

## Building a Multiuser Sequence Analysis Facility Using Freeware

Brian Fristensky

### 1. Introduction

Although many commercial packages exist for molecular sequence analysis, they are typically expensive. Whereas many Web-based applications are available for sequence analysis, the Web interface cannot store data on remote servers and is awkward to use. A good alternative is to build a sequence analysis facility on a local server. *BIRCH*, the *Biological Research Computer Hierarchy*, is an example of such a system (<http://home.cc.umanitoba.ca/~psgendb> and **ref. 1**). *BIRCH* is best thought of as a workbench containing tools for working with sequences, as well as software that minimizes the problems of putting tools together to perform a task. For example, in **Fig. 1**, several steps in phylogeny construction from an alignment were performed automatically. It is not possible to provide detailed instructions on installing all of the 300+ programs that currently reside in *BIRCH*. Rather, my purpose is to outline the strategies and tricks that make building and maintaining a sequence facility a smooth ongoing task.

### 2. Hardware, Software, and Know-How

#### 2.1. Hardware and Operating System

*BIRCH* is currently implemented on a Sun workstation running Solaris 2.5. Since source code is available for most freeware programs used in *BIRCH*, it should be possible to recompile for other platforms. Almost all of the programs implemented in *BIRCH* have run under *LINUX*, and many have run on other UNIX platforms. If you are building *BIRCH* on an existing multiuser system,

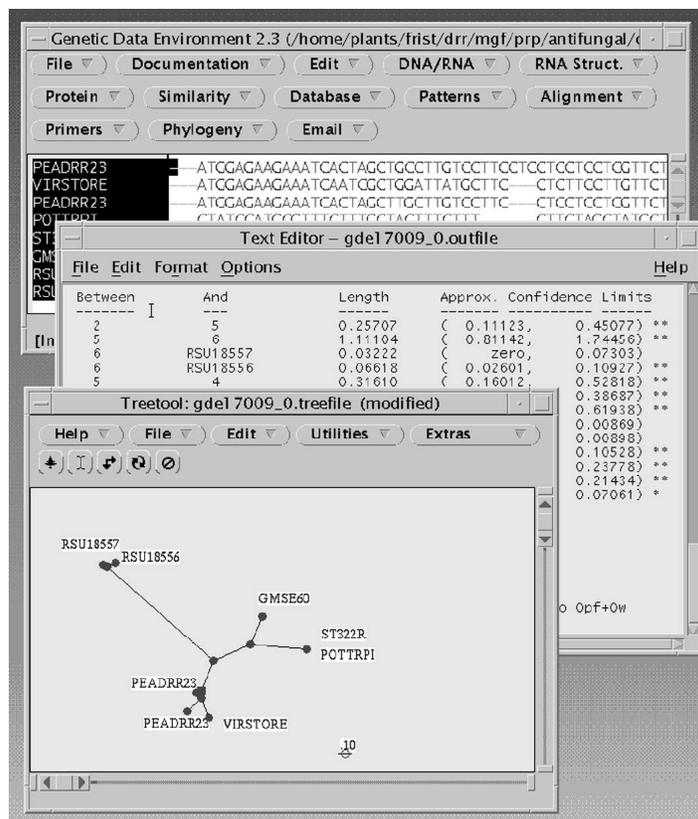


Fig. 1. Automated phylogeny. In *GDE* (2), aligned sequences were selected, and *fastDNAmI* (3) was called to produce a phylogeny. *GDE* automatically calls a text editor and *treetool* (4) to display results.

you will also need *X*-terminals or PC-based *X*-terminal emulators. When you log into your account, all programs run on the server, and *X-Windows* displays everything at the terminal. In comparison to a PC, where each machine has different software, data and hardware, you can log into the server from any terminal. *X-Windows* is one implementation of network computing, which is described in more detail in ref. 5.

## 2.2. Software

A C compiler, preferably *GNU C*, and a Web browser are necessary. *Net-scape Communicator* is recommended because it comes with a visual HTML editor.

### 2.3. Know-How

You will need a working knowledge of UNIX (6), some previous experience in programming (preferably C, C++, or Java), and an ability to write HTML. Even if you don't know all of these things now, this is a good opportunity to gain these practical skills.

## 3. Setting up BIRCH

### 3.1. Create an Administrator's Account

Don't set up *BIRCH* on your own personal account. Get a separate account solely for this purpose. This keeps all of the *BIRCH* directories integrated as a separate unit. On your personal account, you are just another user, which is the only way to really test whether *any* user can run any program. As a corollary, avoid doing preliminary installations on your personal account. Also, even if you have root privileges, avoid working as root except for systems level (i.e., non-*BIRCH*) activities.

### 3.2. Create a Directory Hierarchy

The directories needed to construct *BIRCH*, and their current sizes at our site, are as follows:

**GenBank:** *GenBank* DNA nucleic acid database (5.9 gb, release 106, April 1998).

**PIR:** 183 mb, release 55, February 1998.

**admin:** for administrative files and scripts. (0.4 mb).

**bin:** for executable files (60 mb).

**dat:** special data files for programs (e.g., scoring matrices) (5 mb).

**doc:** documentation files (18 mb).

**install:** working directory for software installation.

**java:** for Java classes.

**man1:** documentation in UNIX manual page format.

**ncbi:** directories for National Center for Biotechnology Information (NCBI) client/server programs (1.7 mb).

**public\_html:** directory for the *BIRCH* web site (2.5 mb).

This directory hierarchy can be downloaded from <http://home.cc.umanitoba.ca/~psgendb/build/build.html>. Initially, all these directories contain a series of shell scripts and datafiles for managing *BIRCH*. To ensure that the most recent versions of software are installed, it is best to download each program or package as you build your local *BIRCH* site.

These directories should be installed in *BIRCH* administrator's **\$HOME** directory. On our system, the *BIRCH* **\$HOME** directory is **/home/psgendb**.

We have set an environment variable, **\$db**, to store this path. That is, when interpreting a command, the shell will replace '**\$db**' with **/home/psgendb**. The administration directory **/home/psgendb/admin** can therefore be typed as '**\$db/admin**' by any user on the system.

Whenever you install a program package, create specific subdirectories for that package in doc and dat, for their documentation and data (if any) respectively. In fact, it is a good working rule that when you start a new project of any kind, always create a directory specifically for that task, even if it is only temporary.

There are several important rules for these directories:

1. All directories and files, including the *BIRCH* **\$HOME** directory, must be world-readable.
2. All directories must be world executable.
3. All programs must be world executable.

### 3.3. Configure the Administration Files

At our site, *BIRCH* contains over 300 programs and two major databases. The programs were written by different authors in different languages on different platforms using different file formats. In many cases they need to know the locations of datafiles, databases, or runtime libraries. If these things had to be set for each user, and changed by each user every time a new program was installed or updated, nothing would ever work. Fortunately, there is a clean solution to all these problems. All settings are read from **\$db/admin**. Never deviate from this rule! The user should never have to configure his/her account for anything. (At the University of Manitoba, *BIRCH* has over 140 users. Imagine having to change settings for each user!)

Two files contain settings needed by all programs in *BIRCH*. **\$db/admin/login.source** contains commands to be executed each time a user logs in. The most important command adds **\$db/bin** to the user's **\$PATH** environment variable. When the shell reads a command, the first nonblank string is interpreted as the name of a command. The shell searches for an executable file in every directory listed in **\$PATH**. Thus, if all *BIRCH* programs are in **\$db/bin**, all we have to do is to add **\$db/bin** to **\$PATH**, and the user can run any program. **login.source** also contains a command to print the contents of **\$db/admin/Login\_Message**, a file containing short announcements of interest to *BIRCH* users.

**\$db/admin/cshrc.source** contains commands that need to be executed every time a new shell is started, e.g., when a window is opened, or a program is run.

Virtually all program settings are defined here. Most of this file contains commands to set environment variables. For example,

```
# Environment variables for sequence work.
# Upper and lowercase are supported.
setenv DB      /home/psgendb
setenv db      $DB
setenv DATA   $DB/dat
setenv data    $DATA
setenv DAT     $DATA
setenv dat     $DATA
setenv GENBANK $DB/GenBank
setenv gb      $GENBANK
```

Here is where we define **\$db**, and then use it to build other environment variables telling where data files are stored.

Each program or package may have specific settings as well. For example, the NCBI programs are configured as follows:

```
#NCBI
setenv NCBI $db/ncbi
alias entrez Nentrez
```

`setenv` tells the NCBI programs where to find necessary directories. The `alias` line tells the shell that when a user types "entrez", the network version of *entrez* (*Nentrez*) should be run.

To use *BIRCH*, the user must run **\$db/admin/newuser**. This script adds a line reading

```
source /home/psgendb/admin/login.source
```

to the user's `.login` file, and

```
source /home/psgendb/admin/cshrc.source
```

to the user's `.cshrc` file. These two lines cause all commands in the `.source` files to be executed when the user logs in or starts a new shell, respectively. In this way, any change or addition to the `.source` files in **\$db/admin** will automatically take effect for every user. The *BIRCH* administrator should never have to do anything to a user's account. This has worked very well, in practice.

**login.source** and **cshrc.source** will need to be modified to reflect local directory structures and installed software. For example, the **\$db** environment variable will have to be changed to your local *BIRCH* **\$HOME** directory. In **login.source** and **cshrc.source**, it is best to comment out all lines that refer to programs or databases that are not yet installed. These lines can be uncommented as *BIRCH* grows.

### 3.4. Create a Web Site

Consider the *BIRCH* web site to be your conceptual model of what you are building. Yes, it is also there to tell the user how to use the system and what is available, but the complexity of *BIRCH* demands a well-structured road map. Always have a copy of *Netscape* running on your screen, so that you can create web pages and modify them as you go. Because *BIRCH* is already documented on the University of Manitoba Web site (*I*), you can often shortcut by downloading web pages and modifying them to meet your needs. The instructions in the following section assume that a web page exists called **programs.html** (see <http://home.cc.umanitoba.ca/~psgendb/programs.html>). This page contains links to all documentation, organized by category.

## 4. Building *BIRCH*

This section describes the overall process for installing several software packages, each chosen to illustrate some of the subtle problems that can be associated with getting programs to work for distributed users. The goal of this section is to provide a short path to getting a reasonably comprehensive suite of programs working quickly. This core of programs serves as the foundation for building a facility tailored to the needs of your local user base. For brevity, URLs from which programs can be downloaded are included in the references. Programs will usually include instructions for installation that are more detailed than what I can present here.

Whereas it is best to install programs on the administration account and test them on your personal account, it would be inconvenient to keep going back and forth between accounts. There are two ways around that problem. The ideal solution would be to have two *X*-terminals side by side, each logged into a different account. Because that is not always possible, the next best thing is to run an *X-Windows* session on your personal account, but log into the administration account in one or more windows. For example open up a command window (e.g., terminal window in *CDE*) and log into your administration account using *telnet*. For simple tasks, keep one or more *telnet* sessions logged into the administration account, one for each working directory. To get *X11* programs to run on the administration account, but display on your personal

account see the script `$db/admin/xdisplay`. It will have to be modified for your own site. If you are using the CDE desktop manager, it may prove less confusing to keep all windows from your personal account on one screen, and all windows from your administration account on a separate screen. Also, the *BIRCH* `newuser` script causes your UNIX prompt to display both the server name and the current directory, which should help you keep track of which window belongs to which account.

#### 4.1. Install readseq (7)

The biggest single problem with sequence software is the plethora of file formats that must be used. *readseq* is a program that converts one format (e.g., *GenBank*) to another (e.g., GCG). The *readseq* source code and documentation are downloaded as a shell archive file, **readseq.shar**. To recreate the files in the archive type `sh readseq.shar`. You can compile *readseq* for your platform by typing `make`, which will create the executable file **readseq**. Make this file world readable

```
chmod a+rx readseq
```

and move it to the bin directory

```
mv readseq $db/bin
```

Also, create a directory to hold the help file

```
mkdir $doc/readseq
chmod a+rx $doc/readseq
```

and move the help file to this directory

```
mv readseq.help $doc/readseq/readseq.asc
```

Normally, I prefer not to rename files from other packages. However, because Web browsers often vary with regard to how they handle different file extensions, it is preferable to have a uniform file extension for all *ASCII* files. I use “.asc” for *ASCII* files. Finally, add a link for **readseq.asc** to **programs.html**.

Read the documentation for *readseq* and test the program on your personal account. For example, if you have a GenBank file called **PEADRRRA.gen**, typing

```
readseq -p -oPEADRRRA.wrp -fPearson <PEADRRRA.gen
```

will create a file in Pearson/*FASTA* format called **PEADRRRA.wrp**. *readseq* was originally developed under VMS, so the “-p” switch is necessary to pipe input to the program using the UNIX input redirection character “<”.

#### 4.2. Install FSAP (8,9)

Many programs come in packages. The *FSAP* package includes programs for many common sequence tasks (e.g., printing sequences, translation, restriction site searches) all of which are run through interactive text-based menus. In this case, the package can be downloaded as a .tar archive, **fsap.tar.Z**. To recreate the directory hierarchy for *FSAP*, first uncompress the file

```
uncompress fsap.tar.Z
```

And then type

```
tar xvf fsap.tar
```

to create the **fsap** directory. If you type `ls -l` in the `fsap` directory, you should see the following:

```
drwx--- 2 frist drr    512 Jun  4 1996 GDE/
-rw--- 1 frist drr   7041 May  3 1996 INSTALL.doc
-rw--- 1 frist drr    970 May  3 1996 RELEASE.NOTES
drwx--- 2 frist drr    512 May  3 1996 bin/
drwx--- 2 frist drr    512 May  3 1996 dat/
drwx--- 2 frist drr    512 Mar  6 18:56 doc/
drwx--- 4 frist drr    512 May  3 1996 src/
drwx--- 2 frist drr    512 May  3 1996 src.c/
drwx--- 2 frist drr   1024 May  3 1996 test/
```

**INSTALL.doc** contains step-by-step installation instructions. **src.c** contains C source code, which generates executable code. **doc** and **dat** contain, respectively, documentation and datafiles used by the programs. *Genetic Data Environment (GDE)* contains menu items and c-shell scripts that make it possible to run these programs through *GDE*. **test** is a directory in which you can run a script that will test all the programs to make sure that they function on your system. Many packages will have test scripts. When you have successfully tested the programs, installation is easy. Copy the contents of **fsap/bin** to **\$db/bin**, **fsap/dat** to **\$dat/fsap/dat**, and **doc** to **\$doc/fsap/doc**. Make sure to add links for these documentation files in **programs.html**.

Again, log into your personal account and try out these programs. The first one to try is **numseq**, as described in **\$doc/fsap/numseq.asc**. Any *GenBank* flat file will suffice for testing this program.

### 4.3. Install FASTA (10)

The *FASTA* package provides programs both for pairwise and database sequence comparisons. Compilation is done using the UNIX **'make'** command, and installation is as simple as copying executable files to **\$db/bin**. This should be one of the easiest packages to install. One twist, though, is that the documentation is in the UNIX manual page format. *BIRCH* has a directory for manual pages called **\$db/manl**. All files in this directory should be in the form 'name.l' (where 'l' stands for local). In **login.source**, the line

```
setenv MANPATH $MANPATH\:$DB
```

tells UNIX to look for manual pages in this directory, as well as in any other directory specified in the system's **\$MANPATH**.

For example, to read the documentation for *align*, the user types 'man align', and the file **\$db/manl/align.l** will be displayed.

It is also useful to create ASCII files from these manual pages for display by the web browser. To create an ASCII file for **align.l**, type `man align > $doc/fasta/align.asc`. Remember to make this file world readable, and create a link in **programs.html**.

### 4.4. Install GDE (2)

*GDE*, is a program that runs other programs. As illustrated in **Fig. 1**, *GDE* combines a multiple sequence alignment editor with a set of pull-down menus. For example, the **Similarity** menu contains calls to most of the *FASTA* similarity programs. The thing that makes *GDE* unique is its ability to have menus and menu items added with no reprogramming or recompiling. When *GDE* is launched, a file called **\$GDE\_HELP\_DIR/.GDEmenus** is read, specifying the contents of each menu, and the commands to be executed to run each program. For example, the *lfasta* menu is shown in **Fig. 2**.

In **.GDEmenus**, the entry to create this menu begins like this:

```
#----- LFASTA ( 7/26/95) -----
item:LFASTA - Fast local alignment
itemmethod:(sed "s/[#%]/>/"<inl >inl.tmp; readseq
inl.tmp -i1 -f8 > inl.seq1; readseq inl.tmp -i2
-f8 >inl.seq2; lfasta -w $RESPERLINE $MARKX -d
$NUMOFALN $MATRIX inl.seq1 inl.seq2 $KTUP >
inl.out;
fastaout.csh $MARKX inl.out; rm inl*) &
itemhelp: FASTA/fasta.asc
```

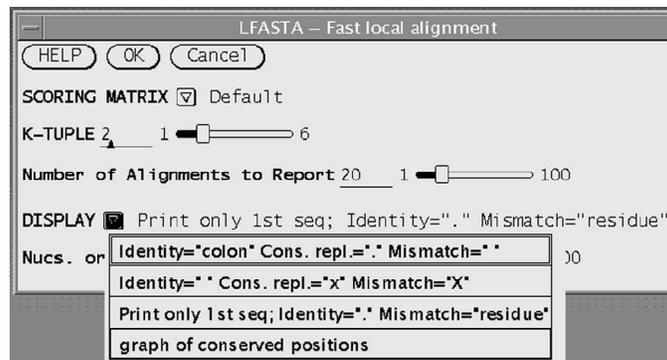


Fig. 2. Example of a *GDE* menu, illustrating pull down menus and sliders. This menu displays when *lfasta* is chosen from the **Similarity** menu.

The most important line is the `itemmethod`, which contains a string of commands to be run. For example, *readseq* is called to convert the selected sequence to *FASTA* format, and arguments are inserted into the command, each preceded by a '\$'. Each argument to *lfasta* can be specified in a few lines, such as that for the pull-down menu **DISPLAY**, shown in **Fig. 2**:

```
arg:MARKX
arglabel:DISPLAY
argtype:choice_menu
argchoice:Identity="colon" Cons. repl.= "." Mismatch=" " :-m 0
argchoice:Identity=" " Cons. repl.="x" Mismatch="X" :-m 1
argchoice:Print only 1st seq; Identity="." Mismatch="residue" :-m 2
argchoice:graph of conserved positions :-m 4
argvalue:0
```

Whereas it is easiest to get *GDE* to run programs such as *readseq* that take all information from the command line, even interactive programs requiring user input can be called by *GDE*. For example, to run *numseq*, *GDE* sends the parameters set in the menu to a script called **numseq.csh**. **numseq.csh** reads the parameters and generates keystrokes that would normally be typed by the user in response to prompts by *numseq*. The ease with which new programs can thus be added to *GDE*'s menus makes *GDE* the foundation from which most of *BIRCH* is run. A **.GDEmenus** file and accompanying shell scripts necessary to run most of the programs in *BIRCH* can be downloaded from **ref. 1**.

#### 4.5. Install NENTREZ (11), SEQUIN (12), BLASTCLI (13), and Cn3D (14)

The NCBI suite consists of networked client/server applications. *Nentrez* is a client that runs on your desktop, allowing text searches and sequence retrieval from the NCBI server. Its helper application, *Cn3D*, can download and display three-dimensional protein structures from structural databases at NCBI. *sequin* automates the process of annotating new sequences and submitting them to *GenBank*. *sequin* can also download sequences from NCBI for resubmission as updates. *blastcli* is a local client that submits sequences to the NCBI *BLAST* server.

Because these programs share a common directory containing configuration and datafiles, it makes sense to install them all at once. Generally, installation is as simple as copying the executables to **\$db/bin** and running *netentcf*, the network client configuration program. All files and directories for these programs should be in a directory specified by **\$NCBI** in **\$db/admin/cshrc.source**. The first time you run *Nentrez*, a file called **\$HOME/.ncbirc** will be created, containing configuration information. If you move this file to **\$NCBI**, it will work for all users.

The workspace menu (**Fig. 3**) can be a valuable means of making it easy for users to know which programs are available on the system. All UNIX window managers have a configurable Workspace menu. The *CDE* manager is available on most UNIX platforms and is now the default on many. Therefore, I have created a **\$db/.dt/dtwmrc** file to configure the *CDE* workspace menu for all *BIRCH* users. Programs are organized into submenus (e.g., Word Processing, Statistics, Molecular Biology). To add *sequin* to the menu,

```
"Sequin - submit seq. to GenBan1" f.exec /home/psgendb/bin/sequin
```

must be in **dtwmrc**. The setup script **\$db/bin/menusetup** replaces the user's **dtwmrc** file with a symbolic link to **\$db/.dt/dtwmrc**. Thus, as the *BIRCH* administrator updates this file, all users get the new menu.

*GDE* is conspicuously absent from this menu. This is a deliberate omission. If launched from the workspace menu, *GDE* will default to the **\$HOME** directory for reading and writing files.

When working with *GDE*, it is best to create a separate directory for each project, e.g., phylogenetic analysis of a multigene family. If you **cd** to that directory and launch *GDE* from the command line, all file input and output

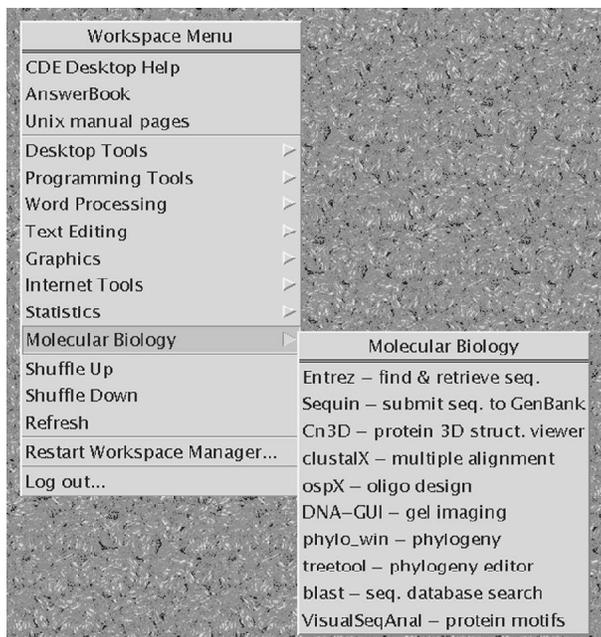


Fig. 3. Customized *CDE* workspace menu. Submenus, organized by category, make it easy for users to find and launch programs.

(including the creation of temporary files and directories) will occur in this directory.

#### 4.6. Install XYLEM (15)

*XYLEM* is a set of tools for local database management. Although designed originally for creating subsets of *GenBank* or *PIR* files for projects such as phylogenetic studies, *XYLEM* can also be used for keyword searches and retrieving entries from these databases. **features** automates the process of extracting *GenBank* features (e.g., exon, intron, mRNA, CDS) from large sets of entries.

Installation for *XYLEM* is almost identical to installation for *FSAP*. The next section assumes that *XYLEM* is installed on your system.

#### 4.7. Install GenBank and PIR

With programs like *Nentrez* and *blastcli*, it may not be necessary to have local versions of sequence databases. Since *GenBank* release 106 required 6 gb of space, this is an important consideration. However, a local copy of the database is useful for several reasons. First, networked *BLAST* programs do not

give you the option of limiting your searches to specific *GenBank* divisions. Local database search programs such as *FASTA* can be tailored for specific needs. Finally *Nentrez* does not allow input of a group of accession numbers for retrieval of groups of sequences.

Automated downloading and formatting of *GenBank* and *PIR* is done through shell scripts called **gbupdate** and **pirupdate**. For example, the names of *GenBank* files to be downloaded are found in **\$gb/master.filelist**. To begin after peak hours (e.g. after 7:00 P.M.) use the **at** command:

```
at 7pm
at>nice gbupdate master.filelist
at><ctrl>-D
```

The UNIX **nice** command runs the job at a low priority so that real time tasks (e.g., moving windows around the screen) will not be slowed down. **gbupdate** downloads each file, verifies that the downloaded file is the same size as the original, uncompresses the file, and for *GenBank* sequence files, runs *splitdb (15)* to split each *GenBank* division into annotation, sequence, and an index. The sequence files are written in *FASTA* format. Separating sequences and annotation into separate files speeds both *FASTA* searches of sequence and **findkey** searches of the annotation (*15*). *fetch* can retrieve sets of *GenBank* entries, rejoining annotation and sequence to recreate the original entries (*15*). Larger *GenBank* divisions are now split among several files. The EST division was split among 22 files, gbest1-gbest22 in release 106. **master.filelist** needs to be updated with each download to reflect these changes. However, the **fetch** and **findkey** automatically detect when divisions are split.

## 5. Training Users

Whereas it is vital to keep the documentation for *BIRCH* consistent and complete, human nature is such that generally people do not read it. Hands-on training sessions can be of great value to the user community, both in terms of teaching people how to use *BIRCH*, as well as in creation of a core of trained users who can help others.

Because most *BIRCH* users are also new to UNIX, it may seem a daunting prospect to cover both areas. Nonetheless, each year during the lab component of my course, Introductory Cytogenetics (*16*), students with no previous UNIX or bioinformatics background have learned enough to complete a simple sequence project over two hands-on sessions. At the end of the second session, each student is given a 300- to 400-bp unknown sequence, derived from the protein-coding sequence of a *GenBank* entry. Each student must be able to identify the parent sequence using *FASTA*, retrieve the parent sequence, iden-

tify the coding sequence from which the unknown was derived, print the entire coding sequence, with translation in the correct reading frame.

The sessions are run as follows:

1. **On screen demo** (30 min.): Starting with a demo gives students an idea of what things should look like. Using an *X*-terminal connected to a 1024 x 768 projector, I briefly explain how *X-Windows* works, and the basics of the *CDE* desktop. I also demonstrate examples of sequence analysis, using both command-line programs and programs run through *GDE*.
2. **Hands-on demo of UNIX, CDE, and simple sequence tasks** (2 h): Demos proceed step by step, making sure that everyone has successfully completed each step before the class moves on. It is valuable to have an assistant to help students when they encounter difficulties. First students run setup scripts **newuser** and **menusetup** found in **\$db/admin**. Then students are introduced to the fundamentals of *CDE*, and use of a Web browser for reading documentation. Next, students learn about working with sequences by running *numseq* from the command line. *numseq* can be used to illustrate the ramifications of working with either one or both strands of a DNA sequence, the differences between linear and circular molecules, and how to translate sequences. Students then launch *GDE* and try repeating some of the same tasks, running *numseq* from a *GDE* menu. (Although graphic interfaces are good, it is still best to give people some exposure to the command line. Doing so provides important insights into what it is that the computer actually does.) The session ends with a quick discussion of *GenBank*. Students learn to search for sequence by keywords and to retrieve them.
3. **Similarity searches and Databases** (2 h): The second session opens with a short discussion of the theory behind both dot-matrix (9) and global (17,10) similarity searches. The concepts of look-up tables and optimal alignments are emphasized. Using *GDE*, students do pairwise comparisons of several related sequences, using *d4hom* (9) for dot-matrix searches and *align* (10) for global alignment. Finally, students run *fasta* to search for a DNA sequence in *GenBank*.

## References

1. Fristensky, B. *BIRCH*. <http://home.cc.umanitoba.ca/~psgendb>.
2. Smith, S., Overbeek, R., Woese, C. R., Gilbert, W., and Gillevet, P. M. (1994) The tenetic data environment: an expandable GUI for multiple sequence analysis. *Comp. Appl. Biosci.* (now *Bioinformatics*) **10**, 671–675. <ftp://megasun.bch.umontreal.ca/pub/gde/>.
3. Olsen, G. J., Matsuda, H., Hagstrom, R., and Overbeek, R. (1994) FastDNAm1: a tool for construction of phylogenetic trees of DNA sequences using maximum likelihood. *Comput. Applic. Biosci.* (now *Bioinformatics*) **10**, 41–48.
4. Maciukenas, M. (1994) Treetool. <ftp://rdp.life.uiuc.edu/rdp/programs/TreeTool/>.
5. Fristensky, B. (1999) Network computing: restructuring how scientists use computers, and what we get out of them, in *Methods in Molecular Biology* vol. 132.

- Bioinformatics Methods and Protocols* (Misener, S. and Krawetz, S. eds.), Chapter 22. Humana Press, Totowa, NJ.
6. Sobell, M. G. (1995) *A Practical Guide to the UNIX System*. Addison-Wesley Publishers.
  7. Gilbert, D. (1993) <http://iubio.bio.indiana.edu/soft/molbio/readseq/>.
  8. Fristensky, B., Lis, J. T., and Wu, R. (1982) Portable microcomputer software for nucleotide sequence analysis. *Nucl. Acids Res.* **10**, 6451–6463. <http://home.cc.umanitoba.ca/~psgendb/FSAP.html>.
  9. Fristensky, B. (1986) Improving the efficiency of dot-matrix similarity searches through use of an oligomer table. *Nucl. Acids Res.* **14**, 597–610.
  10. Pearson, W. R. (1990) Rapid and sensitive sequence comparison with FASTP and FASTA. *Meth. Enzymol.* **183**, 63–98. <ftp://ftp.virginia.edu/pub/fasta/>.
  11. NCBI. *Nentrez*. <http://www.ncbi.nlm.nih.gov/Entrez/Network/nentrez.overview.html>.
  12. NCBI. *Sequin*. <http://www.ncbi.nlm.nih.gov/Sequin/index.html>.
  13. NCBI. *Blast client*. <ftp://ncbi.nlm.nih.gov/blast/network/>.
  14. NCBI. *Cn3D*. <http://www.ncbi.nlm.nih.gov/Structure/cn3d.html>.
  15. Fristensky, B. (1993) Feature expressions: creating and manipulating sequence datasets. *Nucl. Acids Res.* **21**, 5997–6003. <http://home.cc.umanitoba.ca/~psgendb/XYLEM.html>.
  16. Fristensky, B. *Introductory Cytogenetics*, University of Manitoba. [http://www.umanitoba.ca/afs/plant\\_science/COURSES/CYTO/](http://www.umanitoba.ca/afs/plant_science/COURSES/CYTO/).
  17. Needleman, S. and Wunsch, C. D. (1970) A general method applicable to the search for similarities in the amino acid sequence of two proteins. *J. Mol. Biol.* **48**, 443–453.

