17 Management of a Server-Based Bioinformatics Resource

Brian Fristensky

Introduction

The strategies for managing a server-based molecular biology software resource, accessed by a diverse user community will be discussed. It assumes that the reader is familiar with basic UNIX commands and concepts. The approaches discussed here are implemented in the BIRCH system (*see* Website: http://home.cc.umanitoba.ca/~psgendb) but are generally applicable to any centralized multiuser software installation.

Most major UNIX distributions now come with graphic tools that simplify many administration tasks. It is therefore realistic to act as your own *sysadmin*. In fact many of the principles discussed are valid in the larger context of a general purpose multiuser system. Although general system administration is a broad field, particular attention should be paid to: daily and weekly backups, both onsite and offsite; security, including rapid installation of security patches; management of user accounts; and disk space to minimize the work and know-how needed on the part of the user. These topics are beyond the scope of this chapter, and are covered extensively in books on system administration, on USENET newsgroups in the comp.* section, and at various HOWTO websites.

The key factors and considerations when implementing the system are:

1. A user base with a diverse set of needs and usually minimal informatics training.
2. A diverse software base, comprised of programs from many authors, in many languages, and in many styles.
3. Documentation written in many formats and styles.
4. A complex networked server system.
5. Limitations of disk space and computing resources.

This chapter builds on the organizational scheme described in the previous chapter. To summarize, the resource is located in a world-readable directory tree referenced by the $DB environment variable. Program binaries, documentation, and ancillary datafiles are located in $DB/bin, $DB/doc, and $DB/dat, respectively. To use the resource, user accounts are set up by running the *newuser* script. This adds lines to their .login and .cshrc files, which read configuration commands from $DB/admin/login.source and $DB/admin/cshrc.source. The commands in these files are executed when a user logs in or starts a new shell. Thus, as the central configuration is updated,

all users have immediate access to the updates. The means to implement this structure are described in the previous chapter.

Managing Documentation

Documentation is the user's entry point into the system. Keeping documentation organized, accessible, and updated accomplishes several tasks. First, it helps to bring out difficulties that users may face in running programs. Second, it forces the Bioadmin to see the software base from the user's perspective. Third, well-organized documentation works to the Bioadmin's advantage, making it easy to refer users to the appropriate documentation, rather than having to answer the same question over and over. While installation of documentation should be straightforward, there are a few considerations for providing a consistent web-accessible documentation library.

HTML

HTML is rapidly becoming the most common format for documentation because of its dynamic capabilities and universal availability. However, it is probably best to keep a local copy of program documentation on your website, rather than simply linking to the author's website. An author's website will probably describe the most recent version of the software, which may not be installed on your system. As well, if the author stops supporting a software package, he or she may no longer keep documentation on a website. Thus making sure a local copy of the documentation that was obtained at the time the package was installed is guaranteed to accurately describe the version of the software currently installed.

UNIX Manual Pages

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 the 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. For display on the web, UNIX manual pages can be converted to ASCII text by redirecting output from the **man** command to an ASCII file, e.g.,

man fasta > fasta.asc

Postscript and PDF

Although PostScript viewers are usually available on most UNIX workstations, **acroread**, the Adobe Acrobat Reader, has been universally adopted. Therefore, it is probably safest to convert postscript files to PDF for web accessibility using **ps2pdf**, e.g., **ps2pdf primer3.ps**, will create a file called primer3.pdf. ps2pdf is included with most UNIX distributions.

ASCII Text

All web browsers can display ASCII text. It should be noted that file extensions such as .txt or .asc are probably best to use, becuase these are not commonly used by

application software. ASCII files with .doc extensions should be changed to some other extension to avoid confusion with Microsoft WORD files.

Word Processor Documents

Import filters are often less than satisfactory. Therefore, when documentation is in a format specific to a word processor such as WordPerfect, StarOffice Writer, Applix Words, or Microsoft Word, it is best to convert it to the PDF format. Some programs can directly save or print to PDF, while others can only print to PostScript. For the latter, convert to PDF using **ps2pdf** as noted earlier.

Communicating with the User Base

Login Messages

Brief announcements can be printed at the user's terminal by including in login.source a statement such as **cat ~psgendb/admin/Login_Message**, where Login_Message contains a few lines of text with the current announcements. This message is printed in each terminal window.

See companion CD for color Fig. 1

Web Site Organization

The BIRCH website provides a number of views to the system (*see* Fig. 1). The *New User* section provides documents that describe BIRCH, how to set up account, and how to learn the system. The *Documentation* section provides tutorials and other resources for users to develop their informatics skills while getting useful work done. Finally, the complete online documentation is available in the *Software* and *Database* sections, describing the full functionality of the system.

All login messages are archived in the file WHATSNEW.html, which can be viewed in a scrolling window entitled *BIRCH ANNOUNCEMENTS*. This file provides links to more detailed information than appears in login messages, so that even users who have been away from the system for a while won't miss important changes.

Discussion Groups

Although online discussions can be conducted through a mailing list, these often become an annoyance as the number of users increase and the number of lists one is subscribed to increases. Most web browsers such as Netscape and Internet Explorer/ Outlook Express, as well as third-party applications, can be used to read and participate in discussions on USENET newsgroups. Many users are familiar with world-wide groups, including the bionet.* groups (e.g., bionet.software, bionet.molbio. genearrays). However, it is also possible to have local newsgroups on any system that operates a newsserver, as do most campus UNIX systems. The local news Bioadmin can easily create a group such as *local.bioinformatics* or *local.genomics* that will be accessible to the local user community.

Remote Consultation Using VNC

Remote consultation on UNIX platforms is now greatly enhanced by Virtual Network Computing (VNC). VNC is a package of programs freely distributed by AT&T (*see* Website: http://www.uk.research.att.com/vnc/). In essence, vncserver creates an X11 desktop session on a remote login host, which keeps an image of the screen in

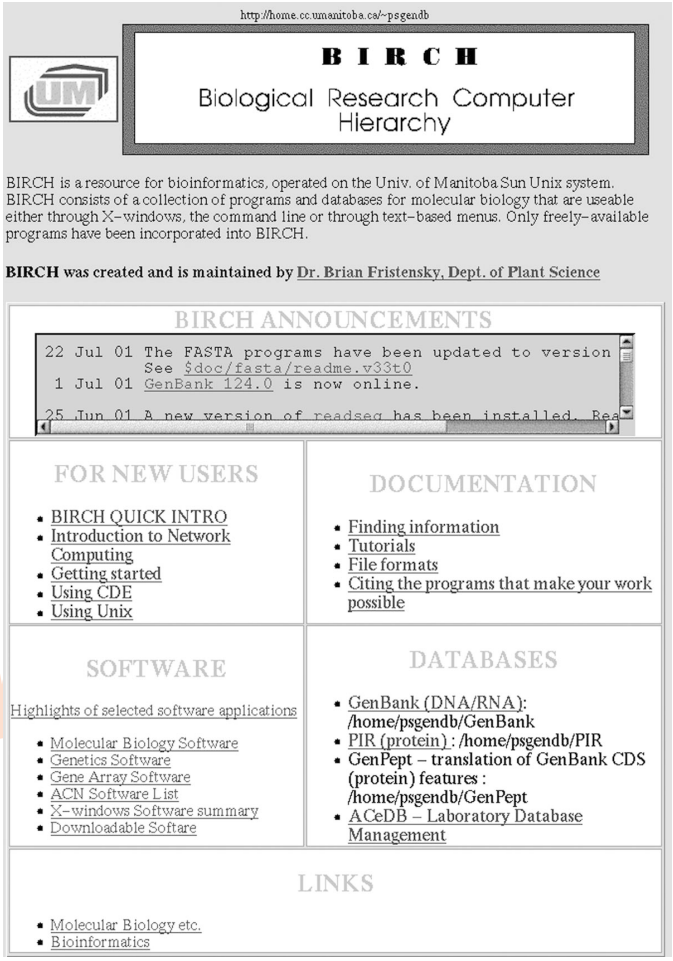

Fig. 1. Organization of web-based documentation on the BIRCH home page (*see* Website: http://home.cc.umanitoba.ca/~psgendb).

See companion CD for color Fig. 2

memory. A copy of vncviewer, running on a PC or workstation anywhere in the world with a high-speed Internet connection, can display and control the screen as if it were running locally. Figure 2 shows a screen in which a *vncviewer* window is displayed. VNC is available for MS-Windows, Macintosh, and UNIX. The vncviewer can also run as a Java applet in a web browser, so that vncviewer does not have to be installed on the local machine. Thus, regardless of where you are, your UNIX desktop looks and acts the same.

For remote consultation, assume that a user has phoned the Bioadmin with a problem. If it cannot be easily described over the phone, the user changes their VNC password using **vncpw**, and tells the Bioadmin the new password. Next the user starts up a copy of vncserver:

vncserver -alwaysshared
New 'X' desktop is mira:8

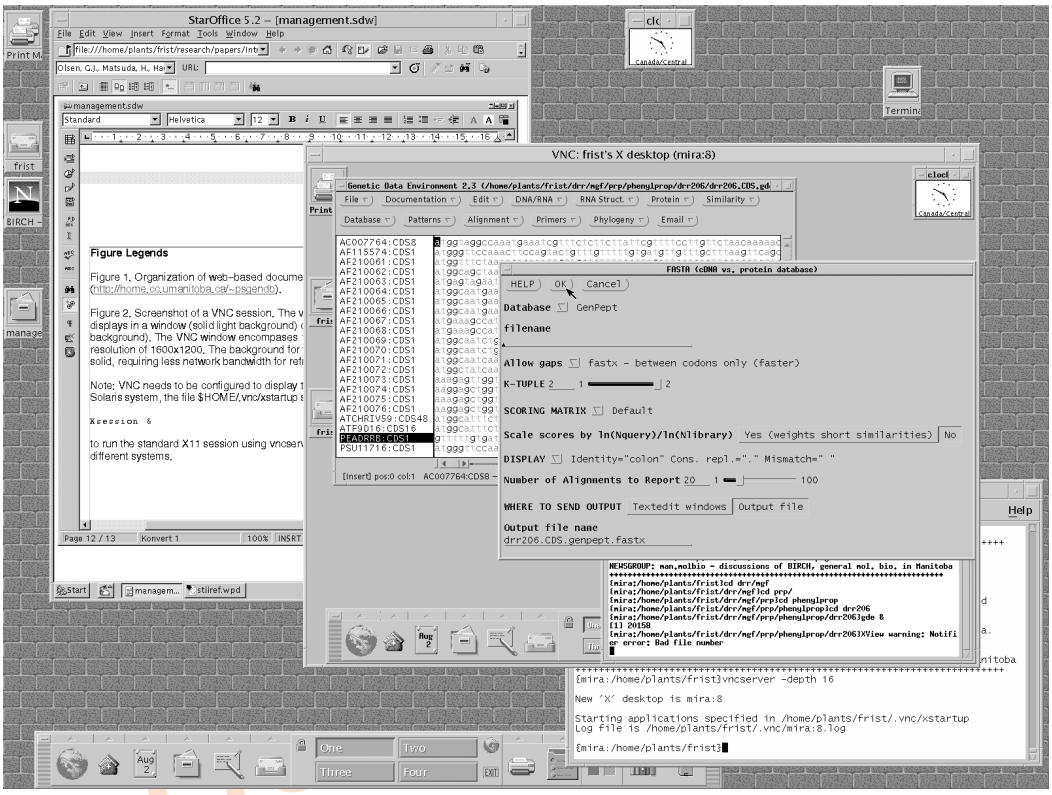

Fig. 2. Screenshot of a VNC session. The vncviewer VNC: frist's X desktop (mira:8) displays in a window (solid light background) on the local desktop (dark brick background). The VNC window encompasses 1024×768 pixels, against a screen at a resolution of 1600×1200. The background for the mira:8 desktop has been changed to solid, requiring less network bandwidth for refreshing the screen across a network.

Note: VNC needs to be configured to display the user's regular X11 desktop. On our Solaris system, the file $HOME/.vnc/xstartup should contain the line Xsession & to run the standard X11 session using vncserver. For the GNOME desktop, this line would read gnome-session &.

The *-alwaysshared* option makes it possible for more than one user to simultaneously display the same desktop. The message tells the user that vncserver has created desktop 8 on host mira.

Next, the user and Bioadmin each type: **vncviewer mira:8** followed by the password, and the same vncviewer window will appear on both of their screens (*see* Fig. 2). If connecting via a browser, vncviewer would be launched for this screen by setting the URL to http://mira.cc.umanitoba.ca:5808, where the last two digits in 5808 indicate the screen number.

Now, both the user and Bioadmin can see and control the same desktop while discussing the various operations over the phone. The user can run a program that is causing difficulty, and the Bioadmin can see everything that happens. The Bioadmin can demonstrate in real time what the user should be doing, and if necessary, datafiles or configuration files such as .cshrc can be examined.

At the end of the session, the Bioadmin reminds the user to kill the VNC session by typing **vncserver -kill :8** and to change their VNC password.

The real value of VNC becomes apparent when traveling. For example, applications such as Powerpoint produce static presentations, and most people travel with their own laptop to ensure that it will work. Over the last 2 years, I have given real-time presentations across North America using VNC on any computer at hand. As long as a fast Internet connection is available, the full functionality of the desktop can be demonstrated anywhere where there is a computer and a data projector.

Detecting, Handling and Preventing Problems

A multiuser system poses challenges in terms of managing shared resources, such as CPU time, memory, disk space, and network bandwidth. Usually it is possible to design a system that will minimize user errors, and in most cases UNIX is intrinsically protected from most catastrophes. For example, unless permissions are explicitly set otherwise, a user can only read or modify files belonging to him, and usually these can only reside in the $HOME directory.

Disk Space

$HOME directories should always reside in a separate file system, and user quotas should be set, regardless of how much disk space exists. The one filesystem that is potentially troublesome is /tmp, which is writeable by all users. In the event that /tmp becomes full, programs that need to write temporary files may hang, resulting in a *filesystem full* error. The best way to avoid this problem is to have applications write temporary files to the current working directory, so that in the worst case, only the user is affected.

CPU Time

Monitoring CPU Usage

Keeping track of CPU usage is critically important. The **top** command gives you a real-time picture of the most CPU intensive jobs currently running on the host you are logged into. If you type **top** at the command line, your system will generate similar information to the following:

```
last pid: 13912; load averages: 2.61, 1.64, 1.31 13:48:41
504 processes: 488 sleeping, 1 running, 6 zombie, 7 stopped, 2 on cpu
CPU states: 16.4% idle, 65.7% user, 17.9% kernel, 0.0% iowait, 0.0% swap
Memory: 640M real, 17M free, 846M swap in use, 3407M swap free

  PID USERNAME THR PRI NICE  SIZE   RES STATE    TIME    CPU COMMAND
11371 umamyks   13  10    0   77M   71M cpu/0   23:19 64.27% matlab
27668 frist     10  58    0   97M   56M sleep    3:06  3.49% soffice.bin
13894 frist      1  33    0 3344K 1672K cpu/1    0:01  1.65% top
13898 umnorthv   1  58    0 6424K 4352K sleep    0:00  0.82% pine.exe
 1629 mills      7   0    0 9992K 7840K sleep    0:24  0.42% dtwm
13704 mhbasri    1  38    0 1464K 1360K sleep    0:01  0.31% elm.exe
 9797 syeung     1  58    0 1000K  816K sleep  267:53  0.28% newmail
 6914 umtirzit   8  58    0   13M 3992K sleep   26:38  0.23% dtmail
26524 mgarlich   1  58    0 9376K 6960K sleep    0:10  0.23% dtterm
29993 simosko    1  58    0 6824K 4528K sleep    0:21  0.23% pine.exe
 7937 jayasin    1  58    0 6112K 3816K sleep    6:55  0.22% Xvnc
 4483 francey    7  48    0 9904K 7920K sleep    0:24  0.21% dtwm
  206 root       6  58    0   76M 8024K sleep  458:13  0.20% automountd
27272 frist      7  58    0   11M 8376K sleep    0:35  0.20% dtwm
  580 syeung     1  48    0 2376K 1976K sleep    2:56  0.19% irc-2.6
```

This display is updated every few seconds in the terminal window. To quit, type **q**. The top command has many options. For example, you can sort jobs by memory used, or list only jobs under a given userid. The owner of a job can also kill that job using top. Type **man top** for full documentation.

The **ps** command with no arguments tells which jobs are running in the current shell (the current window):

```
ps
  PID TTY      TIME CMD
 2122 pts/104 0:00 dsdm
27401 pts/104 2:18 mozilla-
27376 pts/104 0:00 netscape
 2082 pts/104 0:00 zwgc
27384 pts/104 0:00 netscape
27396 pts/104 0:00 run-mozi
 2024 pts/104 0:18 Xvnc
 2041 pts/104 0:00 Xsession
27305 pts/104 0:01 csh
27381 pts/104 0:00 netscape
27457 pts/104 0:14 java_vm
```

while

ps -u userid

tells which jobs are running under a given userid on the host you are logged into.

The following list summarizes the types of tasks that tend to require a lot of processing time:

Jobs That Tend to be CPU-Intensive:

1. Phylogenetic Analysis
   a. Distance matrix methods (e.g., Neighbor Joining, FITCH): Usually require negligible time, the amount of time increases in a linear fashion with the number of sequences.
   b. Parsimony (e.g., DNAPARS, PROTPARS): Moderately efficient, the amount of time increases exponentially with the number of sequences
   c. Maximum likelihood (e.g., DNAML, PROTML, fastDNAML): Very slow, the amount of time increases according to a FACTORIAL function of the number of sequences.
2. Sequence database searches: The amount of time that is required is proportional to product of sequence length and database size; use high **k** values to speed up search; protein searches faster than DNA.
3. Multiple sequence alignments (e.g., CLUSTAL): Cluster type alignments scale linearly in proportion to the number of sequences.
4. Retrievals of large numbers of sequences: The time required is linear, related to number of sequences.
5. The efficiency of any sorting operation with a large number of items depends on the sort algorithm used.
6. Statistical and mathematical packages (e.g., SAS, MATLAB).

Jobs That Should Never Be CPU Intensive

If the following applications are using significant percentages of CPU time, they are not functioning normally, and are probably *runaway jobs*.

1. Graphic front ends: Programs such as GDE, SeqLab, or SeqPup by themselves do almost nothing. If you see one using a substantial amount of CPU time, it is probably a *runaway job*. One exception is when reading large sequence files, e.g., large numbers of sequences or very long sequences that are placed in memory for analysis.
2. Most user apps (e.g., word processors, mailers, spread sheets, drawing programs).
3. Desktop tools (e.g., text editors, filemanagers).
4. Most UNIX commands.
5. Web browsers: For short bursts browsers can be very CPU-intensive, but this should not persist for more than a minute or two.

Managing Long-Running Jobs

On any multitasking system, all jobs are assigned priorities that govern the amount of CPU time allocated to them. In UNIX, the nice command determines the priority. Most user commands default to a nice value of 0. This is especially important for applications run through a graphic interface, which need to work in real time. The higher the nice value, the less CPU time a job will be allocated, and the less of a load it puts on the system. Programs known to be CPU intensive can therefore be set to run at low priority. A higher nice value prevents the program from taking large amounts of time at high loads, but does not prevent it from utilizing CPU resources when the load on the system is light.

CPU-intensive tasks such as database searches or phylogenetic analysis should be run from wrappers, i.e., scripts that set parameters before running the program. The name of the program is preceded by the **nice** command. For example, to run fastDNAml at the default priority, a wrapper might contain the line:

nice fastDNAml arguments... &

The default priority for **nice** varies from system to system. In the BIRCH system, most sequence programs are launched from the GDE interface by calling wrappers that run programs using nice. As well, termination of the command line with an ampersand (&) tells the shell to run the task in the background. Thus, a user can launch a long-running job, quit GDE, and logout without terminating the job. When the program is completed, the output is written to the file, which the user can access when logging in for the next session.

In some cases, programs that use a graphic interface will perform analyses that require very long execution times. The problem with this design is that the user must remain logged in to the terminal from which the program was launched, because quitting the program would terminate the analysis. One can circumvent this problem by running jobs of this type from a vncviewer window. Killing a vncviewer window has no effect on the applications currently running, and the user can open up the same screen at any time from anywhere. This has the added benefit of making it easy to remotely monitor the progress of long-running jobs.

Killing Runaway Jobs

Sometimes a program will not correctly handle an error, and will begin using up large amounts of CPU time. Unless the Bioadmin has root permissions, it is neces-

sary to ask either the owner of the job or a sysadmin to kill it. For example, a runaway netscape job might show up thus when running the top command:

```
  PID USERNAME THR PRI NICE   SIZE    RES STATE     TIME    CPU    COMMAND
25779 frist     13  22    0   24M   11M cpu/0   23:19 64.27% netscape
```

To kill the job, root or the owner would type: **kill -9 25779**.

Applications that do not normally use a lot of CPU time, but may be prone to runaway execution, could be contained by running them from a wrapper, in which the **ulimit** command is issued prior to running the program, e.g., **ulimit -t 900**, limiting CPU time in the current shell to 900 seconds (15 min).

Acknowledgments

Thanks to the Academic Computing and Networking staff at the University of Manitoba for UNIX system support. This work was made possible in part through hardware provided by the Sun Academic Equipment Grants Program.

Suggested Readings

Introduction

Fristensky, B. (1999) Building a multiuser sequence analysis facility using freeware, in: Bioinformatics Methods and Protocols, (Misener, S. and Krawetz, S., eds.), Humana Press, Totowa, NJ, pp. 131–145.

Fristensky, B. (1999) Network computing: Restructuring how scientists use computers and what we get out of them, in: Bioinformatics Methods and Protocols, (Misener, S. and Krawetz, S., eds.), Humana Press, Totowa, NJ, pp. 401–412.

Sobell, M. G. (1995) A Practical Guide to the UNIX System, Addison-Wesley Publishing, Reading, MA.

< Au: location correct?

Detecting, Handling and Preventing Problems

Felsenstein, J. (1989) PHYLIP Phylogeny Inference Package, Cladistics 5, 164–166.

Olsen, G. J., Matsuda, H., Hagstrom, R., and Overbeek, R. (1994) FastDNAml: a tool for construction of phylogenetic trees of DNA sequences using maximum likelihood, Comp. Appl. Biosci. 10, 41–48.

Smith, S., Overbeek, R., Woese, C. R., Gilbert, W., and Gillevet, P. M. (1994) The Genetic Data Environment: an expandable GUI for multiple sequence analysis, Comp. Appl. Biosci. 10, 671–675.

(*see* Website: http://megasun.bch.umontreal.ca/pub/gde/)

Thompson, J. D., Gibson, T. J., Plewniak, F., Jeanmougin, F., and Higgins, D. G. (1997) The CLUSTAL X windows interface: flexible strategies for multiple sequence alignment aided by quality analysis tools, Nucleic Acids Res. 25, 4876–4882.

Au: Please provide "Glossary"