize for my flagrant copyright infringement. Hundreds of local users are forever grateful; thank you. Steve Thompson, July, 1995 (updated 1999 — 2004).

GENERAL INTERFACE ISSUES

UNIX is an Operating System developed in the USA, originally by BELL, then licensed to AT&T and now used in various implementations on many different brands of servers, minicomputers, and scientific workstations the world over. Appropriate communications software connects local personal microcomputers to UNIX hosts. The software used is usually some variation of TELNET or ssh, public domain programs used to access remote hosts via the Internet. Follow the appropriate instructions for your system to access the host UNIX computer at your site.

UNIX is a line-oriented system similar to the old operating systems MS-DOS and VMS. Each command to the operating system is terminated by the 'return' or 'enter' key. UNIX uses the ASCII character set and unlike some operating systems, it supports both upper and lower case. The disadvantage of using both upper and lower case is that commands and file names must be typed in the correct case. Most of the UNIX commands and file names are in lower case. This is different from VMS where everything is mapped into upper case. 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") and the letter c. The short form for this is generally written as CTRL-C or ^C. Using control keys instead of special function keys for special commands is sometimes difficult for users to remember; however, the advantage is that nearly every terminal and terminal program supports the control key, allowing UNIX to be used from a wide variety of different terminals.

UNIX FILE SYSTEM & A SHORT TUTORIAL

The UNIX user interface is often characterized as very unfriendly compared to other operating systems. However, in the area of file systems, UNIX is quite straightforward. UNIX is the precursor of many of the tree structured file systems of today including MS-DOS/Windows, VMS, and Macintosh OS. These file systems all consist of a tree of directories and sub-directories. In any tree structured file system the concept of where you are in the tree is very important. There are two types of file references. You can refer to a file relative to the current directory or by its complete 'path' name. When the complete path of a file is given the current position in the directory tree is ignored.

YOUR LOGIN ENVIRONMENT

When a user logs in to the system, they are placed into their 'home directory,' a portion of the disk space reserved just for them and designated from anywhere in the system by the character string "`$HOME`". UNIX checks the username and password you typed, and if correct, will run your shell program and return the system prompt. On UNIX systems the prompt may appear as many different forms depending on how the system administrator has set up the user environment. Here I will use the 'greater than' sign (>) to represent the system prompt. It should not be typed as part of the command. The shell program is your interface to the system. It interprets and executes your commands. Common UNIX shells include bash, the C-shell, and a popular C shell alternative, tcsh, that, like bash, maintains VMS-like command history arrow key recall and command editing features. When an account is created, the home directory is created and associated with the account. To find the complete path to your home directory type "`pwd`" after you log in. Note that the password is not displayed on the screen as you enter it while logging in:

```
login: example Password:
> pwd
/disk4/usr/local/people/example
```

GENERAL COMMAND SYNTAX

The general command syntax 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 out on the users 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 is as follows:

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

To cause the command to read from a file rather than the terminal, the < sign is used on the command line; use the > sign to cause 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. This feature is called "piping" the output of one program into the input of another:

```
cmd -options parameters | cmd2 –options paramters
```

IMPORTANT UNIX COMMANDS AND KEYSTROKES

Important conventions used in UNIX:

< . >	Current working directory.
< .. >	Parent directory of current working directory.
< ~ >	User's home directory (C and T shell only, also `$HOME`).
< & >	Execute the specified command in another process.

Commands to get information about the operating system:

`man ls`	Gets you a manual page on the ls command.
`man -k batch`	Gets you the title lines for every command with the word batch in the title.

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

Command to change your password:

`passwd`	Change your login password

Commands allow you to move around your directory structure. These commands are generally quite similar between different operating systems. In UNIX these directory commands are as follows:

`pwd`	Print working directory. Shows you where you are at in the file system. Very useful when you get confused. (Also see "`whoami`" if you get really confused!)
`ls`	Shows (lists) the contents of the directory, i.e. your files' names within the directory.
`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 dir`	Makes a new directory in your current directory.
`rmdir dir`	Removes a sub-directory 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`	Change directory back into your home directory from anywhere.
`cd` or `chdir dir`	Move down into a directory from your current directory.

The GCG commands "`up`", "`down`", "`over`", and "`home`" can also be used to move about your directory structure in lieu of the UNIX command "`cd`".

To list the files in your home directory, use the "`ls`" command. There are many options to the ls command that you can look at 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. In general you do not want to mess with these files in your account until you are very comfortable with the operating system. Examples of these types of files in many UNIX systems include the files `.login`, `.cshrc`, and `.pinerc`. You may also create files beginning with a period if you do not want them to show up in a normal ls command. The following are several examples of the ls command:

```
> ls
fileone     two      proj1
> ls -l
-rw-------  1 example     10028 Sep 26 11:00 fileone
-rw-rw----  1 example       281 Oct  5 14:21 two
drwx------  2 example       638 Aug 21 17:27 proj1
> ls -a
.login      .cshrc    fileone     two      proj1
```

In the output from "`ls –l`" additional information regarding the file permissions, owner of the file, size, modification date and file name is shown. Nearly all operating systems have some way to customize your login environment with editable configuration files. On many UNIX systems there is a file that is executed for every login called "`.login`" and another one that sets up the 'shell' environment called "`.cshrc`". These files are shown in the output from "`ls –a`" above. Both of these files are executed when the user logs in, much like `AUTOEXEC.BAT` and `CONFIG.SYS` are in MS-DOS, and `LOGIN.COM` is in VMS, when you log in on those systems. The user can place commands in these files that customize their individual environment once they are comfortable with the operating system.

Sub-directories are generally used to group files associated with one particular project or files of a particular type. For example, a person might store all of their memorandums in a directory called "`memo`". The "`mkdir`" command is used to create directories and the "`chdir`" (or `cd`) command is used to move into directories. The special placeholder file "`.`". allows you to move back up the directory tree. Note that UNIX uses forward slashes "/" to differentiate between subdirectories, not backward slashes "\" like MS-DOS or brackets "[ ]" and periods "." like VMS. The `pwd` command can be used at any point to keep track of your current sub-directory.

```
> ls
fileone     two      proj1
> mkdir memo
> ls
fileone     two      proj1    memo
> ls -l
-rw-------  1 example     10028 Sep 26 11:00 fileone
-rw-rw----  1 example       281 Oct  5 14:21 two
drwx------  2 example       638 Aug 21 17:27 proj1
drwx------  2 example       638 Aug 22 14:47 memo
> pwd
/disk4/usr/local/people/example
```

```
> cd proj1
> pwd
/disk4/usr/local/people/example/proj1
```

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

`cat file`	Shows the contents of a file on your screen and concatenates files, e.g: "`cat file1 file2 > file3`".
`more file`	Shows the contents of your file at the terminal pausing to allow you to press space to continue. Type a "`?`" when the scrolling stops for viewing options (`less` also sometimes available; it is more powerful than `more`).
`pico file`	A text editor provided in the pine mailer; appropriate for general text editing but not on all systems (`vi` also available as default UNIX text editor but it is quite difficult to master; `emacs` may also be available).
`head file`	Shows the first few lines of a file.
`tail file`	Show the last few lines of a file.
`grep ptrn file`	Show the lines in the file which contain the specified pattern.
`wc file`	Counts the number of characters, words, and lines in a file.
`cp file1 file2`	Copies file1 to file2. Any previous contents of file2 are lost.
`mv file1 file2`	Renames (moves) file1 to file2. Any previous contents of file2 are lost.
`mv file dir`	Moves the specified file into the specified directory keeping the original file name.
`rm file`	Deletes (removes) a file. It is unrecoverable!
`chmod perm`	Changes the permissions of a file. See "`man chmod`" for details.
`lpr file`	Prints the specified file on the default system printer. May need to specify a particular print queue with the "`-P`" option.

Some examples using file commands follow below:

```
> ls
main.c     sub.c      a.out
> cat main.c
```

- contents of main.c are displayed to the screen-

```
> cd ..
> pwd
/disk4/usr/local/people/example
> cat /disk4/usr/local/people/example/proj1/main.c

 - contents of main.c -

> cat proj1/main.c

- contents of main.c -

> cat main.c
main.c not found.
```

At the end of this example, the "`cat`" command is used to show the use of relative and absolute file names. The first cat command displays the contents of `main.c` in the sub-directory `proj1` of user example's home directory. This reference is a relative reference because it does not start with the slash "`/`" character. The use of the cd command with two dots to go back up to the parent of the current directory is also shown. After the "`cd ..`" command pwd shows that we are 'back' in the home directory. The next cat command shows the use of an absolute file name that starts with a slash. This cat succeeds because the file name is completely specified as the complete path name and the file name. The command "`cat proj1/main.c`" also succeeds because relative to the home directory, the sub-directory and file name are correctly specified. The command "`cat main.c`" in the home directory does not find the file stored in the sub-directory. Two alternative examples that allow scrolling through directory listings using command redirection and piping are shown below with the ls command and the more command:

```
> ls -la | more
> ls -la /etc > tmpout
> more tmpout
```

The "`grep`" command is very useful; it allows searching through files for particular patterns. The first parameter to grep is a search pattern. If no files are specified, grep will scan its "standard input" for lines containing the pattern and write them to "standard output," i.e. your terminal screen. You may also specify a list of file names on the grep command. For example if you had a bunch of different C programs in several sub-directories and wanted to find the one that called the "`fopen`" subroutine, you could use the following command:

```
> grep fopen /.c
```

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.

`who`	Shows who is logged on to the system.
`w`	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. (Not always installed.)
`finger`	Gets information about a user or machine.
`ps`	Shows your current processes and their status (running, sleeping, idle, terminated, etc.); (use man pages as options widely vary, see especially `-aelfuU`).
`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.

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 program can be restarted by typing "`fg`" (foreground). If you type "`bg`" (background) the job will also be started again, but in background mode. Notice this is NOT the same as VMS's `<Ctrl z>` which is more like UNIX's `<Ctrl d>`.
`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.

The following commands provide simple access to some of the networking capabilities of UNIX (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.).
scp	Secure copy file, syntax: "`scp file user@host:path`" or "`scp user@host:path file`".
`telnet host`	Connects to another Internet host using an insecure connection (discouraged!).

`ssh user@host`	Connect to a host computer using a secure, encrypted protocol.
`pine`	E-mail access/use (not always installed; `mail` is default UNIX mailer).
`talk user`	Allows you to have an on-line screen dialog with another user on the system. Talk may also be able to talk between systems. The form "`talk user@host`" is used in these cases.

THE USE OF WILDCARDS AS FILE PARAMETERS

Certain printing (non-control) characters have special meaning to the UNIX shell environment (see "`man csh`" or "`tcsh`" for more information). These characters are called shell metacharacters. You rarely type shell metacharacters on the command line because they are punctuation characters. However, if you need to use them for some reason, you must precede them with a "\" (backslash) character or enclose them in "'" (single quotes). The "`*`" (asterisk), "`?`" (question mark), and "~" (tilde) characters are used for the shell file name "globbing" facility. When the shell encounters a command-line word with a leading "~", "`*`", or "`?`" anywhere on the command line, it attempts to expand that word to a list of matching file names as follows: 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.

Two globbing characters cause 'wild card' expansion. Generally when a UNIX command expects a file name, such as the command, "`more filename`", it is possible to specify a group of files using one or more wild card expressions:

`*`	Matches any string of characters zero or longer
`?`	Matches any single character

For example, the pattern "`dog*`" will find matches for, among others, files named "`dog`", "`dogg`", and "`doggy`". The pattern "`dog?`" matches, among others, "`dogg`," but not "`dog`" or "`doggy`". More examples of commands using wild card characters follow:

```
more dat*
more d?t
more ???crs
more /.c
more */raisememo
```

The second to last example will find all files ending in "`.c`" in all sub-directories below the current directory and display those files one page at a time to your terminal screen, pausing between each file. The last example will display all files named "`raisememo`" in all sub-directories below your current directory.

Wild cards are very flexible in UNIX, which makes them very powerful but you must also be extremely careful when using them in potentially destructive commands such as "`rm`" (remove file).

Using X between different UNIX computers

These are the bare-minimum instructions necessary for connecting to a UNIX host computer from another UNIX computer using X. Not all commands are necessary in all cases, often they are set by your account environment; however, I'll supply a complete set. In most cases fully qualified Internet names can be used in these procedures, however, depending on local name servers, you may need to specify IP numbers. A fictitious example host machine, zen.art.motorcycle.com, has the following name and number:

```
zen.art.motorcycle.com           999.999.99.99
```

You will need to know your own machine's name and/or number as well as the host's.

Log on to your UNIX workstation account in the customary manner. Depending on the workstation, you may want to specify an xterm terminal window. Sometimes this is launched through your desktop GUI with a mouse button, otherwise:

```
Optional: > /usr/bin/X11/xterm &
          On Solaris: > /usr/openwin/bin/xterm &
```

Following UNIX X commands with an ampersand, "**&**", is helpful so that they are run in the background in order to maintain control of the initial terminal window. Some helpful options supported in most versions of xterm are **`-ls`** so that your login script is read, **`-sb -sl 500`** to give you a 500 line scroll back capability, **`-tn vt220`** to take advantage of vt220 terminal features, and **`-fg Bisque -bg MidnightBlue`** to give you nice light colored characters on a dark blue background.

Then, at your workstation's UNIX prompt, if required, authorize X access to the host with the xhost command:

```
> xhost +zen.art.motorcycle.com
```

Next connect to the host with the ssh, telnet, or rlogin command, whichever is the preferred route at your site; e.g:

```
> ssh -X thompson@zen.art.motorcycle.com
```

(ssh -X sets the X environment for you, so the xhost command above and the setenv command below should not be necessary.)

This should produce a login window. Log in as usual, then, if necessary, setup the X environment on the host (for the c shell and its derivatives), where your_IP_node_name represents the Internet name or number of the workstation that you are sitting at:

```
Host> setenv DISPLAY your_IP_node_name:0.0
```

It is may be best to run commands from an X terminal window rather than a default console window, as is often created by a remote connection. Therefore, after setting up your X environment, if this is the case, an option is to launch xterm by minimally issuing the **`xterm`** command to the host (as discussed above, many options are available).

X server emulations — on PCs and Macs: Pre-OS X Macintoshes can use either MacX or eXodus. Apple's X11 distribution, XFree86's XDarwin, or Tenon's Xtools are good for getting true X under Apple OS X. In the PC/Windows world I have used Xwin32, eXcursion, and eXceed. In most X server emulations the setenv command is not required, as this information is usually sent by the X server software on the personal computer upon connection. The xterm command is usually sufficient for creating an xterm window. Refer to your own software's documentation for more details as it is impossible to describe all situations.

SPECIFIC DIFFERENCES BETWEEN THE VMS AND UNIX GCG PACKAGE

This section is intended for people who were familiar with the old VMS version of the GCG Wisconsin Package. It explains the differences between running the Package in the VMS and UNIX environments. In both the VMS and UNIX versions, you can run a GCG program by entering information on the command line — after the VMS $ prompt and the UNIX > prompt. The command line can contain the name of a command, command qualifiers, qualified parameters (qualifiers with values), and unqualified parameters (usually file names). The general syntax, or structure, of a VMS command is:

```
COMmand /QUALifier /QUALifier=Parameter Parameter
```

As with all VMS commands, spaces on the command line are ignored (except the required space in front of an unqualified parameter), characters can be typed in upper case or lower case (case insensitivity), qualifiers are indicated by a '/' (slash), qualified parameters are indicated by an '=' (equals sign), and the upper case typeface indicates the fewest number of characters you can enter. An example of a GCG command using VMS syntax is:

```
MAPPLot /CIRcular /OUTfile=pBR322.MapPlot Gb:Synpbr322
```

A UNIX command varies in several ways from a VMS command. The general syntax of a UNIX command is:

```
command -QUALifier -QUALifier=Parameter Parameter
```

The main difference between the VMS and UNIX versions of GCG is that command names must be typed either in full or with site specific aliases — generalized abbreviation does not work — and all in lower case. In addition, qualifiers are indicated with a required space and a " –" (hyphen), instead of a "/". In both versions, qualified parameters are indicated by an "=" and spaces are not accepted between a qualifier and its parameter(s). The case of unqualified parameters can vary, as well as that of the qualifier itself, but if the unqualified parameter is a file name, you must enter the file name in the exact case shown in your directory listing. As in VMS, qualifiers can be abbreviated down to the minimum number of characters indicated by upper case. An example of the mapplot GCG command using UNIX syntax is:

```
mapplot -CIRcular -OUTfile=pbr322.mapplot Gb:Synpbr322
```

A quick reference for previous users of VMS who are trying to learn UNIX follows. Look for a task or VMS command to choose the appropriate UNIX command.

To ...	VMS	UNIX
end a program	`<Ctrl y>`	`<Ctrl c>`
suspend a program	(none available)	`<Ctrl z>`
exit current command level	`<Ctrl z>`	`<Ctrl d>`
display list of files	`DIRECTORY`	`ls`
	`DIRECTORY/FULL`	`ls -al`
display contents of file	`TYPE`	`cat`
display file with pauses	`TYPE/PAGE`	`more, less`
display first few lines of file		`head`
display last few lines of file		`tail`
edit a file	`EDT, EVE`	`pico, vi`
copy file	`COPY`	`cp`
compare files	`DIFF`	`diff`
		`cmp`
rename file	`RENAME`	`mv`
delete file or directory	`DELETE`	`rm`
		`rmdir`
change file protection	`SET FILE/PROT`	`chmod`
change file ownership	`SET FILE/OWNER`	`chown`
create directory	`CREATE/DIR`	`mkdir`
change working directory	`SET DEFAULT`	`cd`
display working directory	`SHOW DEFAULT`	`pwd`
get help	`HELP`	`man`
		`apropos`
display date and time	`SHOW TIME`	`date`
display free disk space	`SHOW DEVICE`	`df`
stop process	`STOP`	`kill`
link program modules	`LINK`	`ld`
print file	`PRINT`	`lpr`
display print queue	`SHOW QUEUE`	`lpq`
display print entries	`SHOW ENTRY`	`lpq`
change password	`SET PASSWORD`	`passwd`
display logged-in users	`SHOW USERS`	`who`
and information about them		`finger`
		`w`
display processes	`SHOW PROCESS`	`ps`
change terminal settings	`SET TERMINAL`	`stty`
talk to another user	`PHONE`	`talk`
disable messages	`SET NOBROADCAST`	`mesg n`

This document is intended to give you some perspective on the UNIX operating system and guide you toward learning more about it. UNIX is not the easiest computer operating system to learn. Have patience, ask questions, and don't get down on yourself just because it doesn't seem as easy as some other computer operating systems. The power and flexibility of UNIX is worth the extra effort. UNIX has become the defacto standard operating system in more and more computing environments today, particularly scientific computing, so the effort will not be wasted.

To answer the always perplexing GCG question — “What sequence(s)? . . . .” Specifying sequences, GCG style; 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` (`my.pep` or `~user/subdir/my.seg`). Use the program ‘reformat’ to convert ‘raw’ text format files to GCG format. (Also — SeqLab can directly import GenBank or FastA format text files or ABI/SCF style binary trace files.)

```
This is a small example of ‘raw’ 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! Two periods always separate that documentation from the actual data.

..

ACTGACGTCACATACTGGGACTGAGATTTACCGAGTTATACAAGTATACAGATTTAATAGCATGCGATCCCATGGGA
```

Next the clean GCG format single sequence file after ‘reformat’:

```
This is a small example of ‘raw’ 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! Two periods always separate that documentation from the actual data.
 example.seq  Length: 77  July 21, 1999 09:30  Type: N  Check: 4099  ..

       1  ACTGACGTCA CATACTGGGA CTGAGATTTA CCGAGTTATA CAAGTATACA
      51  GATTTAATAG CATGCGATCC CATGGGA
```

2) The sequence is in a local GCG database 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 can be used for nucleotide databases, such as bacterial. Most GCG logical database names are listed on the next page. A colon, “`:`”, always sets the logical name apart from either an accession code or a proper identifier name or a wildcard expression and they are case insensitive. Several examples follow: `GenBank:EctufBT`, `gb:x57091`, `SwissProt:EFTu_Ecoli`, `sw:p02990`, `PIR:EfEcTA`, and `p:a91475` all refer to the elongation factor Tu in *E. coli*. 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; e.g. `SW:EFTu_*` specifies all of the elongation factor Tu sequences in SwissProt; `SW:*_Ecoli` refers to all *E. coli* sequences. Because all the sequences are available in local GCG databases, it is seldom necessary to put individual database sequences in your account.

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 data, whereas RSF files contain sequence names and data, and all of the database annotation for each entry. 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 sequence specification. For example, to specify all of the sequences in an alignment of elongation 1α and Tu factors, one may use a naming system such as 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 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 other sequence specifications and can even contain other list files within it. The convention to use a GCG list file in a program is to precede it with an at sign, “`@`.” Furthermore, one 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 can be seen in the example below:

```
An example GCG list file of many elongation 1a 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
```

Logical terms for the Wisconsin Package at FSU

Sequence databases, nucleic acids:

GENBANKPLUS	all of GenBank plus EST and GSS subdivisions
GBP	all of GenBank plus EST and GSS subdivisions
GENBANK	all of GenBank except EST and GSS subdivisions
GB	all of GenBank except EST and GSS subdivisions
BA	GenBank bacterial subdivision
BACTERIAL	GenBank bacterial subdivision
EST	GenBank EST (Expressed Sequence Tags) subdivision
GSS	GenBank GSS (Genome Survey Sequences) subdivision
HTC	GenBank High Throughput cDNA
HTG	GenBank High Throughput Genomic
IN	GenBank invertebrate subdivision
INVERTEBRATE	GenBank invertebrate subdivision
OM	GenBank other mammalian subdivision
OTHERMAMMAL	GenBank other mammalian subdivision
OTHER_MAMMALIAN	GenBank other mammalian subdivision
OV	GenBank other vertebrate subdivision
OTHERVERTEBRATE	GenBank other vertebrate subdivision
OTHER_VERTEBRATE	GenBank other vertebrate subdivision
PAT	GenBank patent subdivision
PATENT	GenBank patent subdivision
PH	GenBank phage subdivision
PHAGE	GenBank phage subdivision
PL	GenBank plant subdivision
PLANT	GenBank plant subdivision
PR	GenBank primate subdivision
PRIMATE	GenBank primate subdivision
RO	GenBank rodent subdivision
RODENT	GenBank rodent subdivision
STS	GenBank (sequence tagged sites) subdivision
SY	GenBank synthetic subdivision
SYNTHETIC	GenBank synthetic subdivision
TAGS	GenBank EST and GSS subdivisions
UN	GenBank unannotated subdivision
UNANNOTATED	GenBank unannotated subdivision
VI	GenBank viral subdivision
VIRAL	GenBank viral subdivision

Sequence databases, amino acids:

GENPEPT	GenBank CDS translations
GP	GenBank CDS translations
SWISSPROTPLUS	all of Swiss-Prot and all of SPTrEMBL
SWP	all of Swiss-Prot and all of SPTrEMBL
SWISSPROT	all of Swiss-Prot (fully annotated)
SW	all of Swiss-Prot (fully annotated)
SPTREMBL	Swiss-Prot preliminary EMBL translation
SPT	Swiss-Prot preliminary EMBL translation
P	all of PIR Protein
PIR	all of PIR Protein
PROTEIN	PIR fully annotated subdivision
PIR1	PIR fully annotated subdivision
PIR2	PIR preliminary subdivision
PIR3	PIR unverified subdivision
PIR4	PIR unencoded subdivision
NRL_3D	PDB 3D protein sequences
NRL	PDB 3D protein sequences

General data files:

GENRUNDATA	path to GCG default data files
GENMOREDATA	path to GCG optional data files