

Biomolecular Sequence Alignment and Analysis:

Part I — BioInformatics: A SeqLab Introduction.

A GCG[®] Wisconsin Package[™] SeqLab[®] Tutorial for Iowa State University.

Bioinformatics and Computational Biology
2014 Molecular Biology Building
Iowa State University, Ames, IA 50011
515-294-5122; 1-888-569-8509
Fax 515-294-6790
www.bcb.iastate.edu

November 8 – 10, 2001

author: **Steven M. Thompson**

Florida State University
School of Computational Science & Information Technology
Tallahassee, Florida 32306-4120
telephone: 850-644-1010
fax: 850-644-0098

corresponding address:

Steve Thompson
BioInfo 4U
2538 Winnwood Circle
Valdosta, Georgia, 31601-7953
telephone: 229-249-9751
stevet@bio.fsu.edu

**^vGCG is the Genetics Computer Group, part of Accelrys Inc., a subsidiary of Pharmacia Inc.,
producer of the Wisconsin Package™ for sequence analysis.**

© 2001 BioInfo 4U

Bioinformatics: A SeqLab Introduction.

It's a brand new field in the last twenty years or so, called various, often misunderstood names, that are largely subsets of one another — computational molecular biology, biocomputing, sequence analysis, and now “bioinformatics,” “genomics” and “proteomics.” But what does it mean? One way to think about computational biology is the reverse biochemistry analogy — biochemists no longer have to begin a research project by isolating and purifying massive amounts of a protein from its native organism in order to characterize a particular gene product. Rather, now scientists can amplify a section of some genome based on its similarity to other genomes, sequence that piece of DNA, and, using sequence analysis tools, infer all sorts of functional, evolutionary, and, perhaps, structural insight into a gene within it, and then, perhaps, go on to cloning that gene, expressing the gene product, and finally purifying the protein. The process has come full circle. The computer has become an important tool to be used at the beginning and throughout a research project in assisting experimental design, not just a number-cruncher used at the end of the process. This is only possible because of modern computational speed and power and molecular database growth. Biocomputing's tremendous growth is reflected in and largely a result of the increase in the level of computational processing power available along with a concurrent exponential growth of the molecular sequence databases. GenBank doubles in size about every year! The current release, version 125, August 2001, has 13,543,364,296 bases from 12,813,516 reported sequences.

Definitions — Much confusion abounds in the area, even concerning the names of the disciplines themselves. The terms computational biology, biocomputing, bioinformatics, sequence analysis, molecular modelling, genomics, and proteomics are often bantered about with little regard to what they really mean. All are interdisciplinary by nature, combining elements of computer and information science, mathematics and statistics, and chemistry and biology. Each has elements of one another. Biocomputing and computational biology are the most encompassing terms and can be considered synonyms. They both describe using computers and computational techniques to analyze a biological system, whether that is a biomolecular primary sequence or tertiary structure, or a metabolic pathway, or even a complex system such as the interactions of populations within an ecological niche.

Bioinformatics necessarily intersects with this concept in that it describes using computational techniques to access, analyze, and interpret the biological information in databases. However, these databases can be the traditionally considered nucleic and amino acid sequence databases as well as three-dimensional molecular structure databases, but can even include such disparate data collections as medical records or population statistics. Therefore, bioinformatics is a type of biocomputing but also includes topics such as medical informatics that is not usually considered a part of computational biology.

Within bioinformatics the subdiscipline of sequence analysis has a clearly defined scope. It is the study of biological molecular sequence data for the purpose of inferring the function, interactions, evolution, and perhaps structure of biological molecules. Molecular modeling can also be considered a type of bioinformatics, though it often isn't. It is necessarily a subdiscipline of computational structural biology, but uses the methodology and techniques of that discipline as well as sequence analysis' similarity searching and alignment algorithms. That is why it is often referred to as “homology modeling.”

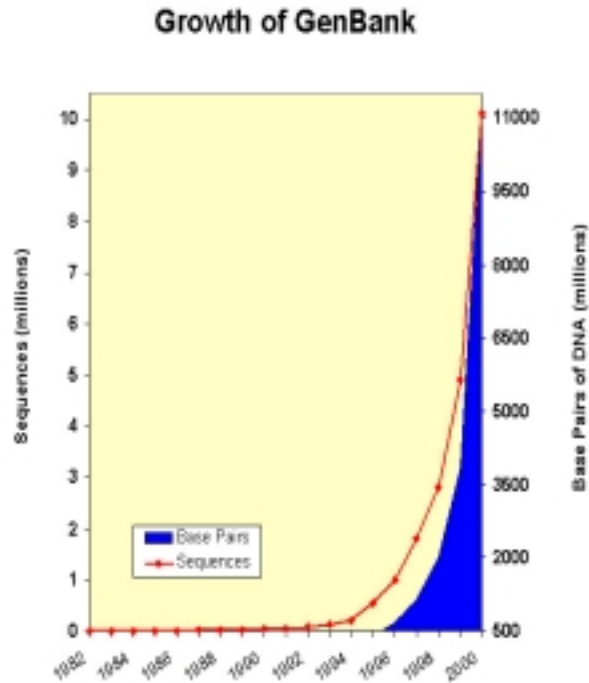
Genomics is the subdiscipline of bioinformatics that is concerned not with individual molecular sequences, but rather with sequences on a genomic scale. That is, genomics analyzes the context of genes or complete genomes (the total DNA content of an organism) within and across genomes. Proteomics can be considered the subdivision of genomics concerned with analyzing the complete protein complement, i.e. the proteome, of organisms, both within and between different organisms. Structural genomics is the acquisition and analysis of the complete set of three-dimensional structure coordinate data for an organism's entire proteome (or a representative set thereof). Through analyses such as these, it may eventually be possible to predict a completely unknown protein's structure and function just based on its

deduced molecular sequence. Obviously this could be an incredible boost to the drug-design process and could go a long way toward curing many disease processes. We have come a long way in structural prediction but are still a long way from this goal. The comparative method is crucial to all these methods but, perhaps most obvious and key to genomics and proteomics.

I. Databases: Content and Organization.

The International Human Genome Sequencing Consortium announced the completion of a "Working Draft" of the human genome in June 2000; independently that same month, the private company Celera Genomics announced that it had completed the first assembly of the human genome. As of August 2001 (GenBank version 125) 64 complete, finished genomes are publicly available for analysis, not counting all the virus and viroid genomes, 669 and twenty respectively. Eleven Archae[bacterial] and forty eight [Eu]Bacteria complete, annotated genomes are present. One nucleomorph genome, from the cryptomonad *Guillardia theta*, and four complete eukaryotic genomes, *Arabidopsis thaliana*, *Saccharomyces cerevisiae*, *Caenorhabditis elegans*, and *Drosophila melanogaster*, are finished and publicly available. *Homo sapiens*, Mouse (*Mus musculus*), and the Zebrafish (*Danio rerio*), as well as Corn (*Zea mays*), Barley (*Hordeum vulgare*), Wheat (*Triticum aestivum*), and Rice (*Oryza sativa*) have been nearly completed but are not yet entirely finished. Over half of the genes in many of these organisms have predicted functions based on previously studied bacterial genes. The Human Genome Project and numerous smaller genome projects have kept the data coming at alarming rates. GenBank growth statistics (<http://www.ncbi.nlm.nih.gov/Genbank/genbankstats.html>) are staggering:

<u>Year</u>	<u>BasePairs</u>	<u>Sequences</u>
1982	680338	606
1983	2274029	2427
1984	3368765	4175
1985	5204420	5700
1986	9615371	9978
1987	15514776	14584
1988	23800000	20579
1989	34762585	28791
1990	49179285	39533
1991	71947426	55627
1992	101008486	78608
1993	157152442	143492
1994	217102462	215273
1995	384939485	555694
1996	651972984	1021211
1997	1160300687	1765847
1998	2008761784	2837897
1999	3841163011	4864570
2000	11101066288	10106023



What are primary sequences?

Primary refers to one dimensional — all of the “symbol” information written in sequential order necessary to specify a particular biological molecular entity, be it polypeptide or nucleotide. The symbols are the one letter alphabetic codes for all of the biological nitrogenous bases and amino acid residues and their ambiguity codes. Biological carbohydrates, lipids and structural information are not included within this sequence; however, much of this type of information is available in the reference documentation annotation associated with primary sequences in the databases.

What are sequence databases?

These databases are an organized way to store the tremendous amount of sequence information that accumulates from laboratories worldwide. This data is piling up at exponential rates, as seen above. Each database has its own specific formats and access to this information is most easily handled through various software packages and interfaces, either on the World Wide Web or otherwise. Three major database organizations worldwide are responsible for maintaining most of this data.

In the United States the National Center for Biotechnology Information (NCBI), a division of the National Library of Medicine (NLM), at the National Institute of Health (NIH), supports and distributes the GenBank nucleic acid sequence database and GenPept CDS (Coding Sequence) translations database. The National Biomedical Research Foundation (NBRF), an affiliate of Georgetown University Medical Center, maintains the Protein Identification Resource (PIR) database of polypeptide sequences.

The European Molecular Biology Laboratory maintains the EMBL nucleic acid sequence database and the excellently annotated Swiss-Prot protein sequence database (also supported by the Swiss Institute of Bioinformatics: SIB), as well as the minimally annotated TrEMBL (Translations from EMBL — those EMBL translations not yet in Swiss-Prot) protein sequence databases in Cambridge, UK; Heidelberg, Germany; and Geneva, Switzerland. Additional, lesser-known, sequence databases include sites with the military, with private industry and in Japan (the DNA Data Bank of Japan: DDBJ). In most cases data is openly exchanged between the databases so that many sites ‘mirror’ one another.

One special sequence database deserves separate mention. NRL_3D is a database began by the U.S. Naval Research Labs, but now maintained by NBRF, of the peptide sequences from entries in the three-dimensional structural Protein Data Bank (PDB). As such it serves as a ‘bridge’ between primary and tertiary information.

What information do they contain, how is it organized, and how is it accessed?

Most sequence databases are examples of complex ASCII/Binary databases. They often contain several very long text files containing different types of information all related to particular sequences, such as all of the sequences themselves, versus all of the title lines, or all of the reference sections. Binary files often help ‘hold together’ all of these other files by providing indexing functions. Software is usually required to successfully interact with these databases although systems level commands can be used if one understands the data’s structure. Nucleic acid databases (and TrEMBL) are split into subdivisions based on taxonomy (historical). Reference headers include much extremely valuable information — author and journal citations, organism and organ of origin, and the FEATURES table. The features table annotation lists all sorts of important regulatory, transcriptional and translational (CDS coding sequence), catalytic, and structural sites, depending on the database. Actual sequence data follows the annotation. Becoming familiar with the general format of sequence files for the type of software you use can save a lot of grief later.

Software specific routines are the most convenient method in which to access these databases. However, most methods require that you know the proper sequence name and/or database identifier and those are usually discovered with some sort of text searching program, again either on the World Wide Web or not. This brings up two points, locus names versus accession numbers: The LOCUS, ID, and ENTRY names category in the databases are different than the Accession number. Each sequence is given a unique accession number upon submission. This number allows tracking of the data when entries are merged or split; it will always be associated with its particular data. Entry names may change; accession numbers are forever; they just pile up.

What changes have occurred in the databases — history and development?

The first well recognized sequence database was Dayhoff's Atlas of Protein Sequence and Structure begun in the late sixties. GenBank began in 1982, EMBL in 1980. They have all been attempts at establishing an organized, reliable, comprehensive and openly available library of genetic sequence. Databases have long since outgrown a hardbound atlas. They have become huge and have evolved through many changes. Changes in format over the years are a major source of grief for software designers and program users. Each program needs to be able to recognize particular aspects of the sequence files; whenever they change, it's liable to throw a wrench in the works. NCBI's ASN.1 format and its Entrez interface, available on the Web or as a dedicated client-server program, attempt to circumvent these frustrations somewhat. EMBL's SRS (Sequence Retrieval System) found on the World Wide Web at all EMBL OutStations and the Wisconsin Package's LookUp derivative of SRS also help people perform text searches in, interact with, and browse in the sequence databases. Both SRS and Entrez provide 'links' to associated databases so that you can jump from, for instance, a chromosomal map location, to a DNA sequence, to its translated protein sequence, to a corresponding structure, and then to a MedLine reference, and so on. They are very helpful!

What other types of bioinformatics databases are used?

Specialized versions of sequence databases include sequence pattern databases such as restriction and protease sites, promoter regions, and protein motifs and profiles; and organism or system specific databases such as the sequence portion of ACeDb (A *C. elegans* Database), FlyBase (*Drosophila* dataBase), and SGD (*Saccharomyces* Genome Database) and the Ribosomal Database Project.

In general two other general types of databases are also accessed in bioinformatics — reference and three-dimensional structure. Reference databases run the gamut from OMIM (Online Mendelian Inheritance In Man) to PubMed access to MedLine (NCBI's access to NLM' bibliographic database of over 4,000 biomedical journals). Other databases that could be put in this class include things like proprietary medical records databases and population studies databases.

Finally, the Research Collaboratory for Structural Bioinformatics (RCSB, a consortium consisting of three institutions: Rutgers University, San Diego Supercomputer Center at University of California, San Diego, and the National Institute of Standards and Technology) supports the three-dimensional structural Protein Data Bank (PDB). Other three-dimensional structure databases include the Nucleic Acid Databank at Rutgers (NDB) and the Cambridge small molecule crystallographic Structural Database (CSD).

II. So how does one do bioinformatics?

- A. 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 that . . .

Bioinformatics and the InterNet: the World Wide Web.

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

<u>Site</u>	<u>URL (Uniform Resource Locator)</u>	<u>Content</u>
National Center Biotech' Info'	http://www.ncbi.nlm.nih.gov/	databases/analysis/software
PIR/NBRF	http://www-nbrf.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 BioInfo'	http://www.bis.med.jhmi.edu/bioinformatics.html	databases/analysis/software
Harvard Bio' Laboratories	http://golgi.harvard.edu/	databases/analysis/software
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.public.iastate.edu/~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://phylogeny.arizona.edu/tree/phylogeny.html	overview of all phylogeny
Ribosomal Database Project	http://www.cme.msu.edu/RDP/	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 teaching and learning
CELLS alive!	http://www.cellsalive.com/	animated microphotography
Genetics Computer Group	http://www.accelrys.com/products/gcg_wisconsin_package	sequence analysis package

B. So what are the alternatives . . . ?

Other client-server protocols such as Network Entrez from NCBI; use and restrictions.

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. Omega, MacVector, DNAsis, DNASStar, etc.,

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

C. 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!

1. **Operating system:** command line operation hassles, communications software — telnet, ssh, xdmcp, etc. and terminal emulation, X graphics, file transfer — ftp, Mac Fetch, and scp, and editors — vi, emacs, pico (or desktop word processing followed by file transfer [save as "text only!"]). So here's a UNIX tutorial for you that we won't take the time to go through today, but I encourage you to do so at some point.

UNIX and Using X guides: 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 lack of credit giving and flagrant copyright infringement. Hundreds of local users are forever grateful; thank you. Steve Thompson, July, 1995 (updated several times since then).

GENERAL INTERFACE ISSUES

UNIX is an Operating System developed in the U.S.A., originally by BELL, then licensed to AT&T and now used in various implementations on many different brands of 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 Ethernet. 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 has often been 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. These file systems all consist of a tree of directories and sub-directories. Commands (or mouse clicks) exist which allow you to move about the directory structure. These commands are generally quite similar between systems. In UNIX these commands are as follows:

`cd` or `chdir` - change directory

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

In any tree structured file system the concept of where you are at 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 refer to a file by its complete 'path' name. When the complete name of a file is given the current position in the directory tree is ignored. 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.' Note that the password is not displayed on the screen as you enter it.

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 percent 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. The most common UNIX shell is the C-shell, although a very popular alternative C shell, 'tcsh,' is often available to maintain 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 simply type 'pwd' after you log in. The pwd command can be used at any point to keep track of whatever current directory you happen to occupy:

```
login: example
Password:

% pwd
/disk4/usr/local/people/example
```

To list the files in your home directory, use the 'ls' command. There are many options to the ls command; look at them 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 led 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. In the output from 'ls -a' the additional files '.login' and '.cshrc' are shown. Both of

these files are executed when the user is logged in, much like `AUTOEXEC.BAT` and `CONFIG.SYS` are executed in MS-DOS and `LOGIN.COM` is executed in VMS when you log in on those systems. The user can place commands in these files which can be used to customize each user's individual environment. Subdirectories 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. Along the same lines, but regarding file names, generally do not use any punctuation other than periods, hyphens, or underscores and do not put spaces in them.

```
% 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

% 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 which starts with a slash. This cat succeeds because the file name is completely specified as the complete path name and the file name. The following 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. Other file system commands:

```
rm - remove a file
mv - rename, i.e. 'move,' a file or group of files
cp - copy a file to another file or a set of files into a directory
```

YOUR LOGIN ENVIRONMENT

Nearly all operating systems have some way to customize your login environment with editable configuration files. On most UNIX systems there is a file which is executed for every login called '.login' and another one that sets up the 'shell' environment called '.cshrc.' Please take a look at these files; once you are comfortable with the operating system you may choose to modify them in order to customize the manner in which your account interacts with the system. A common customization is to automatically activate GCG upon login with the command 'gcg' at the bottom of your .login file.

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.

```
cmd -options parameters | cmd2
```

This feature is called "piping" the output of one program into the input of another. Some useful examples of command redirection and piping are shown below with the ls command and the more command which allows paging through directory listings:

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

THE USE OF WILDCARDS AS FILE PARAMETERS

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. The special characters which cause wild card searching to be performed are as follows:

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

The following are some examples of commands using wild card characters:

```
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 subdirectories 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 subdirectories below your current directory.

The `grep` command is very useful as it allows searching through many files for a pattern. 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 which called the `fopen` subroutine, you could use the following command:

```
grep fopen */*.c
```

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).

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.
```

Command to change your password:

`passwd` Change your login password

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

<code>uptime</code>	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.
<code>who</code>	Shows who is logged on to the system.
<code>w</code>	Shows who is logged in to the system doing what.
<code>top</code>	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.)
<code>finger</code>	Gets information about a user or machine.
<code>ps</code>	Shows your current processes and their status (running, sleeping, idle, terminated, etc.); (use <code>man ps</code> as options widely vary, see especially <code>-a</code> , <code>-e</code> , <code>-l</code> , and <code>-f</code>).
<code>at</code>	Submit script to the at queue for execution later.
<code>bg</code>	Resumes a suspended job in background mode.
<code>fg</code>	Brings a background job back into interactive mode.

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

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

Directory commands:

<code>pwd</code>	Print Working Directory. Shows you where you are at in the file system. Very useful when you get confused.
<code>ls</code>	Shows (lists) your files' names.
<code>ls -l</code>	Shows your files' names in extended (long) format including file size, ownership, and permissions.
<code>ls -al</code>	Shows all files including the system files (.files) in your directory in the long format.
<code>mkdir dir</code>	Makes a new directory in your current directory.
<code>rmdir dir</code>	Removes a sub-directory from your current directory. Directory must be empty to remove the directory.
<code>rm -r dir</code>	Removes all the files, and subdirectories of a directory and then removes the directory. Very convenient, useful and <u>dangerous</u> .
<code>cd</code>	Move back into your home directory from anywhere.
<code>cd dir</code>	Move down into a directory 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:

<code><Ctrl c></code>	Aborts a running process (program); no option for restarting it later.
<code><Ctrl d></code>	Terminates a UNIX shell, i.e. exit present control level and close the file. Use 'logout' to exit from your top level login shell.
<code><Ctrl z></code>	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 <code><Ctrl z></code> which is more like UNIX's <code><Ctrl d></code> .
<code>kill -9 psid</code>	Kills, using the "sure kill" option, a process with the given process ID. 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, e.g. zen.art.motorcycle.com):

<code>ftp host</code>	File Transfer Protocol. Allows a limited set of commands (dir cd put get etc.); help gives help
<code>telnet host</code>	Connects to another ethernet connected host.
<code>ssh host</code>	Secure, encrypted connection to another ethernet host.
<code>pine</code>	E-mail access/use (not always installed; mail is default UNIX mailer).
<code>talk user</code>	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.

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

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 and information about them	SHOW USERS	who 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

SPECIFIC DIFFERENCES BETWEEN THE VMS AND UNIX GCG PACKAGE

This section is intended for people who are already familiar with the VMS version of the GCG Package. It explains the differences between running the GCG Package in the VMS and UNIX environments. In both the VMS and UNIX versions of the GCG Package, you 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
```

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

Certain printing (non-control) characters have special meaning to the UNIX shell (see man csh for more information). Although these characters, called shell metacharacters, may have no particular meaning in a VMS environment, they do have meaning in your UNIX environment. 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. 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'. It is an error to use a globbing character in the absence of any matching files. Because '*' and '?' are frequently used characters for specifying families of database entries, GCG provides an alias for each GCG command that disables globbing. Therefore, you can use these characters on your command line when invoking GCG programs.

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 is becoming the defacto standard operating system in more and more computing environments today, particularly scientific computing, so the effort will not be wasted.

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, as sometimes 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. On most systems:

```
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 the new window in order to maintain control of the initial window. Some helpful options supported in most versions of xterm are "-ls" so that your login script is read, "-sb -sl 100" to give you a 100 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, authorize X access to the host with the xhost command:

```
> xhost +zen.art.motorcycle.com          (may not be necessary)
```

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

```
> ssh -X thompson@zen.art.motorcycle.com  (-X sets the X environment for you)
```

This should produce a login window. Log in as usual, then, if necessary, issue the following command on the host to setup the X environment (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  (again, may not be necessary)
```

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

After GCG has initialized, you can run "setplot" choosing the appropriate choice to produce a colored GCG X graphics window. Run commands from the xterm window. Graphics will be displayed in the graphics window. Another option is to launch the Wisconsin Package Graphical User Interface by typing "seqlab &." This graphical user interface provides GCG functions from a point and click menu interface. More information on SeqLab is available through GenHelp.

X server emulations — on PCs and Macs: On pre-OS X Macintoshes I have tested both MacX and eXodus. MacX is by far the easiest of all the X servers I have tested to install and configure though eXodus seems a bit more robust. Tenon's Xtools is a good bet for getting a true X environment under Apples' new 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.

2. **The Genetics Computer Group** — the Wisconsin Package for Sequence Analysis. Begun in 1982 in Oliver Smithies' lab at the Genetics Department 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 Pharmacia 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.

a. **Specifying Sequences and Logical Terms!** To answer the always perplexing GCG question — “What sequence(s)? ..” Specifying sequences, GCG style; in order of increasing power and complexity:

i. 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.seq`). Use the program 'reformat' to convert 'raw' text format files to GCG format (first use 'chopup,' if the sequence is one continuous line without line feeds).

```
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 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.
```

```
example.seq Length: 77 July 21, 1999 09:30 Type: N Check: 4099 ..
```

```
1 ACTGACGTCACATACTGGGACTGAGATTTACCGAGTTATACAAGTATACAGATTTAATAGCATGCGATCCCATGGGA
51 GATTTAATAGCATGCGATCCCATGGGA
```

ii. 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 accompanying list. 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 EFTu sequences in SwissProt. Because all the sequences are available in local GCG databases, it is seldom necessary to put individual sequences in your account.

iii. 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 names, annotation, and actual sequence data. 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 1 α and Tu factors, one may use a naming system such as the following: efla-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.rsfeftu_ecoli} to point just to the *E coli* sequence or EF1a-Tu.rsfeftu_*} to point at all of the EfTu's as long as you use a sequence naming convention that retains this convention.

- iv. 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 1 $\alpha$  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
Efla-Tu.msf{*}
/usr/accounts/test/another.rsfeftu_*}
@another.list
```

b. Logical terms for the Wisconsin Package.

Sequence databases, nucleic acids:

GENEMBLPLUS	all of GenBank plus abridged EMBL plus EST and GSS	GB	all of GenBank except the EST and GSS subdivisions
GENEMBL	all of GenBank plus abridged EMBL except EST and GSS	GENBANK	all of GenBank except the EST and GSS subdivisions
GE	all of GenEMBL	GB_BA	GenBank bacterial subdivision
BA	GenEMBL bacterial subdivisions	GB_EST	GenBank EST (expressed sequence tags) subdivision
BACTERIAL	GenEMBL bacterial subdivisions	GB_GSS	GenBank GSS (genome survey sequences) subdivision
EST	GenEMBL EST (expressed sequence tags) subdivisions	GB_IN	GenBank invertebrate subdivision
GSS	GenEMBL GSS (genome survey sequences) subdivisions	GB_OM	GenBank other mammalian subdivision
IN	GenEMBL invertebrate subdivisions	GB_OV	GenBank other vertebrate subdivision
INVERTEBRATE	GenEMBL invertebrate subdivisions	GB_PAT	GenBank patent subdivision
OR	GenEMBL organelle subdivisions	GB_PH	GenBank phage subdivision
ORGANELLE	GenEMBL organelle subdivisions	GB_PL	GenBank plant subdivision
OM	GenEMBL other mammalian subdivisions	GB_PR	GenBank primate subdivision
OTHERMAMM	GenEMBL other mammalian subdivisions	GB_RO	GenBank rodent subdivision
OTHERMAMMAL	GenEMBL other mammalian subdivisions	GB_ST	GenBank structural RNA subdivision
OV	GenEMBL other vertebrate subdivisions	GB_STS	GenBank STS (sequence tagged sites) subdivision
OTHERVERT	GenEMBL other vertebrate subdivisions	GB_SY	GenBank synthetic subdivision
OTHERVERTEBRATE	GenEMBL other vertebrate subdivisions	GB_TAGS	GenBank Tags subdivisions
PAT	GenEMBL patent subdivisions	GB_UN	GenBank unannotated subdivision
PATENT	GenEMBL patent subdivisions	GB_VI	GenBank viral subdivision
PH	GenEMBL phage subdivisions		
PHAGE	GenEMBL phage subdivisions	EM	all of abridged EMBL except the TAGS subdivisions
PL	GenEMBL plant subdivisions	EMBL	all of abridged EMBL except the TAGS subdivisions
PLANT	GenEMBL plant subdivisions	EM_BA	EMBL bacterial subdivision
PR	GenEMBL primate subdivisions	EM_EST	EMBL EST (expressed sequence tags) subdivision
PRIMATE	GenEMBL primate subdivisions	EM_FUN	EMBL fungal subdivision
RO	GenEMBL rodent subdivisions	EM_GSS	EMBL GSS subdivision
RODENT	GenEMBL rodent subdivisions	EM_IN	EMBL invertebrate subdivision
ST	GenEMBL structural RNA subdivisions	EM_OM	EMBL other mammalian subdivision

STRUCTURAL	GenEMBL structural RNA subdivisions	EM_OR	EMBL organelle subdivision
STRUCTURAL_RNA	GenEMBL structural RNA subdivisions	EM_OV	EMBL other vertebrate subdivision
STS	GenEMBL (sequence tagged sites) subdivision	EM_PAT	EMBL patent subdivision
SY	GenEMBL synthetic subdivisions	EM_PH	EMBL phage subdivision
SYNTHETIC	GenEMBL synthetic subdivisions	EM_PL	EMBL plant subdivision
TAGS	GenEMBL EST and GSS subdivisions	EM_PR	EMBL primate subdivision
UN	GenEMBL unannotated subdivisions	EM_RO	EMBL rodent subdivision
UNANNOTATED	GenEMBL unannotated subdivisions	EM_STS	EMBL STS (sequence tagged sites) subdivision
VI	GenEMBL viral subdivisions	EM_SY	EMBL synthetic subdivision
VIRAL	GenEMBL viral subdivisions	EM_TAGS	EMBL Tags subdivisions
		EM_UN	EMBL unannotated subdivision
		EM_VI	EMBL viral subdivision

Sequence databases, amino acids:

GENPEPT	GenBank CDS translations
GP	GenBank CDS translations
SWP	all of Swiss-Prot and all of SPTreMBL
SWISS	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 translations
SPT	Swiss-Prot preliminary EMBL translations
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:

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

c. SeqLab — a brief history — Steve Smith's GDE + GCG's WPI.

While working on bacterial ribosomal RNA phylogenies with Walter Gilbert and Carl Woese, Steve Smith realized the 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 PHYLLIP routines and Clustal alignments. He called it the "Genetic Data Environment." 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 graphical user interface (GUI) for the Wisconsin Package. They called it the "Wisconsin Package Interface." Nobody was impressed — it was a terrible attempt. It only provided a menu to their programs, hardly anything more than the "-check" option they've always had. So they did a natural and very smart thing. They hired Steve Smith away from Millipore, where he had newly moved, into their company, so that he could merge his GDE with their WPI. The offspring was SeqLab, and, thank goodness, they threw away the acronyms. As 'they' say "The rest is history" and once more GCG's customers are (generally) happy.

SeqLab, an X-based GUI to the Wisconsin Package — and some illustrative examples: Glutathione Reductase, G-protein coupled TM7 receptors, primate prions, Human Papilloma Virus L1 major coat protein, Major Histocompatibility Class II, Vicilin seed storage proteins, and Elongation Factor 1 α /Tu.

III. For more information participate in the rest of the workshop. An assignment . . .

If you still want to learn more, read the Introduction to my tutorial, section 1 for about the first twenty pages, skim through the remainder of the material, and then work through the tutorial examples, either in the hands-on workshop, or at your own time. **The tutorial** — the 'lower' eukaryotic protein Elongation Factor 1 α .