Netlib Services and Resources

Shirley Browne, Jack Dongarra, Eric Grosse, Stan Green,
Keith Moore, Tom Rowan, and Reed Wade *

Abstract

The Netlib repository contains freely available software, documents, and databases of interest to the numerical, scientific computing, and other communities. The repository is maintained by AT&T Bell Laboratories, the University of Tennessee and Oak Ridge National Laboratory, and by colleagues world-wide. This report includes both the Netlib User's Guide and the Netlib System Manager's Guide, and contains information about Netlib's databases, interfaces, and system implementation. The Netlib repository's databases include the Performance Database, the Conferences Database, and numerous bibliographic and address databases. A variety of user interfaces enable users to access the Netlib repository in the manner most convenient and compatible with their networking capabilities. These interfaces include the Netlib email interface, the Xnetlib X Windows client, the netlibget command-line TCP/IP client, anonymous FTP, anonymous RCP, gopher, and the World Wide Web. Those without networks can get Netlib on CD-ROM. The collection is replicated at several sites around the world, automatically synchronized, to provide reliable and network efficient service to the global community.

*Work on Netlib is supported by National Science Foundation under Grant. No. ASC-9103853 and by AT&T Bell Laboratories. Work on Xnetlib is supported by the Advanced Research Projects Agency under contract DAAL03-92-G-0284 administered by the Army Research Office.

Contents

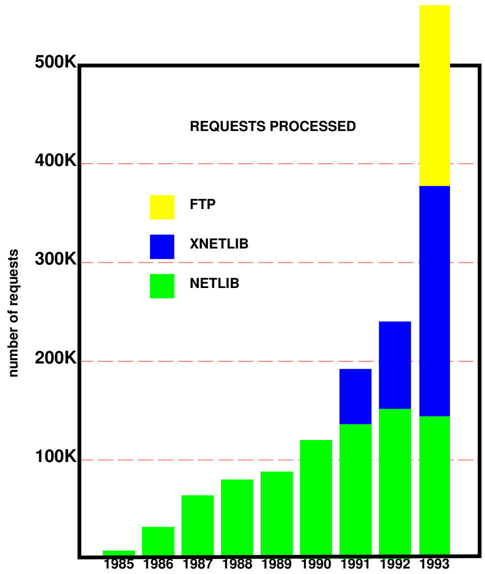

Figure 1: Netlib Requests

Introduction

Netlib began services in 1985 to fill a need for cost-effective, timely distribution of high-quality mathematical software to the research community. Netlib sends, by return electronic mail, requested routines together with subsidiary routines and any requested documents or test programs supplied by the software authors [5]. Xnetlib, a recently developed interactive tool for software and document distribution [4], use an X Window interface and TCP/IP connections to allow users to receive replies to their requests within a matter of seconds. The interface provides a number of modes and searching mechanisms to facilitate searching through a large distributed collection of software and documents. World Wide Web browsers such as Mosaic can also be used to access Netlib via HTTP and FTP, but are not as finely tuned towards software retrieval as Xnetlib. The `netlibget` command-line interface and anonymous FTP and RCP provide services to users who do not need a sophisticated interface. Figure 1 shows the growing number of requests for Netlib services.

A new hypertext/hypermedia version of Xnetlib, currently under development, will interoperate with other information services such as `gopher`, `WAIS`, and `World Wide Web`. It will incorporate a new type of executable document, called an *active object*, that will greatly enhance the flexibility and adaptability of Xnetlib by allowing runtime binding of functionality.

Although the original focus of the Netlib repository was on mathematical software, the collection has grown to include other software (such as networking tools and tools for visualization of multiprocessor performance data), technical reports and papers, a Whitepages Database, benchmark performance data, and information about conferences and meetings. The number of Netlib servers has grown from the original two, at Oak Ridge National Laboratory (initially at Argonne National Laboratory) and Bell Labs, to servers in Norway, the United Kingdom, Germany, Australia, Japan, and Taiwan. A mirroring mechanism keeps the repository contents at the different sites consistent on a daily basis, as well as automatically picking up new material from distributed editorial sites[8]. This mechanism provides redundancy in case of computer or network failures, shares the workload, and broadens human contacts for identifying software to add to the collection.

Netlib differs from other publicly available software distribution systems, such as `Archie`, in that the collection is moderated by an editorial board and the software contained in it is widely recognized to be of high quality. The user is assured of getting an up-to-date copy of the master version of the requested software. We log requests so we can send bug reports and updates to users of our software. However, the Netlib repository is not intended to replace commercial software. Commercial software companies provide value-added services in the form of support. Although the Netlib collection is moderated, its software comes with no guarantee of reliability or support. Rather, the lack of bureaucratic, legal, and financial impediments encourages researchers to submit their codes by ensuring that their work will be made available quickly to a wide audience.

Requests for consideration of software and document submissions to Netlib, as well as questions, comments, and problems concerning Netlib, should be sent to the following address:

`netlib_maintainers@netlib.org`

Netlib User's Guide

1 Repository Contents

1.1 Software and Documents

The Netlib repository contains a large collection of high-quality public-domain mathematical software. In addition, the repository contains other material of interest to the scientific computing community, including software documentation, test data, technical papers, and reports. Most of the software is written in Fortran, but programs in other languages, such as C and C++, are also available. Netlib supports dependency checking for Fortran software, so that all of the routines a particular Fortran routine depends on (i.e., calls) can be retrieved with it.

The software is organized as a Unix directory tree. The subdirectories under the Netlib root are called *libraries*. Each library contains an index file describing the library contents, namely the files for the individual routines and any subdirectories the library may have. Software routines from a variety of sources are currently available from Netlib. Some of the libraries Netlib distributes – such as EISPACK, LINPACK, FFTPACK, LAPACK, algorithms from the ACM *Transactions on Mathematical Software*, and algorithms from the book by Forsythe, Malcolm, and Moler [7] – have long been used as important tools in scientific computation and are widely recognized to be of high quality. The Netlib collection also includes a large number of newer, less well-established codes.

To submit software or documents for inclusion in the Netlib repository, follow the guidelines in `/netlib/misc/contrib`, which you can obtain either by email as follows:

```
mail netlib@netlib.org
send contrib from misc
```

or by opening

```
ftp://netlib.att.com/netlib/misc/contrib.Z
```

using your favorite World Wide Web browser, such as Mosaic or Lynx, or by downloading the `contrib` file from the `misc` library using the Xnetlib client. Send your submission to the following address:

```
netlib_maintainers@netlib.org
```

or, even better, send directly to the editor listed in the relevant Netlib index file covering your area.

1.2 Performance Database

The Performance Database is a publicly-accessible central repository of performance data for all ranges of machines, from personal computers to supercomputers. It provides an on-line catalog of public-domain computer benchmarks such as the LINPACK Benchmark, Perfect Benchmarks, and the NAS Parallel Benchmarks. The benchmark data are presented void of any subjective interpretations of machine performance. The Performance Database allows all branches of the computing community to archive performance metrics and makes them readily available to the public. For further details, see [1].

The performance data are stored in the `performance` directory in the Netlib repository. Although it is possible to download performance data using anonymous FTP or RCP, the `Performance` button on the Xnetlib X Windows client provides browsing and keyword searching mechanisms, as well as formatted display of the data. The Performance Database may also be accessed by opening

```
http://performance.netlib.org/performance/html/PDStop.html
```

using a forms-capable World Wide Web browser.

1.3 Conferences Database

The Conferences Database contains conference and meeting announcements. Although it is possible to download conference descriptions using anonymous FTP or anonymous RCP, use of the `Conferences` button on the Xnetlib client, or opening

```
http://netlib2.cs.utk.edu/confdb/Conferences.html
```

using a forms-capable World Wide Web browser, allows searching separately by title, dates, location, or description keywords, as well as viewing of the results in a window. Conferences may be submitted from the Xnetlib client or from a forms-capable World Wide Web browser. The contents of the title, location, and description fields are added to a full-text index when a conference is entered into the database. A geographical database handles location name aliases and hierarchical geographical relationships.

1.4 Address and bibliography databases

NA-NET is a community of numerical analysts and other researchers who communicate through a common email facility. NA-NET is supported by a mail-forwarding database and a Whitepages Database. The mail-forwarding database gives users an easy method of communicating with each other through the use of a uniform email address. This feature avoids the problem of having to remember an individual's specific email address. Mail to an NA-NET member can be addressed to na.<key>@na-net.ornl.gov, where <key> is the member's NA-NET key. The key is usually the member's last name, possibly prefixed by the first letter of the first name. The mail-forwarding database also serves as the distribution list for the NA-NET News Digest. To join or use NA-NET, use the email interface to NA-NET. See Section 2.2.3 for further details.

The NA-NET Whitepages is a directory service that allows NA-NET members to find out more information about other members. Anyone can join the Whitepages, but NA-NET members are especially encouraged to join. The Whitepages can be accessed by means of the NA-NET Whitepages email interface, the netlibget command-line TCP/IP client, or the Who button on the Xnetlib client. Currently, the Whitepages can be joined only through the email interface. See Section 2.2.4 for further details.

The SIAM Membership Directory is a separate directory that is copyrighted by the Society for Industrial and Applied Mathematics. Queries to the NA-NET whois service that do not find an entry in the NA-NET Whitepages database are referred to the SIAM list. Although the membershp list itself is secret, you may download an encrypted form of it, along with software that allows you to look up individual entries keyed by last name or phone number[6]. This allows some attractive caller-id features if your phone is connected to your computer.

Another service hosted by Netlib is the Bibnet project led by Stefano Foresti and Nelson H. F. Beebe of the University of Utah. This project will make available a bibliographic database in the field of scientific computing, populated by user contributions. Once this project is well underway, it will become easy to keep paper references up to date, even as preprints turn into published papers, and to locate online versions of published papers.

1.5 Metacontents

Netlib should be not just a warehouse but a library, and for that it must have adequate search tools. Each chapter (library) of netlib comes with an "index" file in a specified format, to promote searchability. Full details of the format can be found in Section 4.1 or ftp://netlib.att.com/netlib/bib/thesaurus. Briefly, each index file describes the content of a library by one "paragraph" of attribute–value pairs for each file or sub-library. Some of the more important attributes are: *for*, which tersely describes what problem the tool solves; *alg*, which indicates distinctive features of the algorithm used; *by*, which lists the authors; *gams*, which allows searching by category; *size*, which is useful when accessing netlib over limited network connections. Each library has its own index file, maintained by the author or editor. Every night all the index files are merged together into a global database to facilitate global keyword searching.

A second kind of index is found in netlib/crc/. Here can be found a list of all files in the collection, with dates and checksums. Together with the software provided in that directory, the relatively small crc file allows one to automatically check whether local copies of files are current.

1.6 Remote Execution

On a limited demonstration basis, Netlib also provides trial use of software in the collection. The principal example at present is `f2c`, a Fortran-to-C compiler developed by Stu Feldman, David Gay, Mark Maimone, and Norm Schryer. This program is more complicated to install on one's own machine than, say, just loading in a typical subroutine from `linpack`. To help people judge whether they want to bother, Netlib allows them to send sample Fortran input by email and get back the converted C by return mail. The result are more or less guaranteed correct but hard to read. If that is acceptable, the user is then encouraged to download the `f2c` compiler and runtime libraries and use the software directly.

2 User Interfaces

2.1 World Wide Web

If you are a user of the World Wide Web, for example using a browser such as Mosaic, the most convenient way to use netlib will probably be to open `http://www.netlib.org/` or `ftp://netlib.att.com/home.html`. The two sites present a slightly different face, as we explore different ways to present the collection, and they differ in a few libraries such as local technical reports, but otherwise the contents are identical and the servers may be used interchangably.

Because netlib's WWW interface is relatively new (having started operation in October 1993), some of the more specialized services such as `f2c` have not be added yet. Progress on this front is quite rapid, and as the system is relatively self-explanatory, we say no more about it here.

2.2 Email

Anyone with an email connection to the Internet can access most of the Netlib repository. There are email interfaces to the software and document libraries, and the NA-NET mail-forwarding and Whitepages Databases. To receive more information about the email interface to Netlib, send a message to `netlib@netlib.org` with the message body `send index`. To receive more information about the email interface to NA-NET, send a message to `na.help@na-net.ornl.gov`.

2.2.1 Support Addresses

There are email support addresses for the different Netlib services. Users with comments, questions, or bug reports should send a message to the appropriate support address listed below.

Netlib repository	netlib_maintainers@netlib.org
Xnetlib client	xnetlib@cs.utk.edu
Performance database	utpds@cs.utk.edu
Conferences database	conferences@cs.utk.edu
NA-NET	nanet@na-net.ornl.gov
Gopher server	gopher@netlib.org

2.2.2 Email Interface to the Netlib Repository

Netlib email addresses

The Internet address `netlib@netlib.org` refers to a gateway machine at Oak Ridge National Laboratory in Oak Ridge, Tennessee. This address should be understood on all the major networks through the normal Domain Name System name resolution. If for some reason that machine is not responding, try `netlib@research.att.com`, a machine at Bell Labs, Murray Hill, New Jersey.

For access outside the United States, you may want to use one of the repositories that mirror the UT/ORNL repository. See Appendix A for a list of these other Netlib sites.

Request syntax

A valid Netlib email request has a message body that is of one of the following basic forms:

```
send index
send index from <library>
send <file(s)> from <library>
find <keywords>
whois <name>
mailsize <size>
```

Here are examples of the various kinds of requests.

* To get the master index for netlib:

  `send index`

* To get the full index for a library:

  `send index from eispack`

* To get a particular routine and all it depends on:

  `send dgeco from linpack`

* To get just the one routine, not subsidiaries:

  `send only dgeco from linpack`

* To get dependency tree, but excluding a subtree:
  `send dgeco but not dgefa from linpack`
* To just tell how large a reply would be, don't actually send the file:
  `send list of dgeco from linpack`
* To search for somebody in the SIAM membership list:
  `whois gene golub`
* To do a keyword search for Netlib software:
  `find cubic spline`
* To do a bibliographic search:
  `find schumaker from approximation`
  `find aasen from linalg`
* To convert Fortran to C: `execute f2c  print *,'hello, world'  end`
* To set the chunk size used for reply:
  `mailsize 100k`
* (optional) End of request:
  `quit`

2.2.3 Email Interface to NA-NET

Individual (unicast) messages

Sending email to an individual NA-NET member is the most frequently used feature of NA-NET. Each NA-NET member has a unique NA-NET name, or key. Mail can be sent to an NA-NET member by addressing it to `na.<key>@na-net.ornl.gov`, where `<key>` is the member's NA-NET name. The NA-NET name is usually the member's first initial prepended to her last name, the member's last name, or the member's first name followed by the first letter of her last name. For example, possible NA-NET names for Joan Smith would be `jsmith`, `smith`, and `joans`.

NA-NET News Digest

Any mail sent to `na.digest@na-net.ornl.gov` will be considered for distribution to all members of NA-NET. Once a week, we send out a digest of information contributed by users of NA-NET. The editor of the NA-NET News Digest goes over the messages that have been received, picks out the ones thought to be of general interest to the numerical analysis community, combines them in the News Digest format, and mails the Digest to everyone on the mailing list.

Joining NA-NET

To join NA-NET , send mail to `na.join@na-net.ornl.gov`. In the message body, specify the following three fields:

```
Lastname: <your last name>
Firstname: <your first name>
E-mail: <your e-mail address>
```

The values can be specified in any order. The subject line of your message will be ignored. An attempt will be made to assign to you a unique NA-NET name consisting of your first initial prepended to your last name, your last name, or your first name followed by the first letter of your last name. If at least one of these keys is not already in use, you will receive a message indicating that your join attempt succeeded and telling you which key has been assigned. If all three of these keys fail to be unique, you will receive an error message indicating that your join attempt failed. In case of failure, send a message to `nanet@na-net.ornl.gov`, and you will be assigned a unique key manually.

Removing membership

To remove your membership from NA-NET, send mail to `na.remove@na-net.ornl.gov`. In the message body, specify the following two fields:

```
Lastname: <your last name>
Firstname: <your first name>
```

The values can be specified in any order. The subject line of your message will be ignored. NA-NET will send an acknowledgment message to both the deleted address and the address making the request. If more than one entry exists with the same first and last name, you will receive a message indicating that your removal attempt failed. In this case, you can resubmit the removal request with the additional line:

```
Key: <your NA-NET key>
```

Changing your email address

To change your email address in the NA-NET mail-forwarding database, send mail to `na.change@na-net.ornl.gov`. In the message body, specify the following three fields:

```
Lastname: <your last name>
Firstname: <your first name>
New-address: <your new e-mail address>
```

The values can be specified in any order. The subject line of your message will be ignored. An acknowledgment message will be sent to both the old email address as well as the new address informing you that the change has taken place.

If more than one entry exists with the same first and last name, you will receive a message indicating that your change attempt failed. In this case, you can resubmit the change request with the additional line:

```
Key: <your NA-NET key>
```

Help with NA-NET

Questions and comments about NA-NET should be addressed to `nanet@na-net.ornl.gov`. Mail sent to `na.help@na-net.ornl.gov` will receive a reply message describing both NA-NET and the Whitepages.

Current member list

Mail sent to `na.sendlist@na-net.ornl.gov` will receive a reply message being sent back to you containing the email addresses of all members of NA-NET.

2.2.4 Email Interface to the NA-NET Whitepages

Querying the Whitepages Database

To find out information about a person, send mail to `na.whois@na-net.ornl.gov`. In the message body or on the subject line specify the person's first name and last name, or just the last name. The order of first name and last name does not matter. For example, to find out more about Jack Dongarra:

```
mail to: na.whois@na-net.ornl.gov
Subject:

Jack Dongarra
```

or

```
mail to: na.whois@na-net.ornl.gov
Subject: jack dongarra

<null body>
```

Keyword searching is also possible. For example, to find out more information about all people who are interested in *parallel*, send the following message:

```
mail to: na.whois@na-net.ornl.gov
Subject:

Keyword: parallel
```

This query does a string search for the pattern *parallel* on all fields other than the name fields.

As another example, to find out more about all people who live in Knoxville:

```
mail to: na.whois@na-net.ornl.gov
Subject:

Keyword: Knoxville
```

This query does a string search on all fields other than the name fields for the pattern Knoxville.

Joining the Whitepages

To join the NA-NET Whitepages, send mail to `na.join-wp@na-net.ornl.gov`. In the message body, specify the two mandatory fields and as many of the optional fields as you want.

```
Mandatory
---------
Last_name: <your last name>
First_name: <your first name>

Optional
--------
Middle_name:
Other_name:
Affiliation:
Office_address:
City_state_zip:
Country:
Office_phone:
Research:
Home_address:
Home_phone:
Fax:
E_mail_address:
Other:
```

The fields can be specified in any order. The subject line of your message will be ignored. All fields are entered into the database as characters, so spaces can be used for readability. All fields except first_name, last_name, and middle_name can be multiple lines. A multiple-line field ends when the next keyword (e.g., "Country:") is encountered. Each line should end with a carriage return. If your first_name and last_name combination is not unique, send mail to `nanet@na-net.ornl.gov`, and your name will be manually inserted into the Whitepages Database in spite of the duplication. Such duplication will not cause any problems for people querying the Whitepages Database because the database is set up to return information on all people with a given last name, first_name, or combination. An acknowledgment to your join request will be sent back to you confirming that the operation was successful.

Removing your Whitepages entry

To remove your entry from the NA-NET Whitepages Database, send mail to `na.remove-wp@na-net.ornl.gov`. In the message body, specify the following two fields:

```
Last_name: <your last name>
First_name: <your first name>
```

The values can be specified in any order. The subject line of your message will be ignored. An acknowledgment message will be sent to both the address requesting the removal and to the address listed in the Whitepages Database.

Changing fields

To change the value of a field, to add a field, or to delete a field, send mail to `na.change-wp@na-net.ornl.gov`. In the message body, specify the following two fields:

```
Last_name: <your last name>
First_name: <your first name>
```

plus the fields to be added, changed, or dropped. The fields can be specified in any order. The subject line of your message will be ignored. You can not change your name. If you need to change your name, first remove your entry and then rejoin with the newname. If you wish to clear the value of a field, simply include the field with no value. An acknowledgment message will be sent back to you confirming that the operation was successful.

2.3 X Windows Interface – the Xnetlib Client

Xnetlib is an X Window System application that provides interactive file access and database query processing from multiple servers through TCP/IP connections. Xnetlib currently provides access to the Netlib software and document repository, the NA-NET Whitepages Database, the Performance Database, and the Conferences Database. Future releases of Xnetlib will provide additional features, such as access to remote execution facilities and interoperability with other information services, such as `gopher`, `WAIS`, and World Wide Web.

2.3.1 Acquiring the Xnetlib Software

To acquire the software for the Xnetlib client send email to `netlib@netlib.org` with the line

```
send xnetlib.shar from netlib
```

as the body of the message. Netlib will return the file `xnetlib.shar` by email.

Xnetlib is available by anonymous FTP from `netlib2.cs.utk.edu` in the `xnetlib` directory. Both executables and source are available.

To use anonymous FTP to retrieve an executable file, type

```
ftp netlib2.cs.utk.edu
anonymous
<your email address>
cd xnetlib
binary
get xnetlib.<arch>.Z
bye
```

where <arch> is your machine architecture (`alpha`, `hp9000`, `next`, `pmax`, `rs6000`, or `sun4`). To use anonymous FTP to retrieve the Xnetlib source, type

```
ftp netlib2.cs.utk.edu
anonymous
your email address
cd xnetlib
binary
get xnetlib.shar.Z
bye
```

assuming your system has the imake facility. Then move the executable file, named `xnetlib` to where you want it. See Section 2.3.3 for more information about building Xnetlib.

If you retrieve an executable, you may also wish to retrieve the shar file `xnetlib3.4.doc.shar`, which contains the Xnetlib man page and quick reference card. These documents are already included in the source code shar file.

2.3.2 System Requirements

Xnetlib can be built on nearly any Unix system. It runs under the X Window System, version 11, from MIT. It requires release level 4 or greater and the Athena widget libraries as supplied by MIT.

2.3.3 Building Xnetlib

If you retrieved an executable (e.g., `xnetlib.sun4`), install it by uncompressing it and changing the mode to executable, e.g.,

```
uncompress xnetlib.sun4.Z
chmod 755 xnetlib.sun4
```

Xnetlib makes use of the imake facility (via `xmkmf`) that comes with standard X Windows distributions. A generic Makefile is provided and can be used if imake is not present on your system.

If you retrieved a compressed shar file of the Xnetlib source code, extract the files and build Xnetlib by typing

```
uncompress xnetlib.shar.Z
sh xnetlib.shar
cd xnetlib3.4/src
xmkmf
make
```

After the executable is built, install it by copying the file `xnetlib` to an appropriate directory. There is no application defaults file to install.

```
With imake} (xmkmf)

      Type:
         xmkmf
         make

Without imake (xmkmf)

    First edit Makefile.std to reflect your system characteristics.
    Then type:

         make -f Makefile.std
```

`xmkmf` should be installed on your system as part of the X distribution. If you get an error on the `xmkmf` command, check your command search path or talk to your system manager. If you are using an `IBM/RS6000`, you may need to refer to the information in `xnetlib3.4/doc/README.AIXv3`.

For further information on system-wide installation of the Xnetlib client and on customization of the Xnetlib client, see the section in the Netlib Manager's Guide on the Xnetlib client (Section 7).

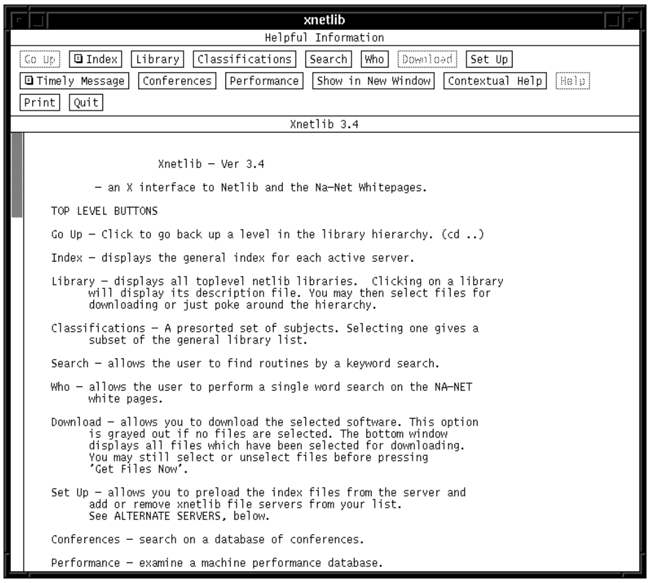

Figure 2: Help Screen

2.3.4 Xnetlib Man Page and Quick Reference Card

The source code distribution in the `xnetlib3.4/doc` directory include the man page for Xnetlib. To view it, `cd` to this directory and type `nroff -man xnetlib.man` You may wish to have your system manager install the Xnetlib man page on your system.

2.3.5 Xnetlib Client Program Operation

Start the Xnetlib client program by typing `xnetlib`. When Xnetlib starts up, it attempts to connect to each server on its list of active hosts. The default active host list is `netlib.org`, `netlib.brl.mil` and `softlib.cs.rice.edu`. When these connections have been tested, a window will appear displaying the Xnetlib help screen (Figure 2).

The top row of buttons controls mode changes. In general, Xnetlib reuses the bottom portion of the main window for displays instead of popping up new windows.

Clicking on `Library` displays the top-level listing of libraries available from the Netlib repository (Figure 3). You can think of this top-level listing as a set of UNIX-style directories. Clicking on a library name displays a description of the library's contents. For example, clicking on `lapack` displays the contents of the LAPACK library in library selection mode (Figure 4).

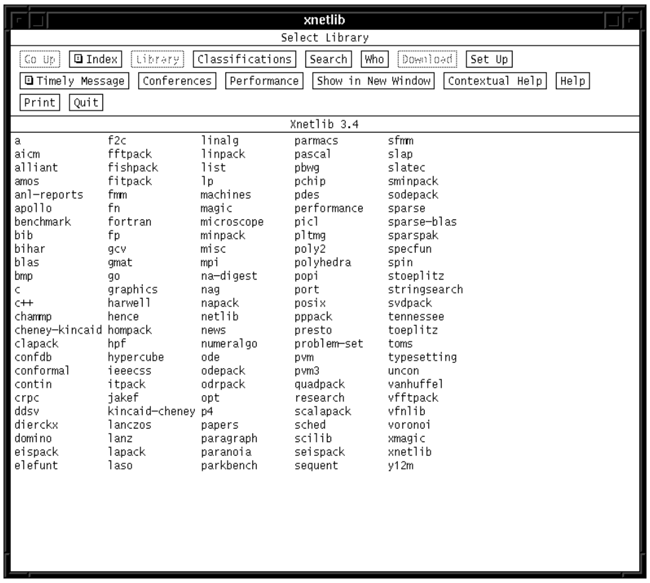

Figure 3: Library List

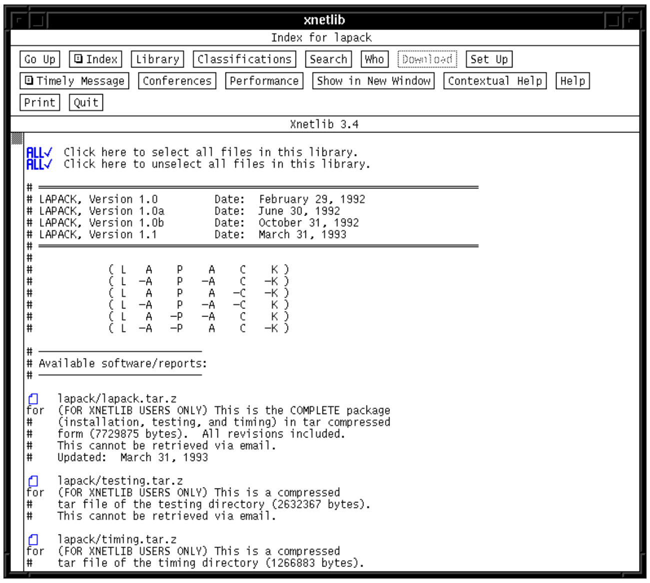

Figure 4: Library Selection Mode

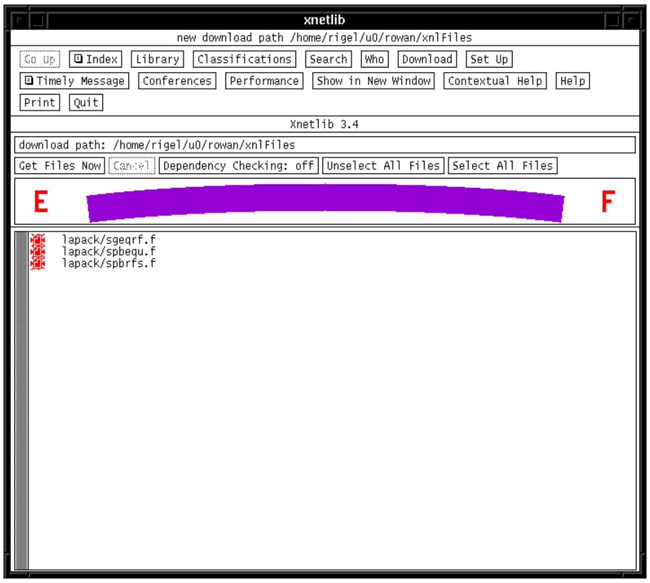

Figure 5: Download Mode

There may be further subdirectories, which are indicated by the folder icon. Files are indicated by the dog-eared page icon. You can select files to be downloaded by clicking on them. Move up the library hierarchy by using the `Go Up` button and move down by clicking on a library name.

When you have selected one or more files to download from the server, click the `Download` button to enter download mode (Figure 5). A list of the files you selected will be displayed. You can alter your selections if desired. Click `Get Files Now` to begin file downloading.

The default directory in which download files are placed is `xnlFiles` in your home directory. You can change the default directory by clicking on the `Download path` button. You can choose to have dependency checking either on or off (default is on) by clicking the `Dependency checking` button. If dependency checking is on, routines required by the requested file will be appended to the file before it is downloaded.

As you traverse the Netlib tree to examine libraries, the Netlib server at UT/ORNL downloads index files for the libraries to your site. Depending on your network connection, this file transfer may be hardly noticeable or may cause a significant delay. If you prefer to have all the index files loaded at once instead of as you need them, click `Set Up`, then `Press to Check Each Index File`. This will check every index file older than `indexLifetime` (see Section 7.4.1 on Xnetlib X Resources), and will copy from the server any that are missing or changed. To find out how to have several users at your site use

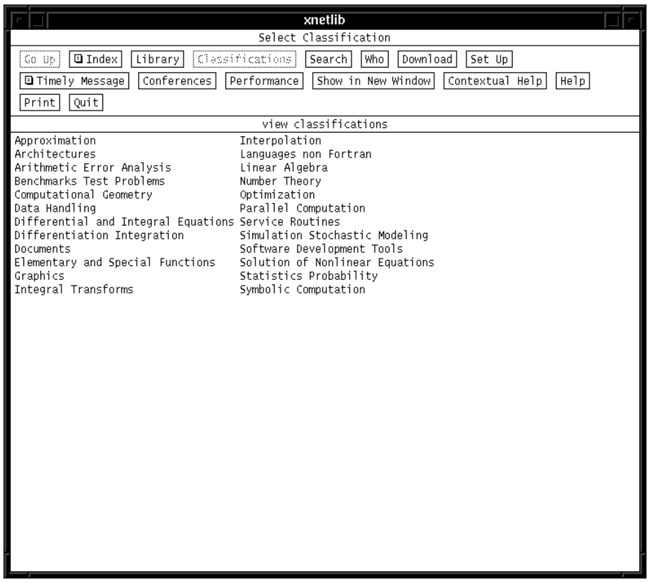

Figure 6: Classifications

one shared collection of index files, instead of several separate collections, see Xnetlib Installation and Customization in the System Manager's Guide (Section 7).

Clicking on `Classifications` lists library topics (Figure 6). The classification is an augmentation of the top level of the GAMS hierarchy. For more information about GAMS, see [2].

Selecting one of these topics displays a subset of the main library list.

Clicking on `Search` takes you into search mode . Buttons for the various types of searches are displayed, along with an explanation of these search types in the main window. The result of a search is a listing of files, from which you can choose files for downloading.

Clicking on `Who` changes to whois mode and allows you to query the NA-NET Whitepages and SIAM membership list for information about members of these groups.

Clicking on `Conferences` takes you into the Conferences Database mode. This mode is an experimental interface to a more general database service based on the relational database model. The Conferences Database contains conference and meeting announcements entered by UT staff and by Xnetlib users.

Clicking on `Performance` takes you into the Performance Database mode. This mode gives you access to benchmark performance data for a wide range of machines. For more information, see [1].

Option	Resource	
-dw	dontWarp	no gratuitous cursor motion
-nocq	confirmQuit	no confirm on quit
-cq	confirmQuit	confirm on quit
-showdown	showDownloadAnyway	download button always on
-noshowdown	showDownloadAnyway	download button not always on
-showwho	showWhoInfo	show help in who mode
-noshowwho	showWhoInfo	don't show help in who mode
-pub[lic]	publicData	write index files world writable
-nopub[lic]	publicData	don't write index files world writable
-dc	depCheck	dependency checking on
-nodc	depCheck	dependency checking off
-path <path>	xnlDownloadPath	where files are placed
-dp <path>	xnlPath	where index files are cached
-es <string>	entrySep	entry separator in who mode
-pc <string>	printCommand	printing command
-life <number of days>	indexLifetime	how long to keep index files
-email <email address>	email	your email address for our logs
-f[ile] <file name>	hostFile	xnetlib server list

Figure 7: Xnetlib options and X resources

2.3.6 Command Line Options

Xnetlib understands the normal X Toolkit options (`-display`, `-rv`, etc.) as well as `-help`, and those listed in Figure 7 next to the X resources they affect. See Xnetlib Installation and Customization in the System Manager's Guide (Section 7 in this document) for an more complete explanation of these X Resources.

2.3.7 Application Defaults File

There is no required application defaults file for Xnetlib. However, you can set application defaults for Xnetlib in your `.Xdefaults` file in the usual manner. For example, to set the background color for the Xnetlib client to seashell, include the following line in your `.Xdefaults` file:

```
xnetlib*background:            seashell
```

For further information about setting application defaults, see [9].

2.3.8 Support Address

Please send questions, comments, suggestions, or bug reports concerning Xnetlib to the following email address:

`xnetlib@cs.utk.edu`

2.4 Netlibget, a Command-line TCP/IP Client

Users who do not have X Windows may use `netlibget`, a TCP/IP command-line interface to the Xnetlib servers. This command-line client is included with the Xnetlib release. To build it from the Xnetlib source, type the following:

```
make -f Makefile.std netlibget
```

Index files retrieved by `netlibget` should not be used by the Xnetlib X client, since they lack certain header information that the Xnetlib client expects.

Running `netlibget` without any options will display the following usage message:

```
usage: netlibget [-v|-q] [-e your_email_address] [-s server[+port]]
       [-d|-k|-w] file or key
 -d to get dependent files
 -k for keyword search
 -ko for keyword-or search
 -ka for keyword-and search
 -w for whois search
 -v for verbose; -q for quiet
```

For example, to retrieve `dgeco.f` from `linpack` with dependent routines, type the following:

```
netlibget -d linpack/dgeco.f
```

For a `keyword-or` or `keyword-and` search, the list of keywords should be enclosed in single quotes. For example, to search for files whose descriptions contain the words *condition* and *number*, type the following:

```
netlibget -ka 'condition number'
```

2.5 Anonymous Access

For users who do not need the search capabilities provided by Xnetlib, anonymous access in the form of anonymous FTP and anonymous RCP has been provided. For more information about anonymous access, try the following:

```
finger anon@netlib.org
```

2.5.1 Instructions for Anonymous FTP

You can use anonymous FTP to retrieve Netlib files. Ftp to `netlib2.cs.utk.edu` and log in as anonymous. Use your complete email address as the password.

The Netlib `ftp` server supports automatic creation of compressed and/or tarred versions of files and directories. Just ask for `<filename>.Z` instead of

<filename>, or <directory>.tar.Z instead of each file in the directory, and it gets packaged up on the fly.

2.5.2 Instructions for Anonymous RCP and RSH

You can use anonymous RCP to copy files and anonymous RSH to list directories.

1. To copy a particular file to your system, type

   ```
   rcp anon@netlib.org:<filename> <local_filename>
   ```

   where <filename> is the filename in the Netlib repository and <local_filename> is the local file in which you wish to have the data stored on your system.

   For example, to retrieve sgetrf.f from lapack, type

   ```
   rcp anon@netlib.org:lapack/sgetrf.f sgetrf.f
   ```

2. To list files from a particular directory, type

   ```
   rsh netlib.org -l anon ls <directory>
   ```

   (On some machines, the command is remsh instead of rsh).

   Normal ls options work; ls -l gets the size, etc. You can use metacharacters by putting the argument to ls in single quotes.

2.6 Access via Gopher

A gopher server is running on a Netlib machine and can be accessed at the address netlib.org on port 70. The gopher URL is gopher:
netlib.org:70. The Netlib libraries are listed as entries in the top-level menu.

3 Future Plans

We plan to expand the Netlib repository from a handful of mirrored servers to a virtual repository consisting of many servers. All of these servers will be accessible from the Xnetlib client program. The services offered will include access to software, documents, and databases, as well as remote execution facilities. A user will not need to know where a particular service is located but will be able to employ both browsing and searching mechanisms to locate and access the desired service.

Future releases of Xnetlib will provide a means whereby newly inaugurated services using the Xnetlib protocol will not require use of a new version of the Xnetlib application. The special-purpose code describing the window layout and functions will be downloaded at run time from the remote service provider. This run-time binding of functionality will greatly enhance the flexibility and adaptability of Xnetlib.

As the software and document base grows in size and complexity, simple syntactic keyword searching mechanisms, such as that provided by `WAIS`, are likely to prove inadequate. We plan to add new search mechanisms that incorporate semantics and make use of more structured knowledge representations.

Netlib System Manager's Guide

This guide is intended to assist a system administrator in obtaining, installing, and maintaining the files, databases, and client/server software associated with Netlib. It also explains how to set up and maintain the underlying directories and databases.

4 Netlib Repository Setup and Maintenance

The Netlib repository is organized on a logical ordering of packages and software into various libraries. Generally, most new contributions can be placed into a pre-existing library (e.g., linalg, c++). Larger contributions, especially those with several seperable modules, can be placed into a sub-library within a top-level library. If the new package is large enough (or doesn't logically fit into a top level library) a new top level library will be created. The depth of nesting of libraries is limited only by what makes sense.

The repository is organized on disk as a UNIX directory tree, hereafter called the "Netlib tree". Assume for purposes of this explanation that the root of the Netlib tree is `/netlib`. Underneath this root are subdirectories for the various libraries. (For example, the EISPACK library is in `/netlib/eispack`). The library subdirectories contain either further subdirectories or files. Although the Netlib tree can be accessed via anonymous FTP and anonymous RCP and can also be browsed via the Xnetlib client, additional searching capabilities were thought to be desirable. Thus, the Netlib tree has been augmented by the inclusion of index, or description, files. The file `/netlib/master/index`, created nightly by concatenating the `.master` files in each top-level directory, contains a listing of all the libraries with descriptions. An index file in the subdirectory for each library lists and describes the library contents. The index files are in a format that is intended to be easily parsed by searching tools. Both the Netlib and the Xnetlib servers use the index files as the basis for their searching mechanisms.

A site may wish to replicate its repository contents, in order to achieve greater reliability and take advantage of load balancing. This replication has been carried out at UT/ORNL, where the Netlib tree is duplicated on two machines, one at UT and the other at ORNL. The copy at UT is the master, and the copy at ORNL is the slave. Whenever someone updates the master copy, that person manually forces propagation of the update to the slave copy using a program that runs the `rdist` command.

Autonomous sites that maintain copies of the same files need a mechanism to keep those copies consistent. Each file has one master copy and some number of slave copies. The site holding the master copy may be different for different files. Netlib has adopted a low overhead repository mirroring scheme, based on checksum files and `ftp`, that keeps slave copies of files consistent with the master copies. For more information on this repository mirroring mechanism, see [8].

4.1 Netlib Index File Format

The index is as flexible as we could think to make it while retaining a simple model. An index file contains one paragraph per regular file or directory, with paragraphs separated by empty lines. Each line of the paragraph starts with an attribute name, white space, and a corresponding value. If no value is known or applicable, that line is omitted. If all the files in a directory share a value, that line is moved up in the hierarchy to the directory entry in the parent index. Long values (e.g. past column 80) should be continued by starting the next line with a comma.

The entry for a regular file starts with the attribute name "file" and a full path name relative to netlib. Directories use the attribute name "lib". (There is exactly one file or library described per paragraph.)

The attributes and allowed values or intended meaning are as follows:

attribute	value
file	path/name *use lower case; many systems confuse upper/lower*
lib	path/name
for	*what problem does it solve?* This should be intelligible out of
	context, since a common use of the index is keyword search.
alg	*algorithm: what methods does it use?*
by	*name ¡email address¿, ...*
ref	*terse citation*
gams	classification codes ``‘‘send gams from bib’’``
size	999kB
prec	
	single *half; may be written "single real" in some contexts*
	double *full*
	single/double *contains both precisions with #ifdef or other switch*
	complex
	doublecomplex
see	*related files*
title	*if not same as file*
master	*site from which all others mirror this file or directory*
editor	*name ¡email address¿ appointed by editor-in-chief*
rel	excellent *widely tested code; firm theory; tractable problem*
	good *good reputation, but intrinsically hard problem*
	ok *some counterexamples, but as good as most alternatives*
	weak *fails without warning; better methods known*
age	stable *untouched in years; author still supports it*
	old *untouched in years; no one supports it*
	research *believed stable, but still relatively new*
	experimental *known to need polishing*
	superseded *Obsolete; kept for archival purposes*
kind	function *library routine (This is the default.)*
	command *standalone program*
	data *input data, measurements, sample output, etc.*
	text *documentation*
lang	*what languages are used?*
keywords	*terms as would be drawn from a subject thesaurus*
	see toms/index for examples
instance_of	file that this is derived from by some automatic means
	such as compression. The practice is discouraged, but may be
	necessary in isolated cases.
encoding	encoding of the files. In "MIME-type" format. For example,
	`application/x-tar`. They are listed left to right in the order that
	they were encoded, separated by white space. This may be elided
	if obvious from the filename, as in `foo.tar.gz`.
,	continuation
#	miscellaneous comments; usually bad, because automated tools
	won't know what to do with them. Use the proper keyword from this
	list, and put general information in the `readme` file.

The *lang* field is omitted if the filename suffix matches one of the implicit rules:

.ada	Ada
.awk	awk
.bas	Basic
.C	C++
.c	C
.f	Fortran-77
.html	HTML (hypertext)
.ps	PostScript
.r	Ratfor
.tex	TeX (commonly LaTeX)
.bib	BibTex input
.bbl	BibTeX output

See `netlib/crc/net/compressed` for more suffixes.

In general, we do NOT put information in the index that can be derived automatically. For example, we do not put the revision date, because that can be obtained from Netlib's checksum file or by an ftp list command. One exception is size; for files larger than 200kB, a remark is justified to warn people that might inadvertently download something much larger than they expected.

For several of the keywords (such as gams, see, lang) the attributes may be a comma-separated list.

Besides the index file, each chapter also has (in principle) the files

readme	general, unstructured information about the chapter;
changes	who did what to which files when;
index.html	same as index, but reformated for World Wide Web clients
.depend	which symbols are defined in which files;

The latter two are generated automatically and are not listed in the index.

Finally, here are some attributes that are not yet in use in Netlib, but may be in the future. See the RIG Proposed Standard RPS-0002 (1994), A Uniform Data Model for Reuse Libraries (UDM), available from the Reuse Library Interoperability Group via AdaNET at 800-444-1458. Their concept of "Asset" corresponds very roughly to Netlib's concept of "lib"; their "Element" is our "file". Some possible future attributes include the following:

URN	"Universal Resource Name" (when appropriate mechanism is someday agreed upon by the Internet community)
Abstract	an extended version of "for" and maybe other material
AcceptanceDate	date first version entered the collection
ComplianceToStandards	e.g. does the source assume only ANSI C, or also POSIX.1, or TCP/IP, or Winsock
Encrypted	true/false. By some other means you ask how and with what key.
DerivedFrom	(This might allow searches analogous to Science Citation Index in document publication.)

4.2 Repository Replication in Netlib

At UT/ORNL, the Netlib repository contents are replicated on two machines. The master copy resides on `netlib2.cs.utk.edu`, while the slave copy resides on `netlib1.epm.ornl.gov`. Both machines are registered with the Domain Name System as host addresses for the domain name `netlib.org`. Theoretically, the load balance for Xnetlib requests between the two machines should be about 50/50. The mail preference is currently set up so that `netlib1` gets most all of the email requests. If `netlib1` is down, however, email requests should get sent to `netlib2`.

Updates to the repository must be made on the master copy on `netlib2`. When a staff person makes an update, either by installing a new library or by changing or adding files in an existing library, that person is supposed to run a script that logs the change and asks him or her whether to propagate the change to `netlib1`. When the person answers yes, the update is propagated by running the `rdist` command. A nightly cron job runs an `rdist` job using a distfile that lists all the libraries that are replicated. The `rdist` job checks for discrepancies, propagates any changes, and notifies the Netlib maintenance staff of any such changes.

At AT&T, the email server runs on one machine (inside the security firewall) where the editorial staff can conveniently maintain the file tree. A second machine (outside the firewall) replicates the files, in compressed form, for `ftp` service. The latter machine runs the Plan9 operating system, which provides a file system that allows one to see the state as of any date in the past.

5 Email Netlib Server Installation

The Netlib email server software is available from the Netlib repository itself. The software provides mechanisms for processing user email requests, which may involve keyword searching and file retrieval. Slightly modified versions are in use to distribute statistical software from **statlib@temper.stat.cmu.edu** and by the TeX User Group to distribute TeX-related software from **tuglib@math.utah.edu**. Appendix A lists a selection of Netlib sites and sites running the Netlib email server.

5.1 Acquiring and Installing the Netlib Software

CAUTION: The most common problems with the email interface to Netlib are corrupted mail addresses, network errors, and so on. You ought to be reasonably expert with email before installing the Netlib email server.

To obtain the Netlib email server software, send the following message to `netlib@netlib.org`:

```
send netlib from misc
```

Netlib will return a shar file with instructions at the beginning as to how it should be unbundled. This guide assumes that the software is unbundled in a directory named `/netlib/admin`. After unbundling, carry out the following steps:

1. Edit the call to chdir in `bin/reply.c` to point to the place where the source code is stored on your system. `/netlib` is assumed for illustration here.

2. Edit `/netlib/admin/LIBS` to reflect what you are distributing. Items strictly for local users go in `LIBS.lcl` and `index.lcl`; edit function `groupid()` in `reply.c` to define "local users". If everything is public, `groupid()` can simply return 0. As it currently stands, "local" addresses are those whose machines are in `/netlib/admin/groups/local`; if any machine name in `groups/enemies` is found, processing is aborted.

   Suppose you have a library `eispack` that you want to install. Make the directory `/netlib/eispack`; copy in files such as `rg.f`; create a file `index` there; add some lines to `/netlib/admin/LIBS` like

   ```
   eispack => eispack
   eispac => eispack
   ```

   (That second line is to allow for misspellings; use your imagination and watch the logfiles for common mistakes.) Be sure that the line

   ```
   master => master
   ```

   is in LIBS, so that people can get the main index.

3. Edit the various disclaimers in `/netlib/admin/mess`. You may also wish to add disclaimer files in the source directories.

To activate mail processing:

```
*  if you run \command{sendmail}, put
          netlib: "|/netlib/admin/bin/reply"
          netlibd: "|/netlib/admin/bin/netlibd"
   into /usr/lib/aliases and execute newaliases;
*  if you run \command{upas}, put
          Pipe to \file{/netlib/admin/bin/reply}
   in \file{/usr/spool/mail/netlib} and make that file owned by
   "netlibd" and similarly put
          Pipe to \file{/netlib/admin/bin/netlibd}
   in \file{/usr/spool/mail/netlibd} and make that file owned by
   yourself;
*  else if your system has no equivalent mechanism, try the
   daemon in /netlib/admin/bin/Old-mail-sys.
```

The script `admin/bin/netlibd` contains (on line 3) `cd /netlib`, which you may need to change. Because Berkeley's alias facility provides no way to set the userid, you probably should put your name and address in the message so people know who is actually sending the mail.

To try the system out, `echo send index | mail netlib` and expect return mail in a couple minutes. A line should be added to `/netlib/admin/log` and `admin/stderr` should remain empty.

Once the basics are working, you can polish things a bit.

- Create ".depend" files in each Netlib directory to represent the relationship between source files there. Not only is this file itself useful to browsers, but by changing the lines in LIBS to the form

  ```
  eispack => -leispack
  ```

  you can teach Netlib to respond to a request for `rg from eispack` by sending not just `rg.f`, but also `balbak.f`, `hqr2.f`, and so on.

- You should permanently save `/netlib/admin/log` so that bug fixes can be distributed, traffic measurements made, and annual summaries sent to code authors. The format of the log is: date time [address] bytes-sent library/item possibly followed by: L = list, F = find. The [address] is followed by "l" if the address was recognized as local. In contrast, the copy of incoming messages kept in `/tmp/netreq` is for debugging mail headers and monitoring illegal request syntax. Discard when convenient (perhaps by an `rm` in `/etc/rc`) or, if you prefer, comment out the line in `reply.c:handle()` that writes the file.

6 NA-NET Database Setup and Maintenance

6.1 Acquiring the NA-NET Software

The NA-NET software in use at UT/ORNL is not yet in general release. For more information about a possible future release date, send email to `nanet@na-net.ornl.gov`.

6.2 The NA-NET Program

The NA-NET program handles the following:

- receipt of incoming mail for all NA-NET recipient addresses, including ordinary subscriber addresses (e.g., `na.joe`) and special functions (e.g., `na.help`, `na.join`, `na.whois`, etc.)
- receipt of bounced mail from attempts to send na-digests. These failed attempts are logged.

The NA-NET program is invoked by a modified version of sendmail whenever the modified sendmail receives a message addressed to:
`<something>@na-net.ornl.gov`.
Other machines know to send mail addressed to `na-net.ornl.gov` to `netlib2.cs.utk.edu` because the latter machine is registered as the "mail exchanger" for `na-net.ornl.gov` with the Internet Domain Name System.

6.3 NA-NET Files

The NA-NET programs and files are in the directory `/usr/local/na-net`. The NA-NET main program is in `/usr/local/na-net/na-net`. The NA-NET program gets the location of all files from a config file that is passed to NA-NET on the UNIX command line when NA-NET is called by sendmail to deliver mail. The config file is in `/usr/local/na-net/config`. It contains the following:

```
-----------------------------------------------------------------
libdir=/usr/local/na-net/lib
workdir=/var/spool/na-net
prefix=na
helpfile=/usr/local/na-net/help.txt
human=na-net@netlib2.cs.utk.edu
mail_domain=na-net.ornl.gov
errlog1=na.errlog1@na-net.ornl.gov
errlog2=na.errlog2@na-net.ornl.gov
master_file=/usr/local/na-net/nanet_names.master
digest_file=/var/spool/na-net/digest
```

```
logfile=/var/log/na-net/na-net.log
digest_ack_file=/usr/local/na-net/digest-ack.txt
delivery_error_file_1=/var/log/na-net/misc-delivery-errors.log
delivery_error_file_2=/var/log/na-net/delivery-errors.log
white_pages_file=/usr/local/na-net/whitepages.database
----------------------------------------------------------------
```

The purpose of these entries is as follows:

`libdir` – location of subsidiary programs to be called by NA-NET. (currently unused)

`workdir` – a scratch directory for NA-NET to write temp files, etc.

`prefix` – This is the prefix that NA-NET will accept at the beginning of addresses. NA-NET will remove it from the local part of a recipient address before processing.

The prefix is used to distinguish NA-NET addresses from ordinary local addresses on systems where the NA-NET system shares a mail domain with local users.

Normally the prefix doesn't change. However, you could run multiple NA-NET domains with different prefixes, simply by prefixes, simply by having sendmail call NA-NET with a different config-file for each prefix.

`helpfile` – location of the text file sent in response to a message addressed to `help`

`human` – Internet email address (not an NA-NET address) of a human being. This is used for mail to any of the following:

`na.net@na-net.ornl.gov`

`na-net@na-net.ornl.gov`

`nanet@na-net.ornl.gov`

`postmaster@na-net.ornl.gov`

It is also used as a reply address on responses to help, join, change, remove, sendlist, whois, join-wp, change-wp, and remove-wp commands, and digest submissions. (But not as the reply address on digests sent to subscribers!)

`mail_domain` – Internet mail domain for NA-NET addresses. This is used in error messages and also so that NA-NET will recognize addresses such as `user%na-net.ornl.gov@na-net.ornl.gov` which shouldn't occur, but do.

`errlog1` – This address is used for the return address on most responses to NA-NET commands. If a response bounces, the message will get sent back to this address. Normally `errlog1` is set up to feed back into into `errlog1@na-net.ornl.gov`, so that we can save the returned message. Unfortunately, if one of these response messages bounces, we can't do much about it; but the logs are sometimes useful to help answer questions or diagnose problems with someone's mail system. (See delivery_error_file_1, below).

`errlog2` – This should be an Internet email address. NA-NET will use this address as the envelope return-address on all outgoing na-digests. Normally this will point back to `<prefix>.errlog2@<mail_domain>`, which allows NA-NET to process and log bounced mail messages. (Unlike messages sent in response to commands, we can use the bounced digests to let us know whose addresses are no longer valid.) (See delivery_error_file_2, below).

`master_file` – location of the NA-NET subscriber database. See "File Formats" below.

`digest_file` – This is where incoming digest articles are stored. Each article is separated from the others by a line of the form "From sender date". Two blank lines are added before the "From" line and after each message.

`logfile` – This is a log of messages sent to NA-NET. Each line is of the form:

date time sender recipient status-code status text...

`digest_ack_file` – This file contains a message that is sent to people who send mail to the na.digest.

`delivery_error_file_1` – This is where bounced messages in response to NA-NET responses get filed.

`delivery_error_file_2` – This is where bounced digests get filed.

`white_pages_file` – location of the NA-NET Whitepages Database.

6.4 File Formats

1. na-digest subscriber database

   The subscriber database is an ordinary text file consisting of lines of the form

   lastname, firstname (na.key) email-address

   These lines are sorted lexicographically by lastname then firstname.

2. Whitepages Database

   The format of the Whitepages Database is as follows:

   (a) Each record is separated from the next by a newline.
   (b) Each field of the record is separated from the next field by a CR (carraige return, control-M)
   (c) Any newlines within a field are represented as control-A.

   It's difficult to read the file, but if the need arises it can be edited with a text editor.

   The fields are, in order:

```
last name
first name
middle name
other name
affiliation
office address
city_state_zip
country
office phone
research
home address
home phone
fax
email address
other
date                -- preferably in rfc 822 format
added by            -- email address of whoever added
                       the record
```

6.5 NA-NET Source Files

The source files are in `/usr/local/src/na-net`. Simply typing `make` should rebuild them.

Notus of any changes are in the file `ChangeLog`. The software is currently stable, but changes are made occasionally to fix (hopefully minor) bugs, to make the software more tolerant, easier to use, or easier to maintain.

6.6 Database Changes, Backups, and Cron Jobs

Database updates are currently performed by sequentially copying the nanet_names.master or whitepages_database files to "new" files (those ending in .new). If the file copy happens successfully, the current database is linked to <filename>.old and the .new file is renamed to nanet_names.master or whitepagesdatabase, respectively. The previous version of the file remains in <filename>.old.

The new files are locked during updates so that two concurrent writes cannot happen, though others can read the database while it is being updated. The actual update – replacment of the old database file with the new – is atomic; queries can never see an inconsistent copy of the database file.

In order to prevent race conditions or locking the database for an excessive period of time, some operations may have to be backed out. For example, if someone tries to delete a subscriber record using the first and last names of the subscriber, and more than one subscriber has those names. In this case NA-NET will detect the condition, issue an appropriate message, and discard the new copy of the database (which may already have some records deleted) instead of replacing the old file. (Thus, the presence of a .new file does not mean that file contains more current information.)

In addition to the normal system backups, and the .old files, there is a cron job (/usr/local/na-net/rotate.sh) that gets run once per day that saves an extra copy of the NA-NET databases. Currently these are kept for five days.

A summary of NA-NET activity for a given day is run from cron at 11:59 pm. The summary script is in (/usr/local/na-net/summary.sh)

There is currently no mechanism for rotating log files.

6.7 Sending to Digest Subscribers

Digest mailings are accomplished by sending a message to a special NA-NET address known to the digest moderator.

Replies to digests are currently sent to the na.digest address.

6.8 Surgery

If it is necessary to change NA-NET databases "by hand", the NA-NET system should be suspended. In the directory /usr/local/na-net there is a script called na-net.sh; this does nothing but exit with a "temporary failure" status. If the normal NA-NET binary is moved aside and the na-net.sh file put in its place, sendmail will queue any messages for NA-NET and keep trying every half hour or so. After renaming the na-net.sh file, wait a few minutes so that any NA-NET processes can finish up before editing the database. After making whatever changes are necessary, don't forget to rename the real NA-NET program back to na-net.

7 Installation and Customization of the Xnetlib Client

7.1 Acquiring the Xnetlib Software

To acquire the Xnetlib client software from Netlib send email to `netlib@netlib.org` with the line

```
send xnetlib.shar from xnetlib
```

in the message. Netlib will return the `xnetlib.shar` file by email.

Xnetlib is available by anonymous FTP from `netlib.org` in the xnetlib directory.

7.2 System Requirements

Xnetlib will build on nearly any Unix system. It runs under the X Window System, version 11, from MIT. It requires release level 4 or greater and the Athena widget libraries as supplied by MIT.

Xnetlib is known to run on the following systems:

- Convex.
- DECStation running Ultrix 4.1, 4.2, and 4.2A, and DecWindows. (For Xnetlib to compile with the vendor-supplied X Windows libraries, you must have the "Unsupported X11 Components" software subset loaded.) Xnetlib should also build and run with MIT's X11R4 or X11R5, but this configuration has not been tested.
- HP 9000 and MIT's X11R5. (Xnetlib may work with the vendor-supplied X Windows libraries, but this is not recommended.)
- IBM RS/6000 running AIX 3.1, 3.2, and MIT's X11R5. (Xnetlib may work with X11R4, but R5 is preferred.)
- NeXT Dimension and Co-Exist X11R4.
- Sequent Symmetry and MIT's X11R4.
- SGI 4D/25 running IRIX 3.3.3 and X11R4.
- Stardent (Kubota) Titan and vendor-supplied X11R4.
- Sun 3 running SunOS 4.1 and X11R4.
- Sun 4 running SunOS 4.1 or later and X11R4 or X11R5.

7.3 Building Xnetlib

Xnetlib makes use of the imake facility (via `xmkmf`) that comes with standard X Windows distributions. A generic Makefile is provided and can be used if imake is not present on your system.

After the executable is built, install it by copying the file `xnetlib` to an appropriate directory. There is no application defaults file to install.

```
With imake (xmkmf)

Type:

        xmkmf
        make


Without imake (xmkmf)

first edit Makefile.std to reflect your installation.
Then type:

        make -f Makefile.std
```

7.4 Customization of Xnetlib

7.4.1 X Resources

`publicData` – When a user runs Xnetlib, the index files for the libraries are downloaded from the server and cached in a special directory. For sites where many people use Xnetlib, it will save disk space if users share these index files . This can be accomplished by setting the `xnlPath` resource to some commonly writable directory and by setting the `publicData` resource to `True`. The effect of the `publicData` resource is to cause all index files to be saved world writable so they can be updated by anyone. Some sites set these defaults at compile time by setting the fallback resources for these variables. (Look for `fallback_resources` in `main.c`.)

`indexLifetime` – controls frequency of client to server communication. This resource sets the number of days an index file will be used before Xnetlib checks with the server to find out if the file is out of date. The default value is seven days.

`hostFile` – sets the name of host file, which contains the lists of servers to contact. The default is `$HOME/.xnetlibHosts`.

`email` – specifies the Internet email address of the user. Xnetlib will attempt to guess the email address but will very often be wrong. The email address is recorded in the server's log and is used to inform users of software bugs and updates.

`printCommand` – sets the format string of the print command. It should contain a "%s" which is replaced when the command is executed by the name of the temporary file used in printing the text. The default is "lpr %s".

`showWhoInfo` – determines whether instructions for adding your name to the NA-NET Whitepages Database are shown in Who mode. The default is True.

`entrySep` – sets the string to be displayed between entries in Who mode. The default is "——-".

`xnlDownloadPath` – sets the directory where files selected for downloading are to be placed. The default is `$HOME/xnlFiles`.

`depCheck` – sets the default value for dependency checking in Download mode.

`confirmQuit` – If the `confirmQuit` resource is True, you will be asked for confirmation before quitting Xnetlib.

`dontWarp` – turns off automatic cursor positioning if True.

`okColor`, `badColor`, and `cautionColor` – affect the status message window background. The defaults are green, red, and yellow.

`dial`, `needle`, `ef`, and `gasGauge.background` – affect features of the gas gauge in Download mode. The defaults are DarkViolet, yellow, red, and white.

8 Installing and Running Nlrexecd

Nlrexecd is the service provider daemon for the Xnetlib services. The nlrexecd daemons running at UT/ORNL currently provide access to the Netlib software and document repository, the NA-NET Whitepages Database, the Performance Database, and the Conferences Database. `Nlrexecd` is written to be a general service provider, however, and can be configured to offer an arbitrary set of services, as long as they speak the Xnetlib protocol. A new service can be added easily by providing the code for the function to be called when the service is invoked.

8.1 Acquiring the Nlrexecd Software

Nlrexecd is not yet in general release. Send email to `xnetlib@cs.utk.edu` for more information.

8.2 System Requirements

Nlrexecd should build and run on any Unix system supporting TCP/IP domain sockets. Nlrexecd does not require any type of X Windows support.

8.3 Building Nlrexecd

There are two flavors of Nlrexecd. The large server distribution provides file transfer and keyword and database lookup and is the basis for the main Xnetlib server. The small server distribution provides only the file transfer service. Site-specific services can easily be added to either server.

Building the small server requires a file called `nlrexecd_small.tar` and otherwise requires a file called `nlrexecd.tar`.

Untar the file in a suitable area. Examine the Makefile and make any site-specific changes you may require. Type `make` to build `nlrexecd`.

Modules used by the small server are also used by the large server. Code that is specific to one or the other within these common modules is differentiated by the SMALL_SERVER preprocessor symbol.

8.4 Services and Protocol

All actions performed by nlrexecd are indicated by a unique service name. The service required is passed by name to nlrexecd after a TCP connection is established from the client.

Xnetlib Protocol	
server (nlrexecd)	client (xnetlib or telnet)
listens on TCP port 5555	
	opens TCP connection to server
	sends newline terminated email address
	sends newline terminated service name
	sends newline terminated service specific data
service specific internal call is made passing client data and socket descriptor	

The nlrexecd protocol describes only what occurs up until the service name and data are correctly specified, after that point the connection is "taken over" by that service.

Note that requiring the client to provide any service specific data is a violation of the spirit of separation between the service and the nlrexec layer. It was done entirely to simplify the job of the service module writer. There are several instances in which the service simply ignores this data. In any case, it may not be omitted by the client.

8.4.1 Adding a Service to Nlrexecd

Adding a new service to the server code involves writing the function to be called when your service is invoked and adding the service name in the main module. The following example shows how to add a service called "howdy".

To add the service name in the main module edit the file `nlrexecd.c`.

1. Declare your function (which should return a `char*`) where the other service functions are declared.

   ```
   char *howdy();
   ```

2. Add the name of your service to the `service_list` structure.

   ```
   "howdy",
   ```

3. Add the name of your function to the `service_call` structure.

   ```
   howdy,
   ```

   Notice that the position of `"howdy"` in `service_list` should correspond to the position of `howdy` in `service_call`.

4. Create a new file called `howdy.c`. It should look something like this.

   ```
   #include <stdio.h>
   #include "nlrexecd.h"

   char *howdy(s, service, extra)
   int s ; char *service, *extra;
   {
       swrite(s, "hello world\n");
       return "howdy ok";
   }
   ```

   The string returned by a service function is written to the log file.

5. Add `howdy.o` to OBJS in the Makefile.

   Now type `make`.

   To test your new service start the server and use telnet to talk to it.

   ```
   csh> telnet localhost 5555
   Trying...
   Connected to localhost
   Escape character is '^]'.
   wade@cs.utk.edu                       -- you type this
   howdy                                 -- and this
   ```

```
nothing                                       -- and this
hello world
Connection closed.
csh>
```

8.4.2 Reserved Service Names

To avoid service name conflicts it is intended that Xnetlib service name prefixes will be maintained in a central registry. Entities will be provided a service name prefix that they can use to manufacture unique service names. This would work in a manner similar to the Domain Name System except with the most general qualifier at the beginning of the name instead of the end.

Reserved Service Names	
list-services ?	required by all servers
file-tag file-get file-get-dep	used by all servers
who keyword keyword-or keyword-and keyword-lsi keyword-literal keyword-literal-case	used by large server
performance dataserve f2c	reserved prefixes

8.5 Command Line Options

Option	Explanation	Default
-port PORTNUMBER	port at which nlrexecd will listen for client requests	5555
-dir DIRECTORY	effective root directory (argument to chroot)	`"/netlib"`
-nochroot	Don't do chroot	
-log PATHNAME	pathname for log file	`"/usr/local/logs/nlr.log"`

8.6 Keyword and Database Lookup Services

8.6.1 Keyword Lookup

The keyword lookup services `keyword`, `keyword-or`, and `keyword-and` use a keyword database that is manufactured nightly from the Netlib index files. This database is a full-text index and includes all words in the Netlib index files.

The `keyword-literal` and `keyword-literal-case` searches are slower because they do string matching on the index file descriptions themselves.

The keyword searching mechanisms are expected to be changed to use `WAIS` indexing and searching in the near future.

8.6.2 Latent Semantic Indexing

Latent Semantic Indexing is a method for automatic indexing and retrieval that tries to take advantage of the semantic, or conceptual, content of documents. The particular LSI technique used in Xnetlib at UT/ORNL employs singular-value decomposition to take a large matrix of term-document association data (in the case of Xnetlib, the documents are the netlib index files) and construct a "semantic" space wherein terms and documents that are closely associated are placed near one another. LSI tries to tackle the problems of *synonymy* (many ways to refer to the same object) and *polysemy* (more than one meaning for a term), so as to improve the recall and precision of retrieval. In fact, terms that do not actually appear in a document may still end up close to the document, if that is consistent with the major patterns of association in the data. Retrieval is carried out by using the terms in a query to indentify a point in the semantic space and by returning documents in the neighborhood of this space. At UT/ORNL, the SVD is done periodically on a matrix constructed from the netlib index files to produce a semantic space for the netlib repository. The `keyword-lsi` service invoked from nlrexecd carries out retrieval. For more information about LSI, see [3].

The particular LSI technique currently used in Xnetlib is patented and proprietary and can be used only with the written permission of Bell Communications Research.

8.6.3 Whois Service

The nlrexecd whois service does a lookup on the NA-NET Whitepages Database, augmented by the SIAM Membership List. Nlrexecd opens the file `/netlib/whois/whitepages.database` and does a linear search for matches to the user's query.

8.6.4 Performance Database Service

The Performance Database service uses the public-domain RDB relational DBMS developed by Walter Hobbs of Rand Corporation. RDB tables are stored as regular UNIX ASCII files and thus can be manipulated by the normal UNIX utilities. The RDB tables for UT/ORNL Performance Database are stored in the `performance` library in the Netlib tree. The `performance-query`, `performance-or-search`, and `performance-and-search` services invoked from nlrexecd carry out searches on the RDB tables.

8.6.5 Conference Database Service

The Conference Database service currently uses the Postgres extended relational database system. Postgres is in the public domain and is available via anonymous FTP from the University of California at Berkeley. The function called by nlrexecd for this service is `dataserve`. `dataserve` takes a database-system-independent client request and translates it into the appropriate Postquel language queries which are then executed on the Postgres database.

The conference description files are stored in the `confdb` library in the netlib tree. The filenames for these descriptions are the Postgres Object IDs for the corresponding entries in the Postgres database. Although the descriptions themselves are not stored in Postgres, a full-text index derived from these descriptions is stored in a Postgres relation. Postgres is used to handle searching by dates, keywords, and location. A geographical database stored in a Postgres relation handles geographical aliasing and hierarchical geographical relationships (e.g., retrieving entries for Belgium when asked for those in Europe).

9 Anonymous FTP Server for Netlib

An anonymous FTP server has been installed on `netlib2.cs.utk.edu` to provide anonymous FTP access to the Netlib directory tree. The Netlib anonymous FTP server is based on the Washington University ftp daemon, which in turn is based on the 4.3-Reno BSD ftp daemon. Minor modifications were necessary to get it to compile in a vanilla SunOS environment. The server supports several useful features including per-site, per-user, and per-directory access control, extensive logging, automatic display of "readme" messages when you cd to a particular directory, and automatic creation of compressed and/or tarred versions of files and directories. (Just ask for `<filename>.Z` instead of filename, or `<directory>.tar.Z` instead of each file in the directory, and it gets packaged up on-the-fly.)

In "anonymous" mode, `ftp` logs into the "ftp" account and changes its root directory to that account's home directory, so that it becomes impossible to access files outside of that directory. On `netlib2`, the "ftp" account's home directory is `/netlib`.

Configuration files are as follows:

```
/etc/ftpusers

        This file contains a list of users (like "nobody") who are
        not permitted to log in via ftp.

/usr/local/etc/ftpaccess

        This file specifies who can use the ftp server
```

```
        (you can create classes based on where someone logs in
        from and who they say they are), how many members of each
        class can log in at once, which files get printed out
        when you cd to a particular directory, a message to be
        displayed at login time, whether a class of user can
        request auto compression or auto-tar, what kinds of things
        can be logged, who can "upload" files, and where warning
        messages get mailed.

        See "man ftpaccess" for more information.

/usr/local/logs/ftpd.log

        This is where transactions get logged.
```

There are other config files, detailing other features of ftp, but they are not used by the netlib implementation.

For more information, see the man pages for `ftpaccess(5)`, `ftpconversions(5)`, `ftpcount(1)`, `ftpd(8)`, `ftphosts(5)`, `ftpshut(8)`, `ftpwho(1)`, and `xferlog(5)`. (On `netlib2`, these man pages are installed in `/usr/local/man`; make sure your MANPATH environment variable contains `/usr/local/man` ahead of `/usr/man`).

Netlib's ftp server is installed in `/usr/local/etc/ftpd`. The Sun-supplied binary is in `/usr/etc/in.ftpd`; but the file `/etc/inetd.conf` has been changed to point to `/usr/local/etc/ftpd`.

A few other programs are also needed to make anonymous FTP work properly – special (statically-linked) versions of `ls`, `compress`, and (GNU) `tar`. These are installed on `netlib2` in `/netlib/bin`. On `netlib2` there is also a dummy `/netlib/etc/passwd` file, which contains dummy entries for `root` and `ftp` – with fake passwords. These are so that the output of `ls -l` can use meaningful user names rather than uid numbers.

The sources are in `/usr/local/src/wu-ftpd-2.1a`. There are some documentation files there that detail how to configure it. For these instructions see `/usr/local/src/wu-ftpd-2.1a/INSTALL` and `/usr/local/src/wu-ftpd-2.1a/NOTES`.

10 Netlib Anonymous RCP Implementation

10.1 "anon" Account

An "anon" account exists on the machines `netlib1.epm.ornl.gov` and `netlib2.cs.utk.edu` for the purpose of allowing anonymous RCP access. Normal logins to this account are disabled by giving it a passwd field of "*". In addition, the shell for this account is a special shell named "anon-shell".

10.2 Invocation of Anonymous RCP

The remote client's rcp command invokes the rcp command on the netlib server machines via the remote shell service (`rshd`). `rshd` on the Netlib machines has been specially modified to accept the following syntax in `.rhosts` files:

- If the remote-user field is "*", and the remote-machine field is filled in, any remote user at that machine can execute commands.
- If the remote-machine field is "*", and the remote-user field is filled in, any user by that name on any machine can execute commands.
- If both fields are "*", anyone can execute commands for this particular user.

The `.rhosts` file for user `anon` on the netlib machines currently consists of the following line:

```
* *
```

The `rshd` program defines the environment variables REMOTE_HOST and REMOTE_USER for use by programs that it runs.

10.3 anon-shell

"anon-shell" is a very primitive command parser. Basically, it understands c-shell style quoting and globbing (wildcard expansion). When given a command, it splits it up into arguments, expands wildcards on each argument, and then attempts to execute that command. It has a built-in table of commands that it will attempt to run. It will refuse to run any commands that are not in its hard-coded table.

"anon-shell" also logs every command executed, along with the remote user and host, via the syslog facility. Currently it uses `LOG_DAEMON` and `LOG_INFO`. `netlib2`'s syslog currently stores such entries in `/usr/adm/anon-rcp-log`.

"anon-shell" currently has two commands: `rcp` and `ls`. `rcp` is used by the client rcp program to retrieve remote files. `ls` can be used to browse directories.

IMPORTANT NOTE: "anon-shell" runs set-uid to root and passes root privilege to any commands that it runs. Thus, it is dangerous to add new commands without going over them carefully.

10.4 Modified `rcp` and `ls` commands

The versions of `rcp` and `ls` installed on `netlib1.epm.ornl.gov` and `netlib2.cs.utk.edu` have been modified as follows:

- Both `rcp` and `ls` immediately do the following:

```
chdir ("/netlib");
chroot ("/netlib");
setuid (getuid ());
```

  thus limiting their view of the world to everything under `/netlib`, and turning off any special privileges.

- The modified `rcp` can deal with not having an `/etc/services` or `/etc/passwd` file.

- The modified `rcp` has all calls to `mkdir()` and `open(...,O_CREAT,...)` `#ifdef`-ed out and replaced with code that prints "Permission denied". In general, the file receiving code is disabled, but it will talk protocol with the client rcp and return error messages. The file sending code works normally.

- `rcp` and `ls` are statically-linked binaries, since they have no access to system shared libraries.

10.5 Locations of files

The source to the modified `rsh` program is in the directory `/usr/local/src/rsh`.

The specially modified `rshd` program is installed in `/usr/etc/in.rshd`. (the original one is in `/usr/etc/in.rshd.ORIG`)

The sources to the other commands are in `/usr/local/src/anon-rcp` and its subdirectories. The subdirectories `ls`, `shell`, and `rcp` contain the sources to `anon-ls`, `anon-shell`, and `anon-rcp`. These are installed in `anon`, which is currently `/usr/local/homes/anon`.

Logs of transactions are currently kept in `/var/adm/anon-rcp-log`.

A Netlib Sites

A.1 Sites Mirroring the Netlib Repository

```
New Jersey      netlib@research.att.com
Tennessee       netlib@netlib.org
Norway          netlib@nac.no
England         netlib@ukc.ac.uk
Germany         eLib@zib-berlin.de
Taiwan          netlib@nchc.edu.tw
Australia       netlib@draci.cs.uow.edu.au
```

A.2 Some Sites Using the Netlib Email Server to Distribute Other Types of Software

The software that runs the Netlib email server is available from the Netlib repository. It can be retrieved by sending the message **send netlib from misc** to **netlib.org**. A number of groups have acquired the email distribution software, and a few of these are listed below.

```
The former \file{netlib/matlab} directory is now maintained at
          matlib@mathworks.com.
A collection of statistical software is available from
          statlib@temper.stat.cmu.edu.
The TeX User Group distributes TeX-related software from
          tuglib@math.utah.edu .
The symbolic algebra system REDUCE is supported by
          reduce-netlib@rand.org.
Parallel software and information about parallel processing
is available from
          parlib@hubcap.clemson.edu
```

References

[1] M. W. Berry, J. J. Dongarra, and B. H. Larose. PDS: A performance database server. *Scientific Computing*, 1993. (to appear).

[2] R. F. Boisvert, S. E. Howe, and D. K. Kahaner. The Guide to Available Mathematical Software problem classification system. *Comm. Stat. - Simul. Comp.*, 20(4):811–842, 1991.

[3] S. Deerwester, S. Dumais, G. Furnas, T. Landauer, and R. Harshamn. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407, Sept. 1990.

[4] J. Dongarra, T. Rowan, and R. Wade. Software distribution using XNETLIB. Technical Report CS-93-191, University of Tennessee-Knoxville, June 1993.

[5] J. J. Dongarra and E. Grosse. Distribution of mathematical software via electronic mail. *Commun. ACM*, 30(5):403–407, May 1987.

[6] J. Feigenbaum, E. Grosse, and J. A. Reeds. Cryptographic protection of membership lists. *Newsletter of the International Association for Cryptologic Research*, 9(1):16–20, 1992.

[7] G. E. Forsythe, M. A. Malcolm, and C. B. Moler. *Computer Methods for Mathematical Computations.* Prentice Hall, Inc., 1977.

[8] E. Grosse. Repository mirroring. *ACM Trans. Math. Softw.*, 1994. to appear.

[9] A. Nye and T. O'Reilly. *X Window System User's Guide.* O'Reilly and Associates, Inc., 1993.

Index