The TUGLIB Server[1]

Nelson H. F. Beebe

Center for Scientific Computing
Department of Mathematics
South Physics Building
University of Utah
Salt Lake City, UT 84112
USA

Beebe@science.utah.edu

1 Introduction

Scores of sites on the worldwide Internet now provide access to assorted collections of software relating to TEX and METAFONT. In many cases, these are accessible only via the Internet mechanism known as *anonymous* `ftp`, a scheme that permits logins from unknown users, usually on other machines, with very restricted access. The name `ftp` is an acronym for *file transfer protocol*.

Generally, only a portion of the file tree is visible to the anonymous user, and the command repertoire is usually limited to little more than directory listings and file retrieval. Only a few sites permit the anonymous user to deposit files in the anonymous login directory. Anonymous `ftp` provides a means whereby individual remote users can access file archives, browse around in the file tree, and retrieve selected files, all without troubling the staff or other users of the local machine.

While anonymous `ftp` has been enormously useful to the Internet community, it is available only between sites that have direct Internet connections, and on one of which, anonymous `ftp` logins have been enabled. Sites with only electronic mail connections to the Internet, such as those on other networks, like Bitnet, Junet, SPAN, and Usenet, and those on networks which are incompatible with the Internet, such as JANET in the UK, are prevented from using anonymous `ftp`. Similarly, sites on the Internet that have security restrictions, which includes many commercial, government, and military connections, may have restrictions that allow only e-mail access.

2 The TUGLIB connection

To improve the access to the TEX archives and other software at Utah, I have installed a modified version of the `netlib` server [1], which has been renamed `tuglib`. This server provides a means whereby remote users can send electronic mail messages containing service requests to a daemon program. The daemon parses the requests, logs them, and responds to them.

The `tuglib` daemon program runs on a local UNIX system at Utah, but the mail access is actually through a mail forwarding address on another machine, `tuglib@science.utah.edu`. The reasons for this separation are:

- `science.utah.edu` is a more widely-known host with a name which has been registered on the Internet for several years. It is therefore likely to be known on those machines which still have not upgraded from fixed host tables to domain name servers for Internet addressing.
- The `tuglib` software runs only on the UNIX operating system. `science.utah.edu` is a DEC-20/60 running TOPS-20, but in late 1990, it will likely be retired and replaced by a UNIX system that will answer to the same Internet host name (but a different numeric address).
- Separation of the server from the mail drop provides flexibility in configuration. In response to load patterns, we could change the machine running the `tuglib` daemon without having to make the change known to thousands of users who might wish to use the `tuglib` service.
- Through the wonders of NFS (*Network File System*), the UNIX system running the `tuglib` daemon is able to mount the TOPS-20 file system, because `science.utah.edu` runs an implementation of NFS developed by Mark Lottor at SRI. This makes the archives of two quite different machines available through a single service.
  Had we purchased NFS support software for VAX VMS, it would have been possible to provide ac-

[1] To be published in TUGboat, © 1991, TEX Users Group.

cess to our VAX 8600 as well; that will regrettably not happen, because our plans are to retire it a few months after the DEC-20.

The present configuration of `tuglib` provides several services:

- ask for help;
- list contents of a file directory;
- find a file in the archive;
- send one or more files; binary files are automatically encoded into a subset of the printable ISO/ASCII characters for e-mail transmission;
- query the TUG address file for membership information;
- check load libraries to determine file dependencies, so that if a request is made for a particular file, all other files that it references are automatically sent as well.

The last capability is not currently used by `tuglib`, since the bulk of the software in the distribution is TEX files (macros and fonts), rather than source code of mathematical libraries like LINPACK and EISPACK for which `netlib` library support was originally developed.

`tuglib` also provides some internal services, such as logging of requests (both successful and failed), and exclusion of users listed on an 'enemies' list. The latter has not yet been needed, but support is already there should it ever become necessary.

The log of successes provides a useful record of utilization of the service that may be needed to convince local administrators of its value.

The log of failures is useful in guarding against break-in attempts, or hogging of resources.

The log can also be for finding out whether alternate query syntaxes might be useful; for example, `envoyer`, `get`, `mail`, `request`, and `sned` are all recognized as synonyms for the `send` command.

`tuglib`'s e-mail responses come mostly from external files, rather than from text embedded in its programs, making it easy to customize for particular applications, and updates of textual information can be installed without recompilation of the software.

3 Getting help

The simplest command recognized by `tuglib` is `help`. It produces a response containing a short description of the `tuglib` service with sample commands.

Synonyms for `help` include `directory`, `index`, `info`, and `information`.

4 File retrieval

Files can be retrieved by commands of the form
`send filename` or
`send filename from directory`.
Punctuation is ignored, and unrecognized words are discarded. A request like

```
Please send me the index from the ftp
directory.  Thanks for your help.
```

is recognized: you will get both an index, and a help response.

Directories are named as in UNIX, that is, a series of one or more names, separated by slashes, with no embedded blanks. If no directory is specified, the top-level `tuglib` directory is assumed. Here are some examples:

```
send index
send index from ftp
send plain.tex from tex/inputs
send uudecode.c from support
send 00tdir.lst from tex/pub/cweb
```

UNIX symbolic links (duplicate names for the same file) are used to make particular file trees accessible to `tuglib`; the file in the last example, `tex/pub/cweb/00tdir.lst`, actually resides elsewhere in the file system (in fact, on the DEC-20 as the file `aps:<tex.pub>00tdir.lst`), but appears to `tuglib` to be the UNIX file with the absolute path `/tuglib/tex/pub/cweb/00tdir.lst`.

The top-level `tuglib` directory, `/tuglib`, contains only a small number of files at present:

`ftp`	symbolic link to the anonymous `ftp` directory on `science`
`support`	support software, mostly for encoding of binary files into printable characters
`tex`	symbolic link to the TEX tree on `science`

For security reasons, you cannot trick `tuglib` into sending an arbitrary file from elsewhere in the file system by specifying an absolute directory path in the `send` request. The leading slash is stripped, so that the file name *always* appears at or below the `tuglib` home directory.

`tuglib` does not provide any mechanism for wildcard matching of file names, mostly out of concern for security, and limiting e-mail traffic. Only those files that are explicitly listed in index files are visible via `tuglib` (unless the remote user is rather good at guessing names).

Ideally, each directory accessible to `tuglib` should have an index file named (naturally), `index`, containing names of files and a short description of their contents. Here is a portion of the index from the `ftp` directory, slightly edited to fit in these narrow columns.

```
PS:<ANONYMOUS>INDEX..7, 9-Dec-89 17:17:54,
Edit by BEEBE

This is an index of files available in
the anonymous ftp login directory on
science.utah.edu.

00DIR.CMD           -- FTP command file
                       to retrieve files
                       alphabetically
00DIR.LST           -- alphabetical
                       directory listing
00NEWS.TXT          -- brief
                       announcements
                       of new additions
                       to the collections
00PCDOS.TXT         -- setting up TeX for
                       PC DOS
...
TEX-FOR-APPLE-MACINTOSH.TXT
                    -- sources of
                       Macintosh TeX
TEX-FOR-ARABIC.TXT  -- sources of TeX for
                       Arabic typesetting
TEX-FOR-IBM-PC.TXT  -- sources of IBM PC
                       TeX
TEX-FOR-PICTURES.TXT-- combining graphics
                       with TeX
...
```

However, keeping such an index up-to-date is a demanding task, considering that at present, there are nearly 130 file directories and 8000 files in the TEX tree alone. Consequently, in most cases, only major directories will have an `index` file. To supplement these index files, a batch job is run periodically to automatically create four special files in every directory accessible to `tuglib`:

`00dir.cmd`	alphabetical list of files as `ftp` and `tuglib get` commands
`00dir.lst`	alphabetical verbose directory listing
`00tdir.cmd`	reverse time-ordered list of files as `ftp` and `tuglib get` commands
`00tdir.lst`	reverse time-ordered verbose directory listing

The `00dir.cmd` and `00tdir.cmd` files are handy for initiating a retrieval of a complete file directory, since their contents can be shipped back almost verbatim as requests to `tuglib`. The verbose directory listings contain file sizes in bytes and the last-write time stamp. A reverse time-ordered listing makes it easy to find out what is new.

The format of the directory listings depends on the particular operating system; here is part of the `00tdir.lst` file in the `tex` directory, which resides on TOPS-20; some reformatting has been necessary to make it fit here:

```
  APS:<TEX>
00LAST30DAYS.TXT.20;P777752 48 121566(7)
            23-Aug-90 03:08:41 OPERATOR
00LAST7DAYS.TXT.23;P777752   2 3641(7)
            23-Aug-90 03:04:47 OPERATOR
00RECENT.LOG.1;P777752       6 2707(36)
            23-Aug-90 03:02:14 OPERATOR
00DIR.CMD.1;P777752          1 2171(7)
            21-Aug-90 07:21:07 OPERATOR
00DIR.LST.1;P777752          5 11116(7)
            21-Aug-90 07:20:59 OPERATOR
00INVERTED-INDEX.TXT.15;P777752 6 15360(7)
            19-Aug-90 18:42:54 OPERATOR
...
```

Here is how to dissect one of these entries:

`00LAST30DAYS.TXT`	file name
`.15`	generation number
`;P777752`	protection bits
`6`	count of disk pages (512 36-bit words)
`15360`	count of bytes
`(7)`	byte size
`19-Aug-90 18:42:54`	time of last write
`OPERATOR`	user who last wrote the file

Unlike the UNIX file system, the TOPS-20 file system is case-*insensitive*; file names are conventionally spelled in upper-case, but you can write them in lower-case, or even mixed-case. Text files are normally stored with 7-bit bytes, which is sufficient for the ASCII character set; TEX binary files, and files intended for use on other systems, have 8-bit bytes; native binary files have 36-bit bytes. You can always omit the generation number, since the default is to return the highest existing generation. We could fetch the sample file by a request of the form `send 00last30days.txt from tex`.

While it would be possible to generate these directory listing files from the UNIX host, doing so would lose the byte-size information, which is needed for correct `ftp` access, so I prefer to generate them on the TOPS-20 host instead.

5 Finding files

With the large number of directories, and the limited directory listing access provided by `tuglib`, finding your way around a file tree as big as the TEX one can be a daunting task. To that end, batch jobs that run at regular intervals produce other helpful indexes:

`00inverted-index.txt`	inverted index giving, for each unique file name in the tree, a list of directories that contain it
`00last30days.txt`	a verbose directory listing of files changed in the last 30 days anywhere in the TEX tree
`00last7days.txt`	a verbose directory listing of files changed in the last 7 days anywhere in the TEX tree

Some directories will contain a file named `00readme.txt` which gives an overview of the directory contents. The `00` prefixes on these files are to make them come near the beginning of directory listings, where they are more likely to be noticed. On a case-sensitive file system like UNIX, they would probably be named entirely in upper-case letters, which, in ASCII, collate before the lower-case letters normally used for file names.

The large number of files available in an archive such as the one at Utah makes it rather difficult for external users to find desired files, since they are not able to login directly and use directory listing commands. The `tuglib find` command helps to remedy this problem. It searches two standard files, one of which is prepared by the `tuglib` maintainer and contains one-line summaries of the contents of every file directory, and the other is the `00inverted-index.txt` file described above.

A request like `find latex.tex` will produce a response with the names of all the file directories that contain the file `latex.tex`. Similarly, the request `find music` will list not only the name of the directory containing music fonts, but also all of the files found in that directory, because every line in `00inverted-index.txt` containing the text 'music' is matched.

6 Large files and binary files

Compared to `ftp`, electronic mail places some severe restrictions on file transfers:

- Message lengths are limited. 32 kilobytes is a reasonable upper bound; larger messages may be delayed, or returned to the sender, by some mail gateways.
- Messages may contain only printable characters; binary files cannot be sent without further encoding.
- Some IBM mainframe mail gateways corrupt mail that passes through them by having inconsistent inbound and outbound translations between ASCII and EBCDIC character sets. These may result in unrepairable many-to-one mappings; for example, curly braces are often mapped into the letters E and L, which is a disaster for TEX and C files.
- File names and file attributes cannot be automatically attached to e-mail messages.
- `ftp` is based on reliable network protocols like TCP/IP, and `ftp` transfers are always between pairs of machines, with no intermediaries, so file transfers do not corrupt files. E-mail often goes through intermediate machines that alter characters, truncate long messages, trim long lines or trailing blanks, or just discard the message altogether.
  These problems do not exist for Internet-to-Internet electronic mail, since it too is based on point-to-point reliable protocols, but they often occur in e-mail between an Internet site and one on some other network, like Bitnet or Usenet.
- A line beginning with a period terminates the mail message on some systems.

To deal with the message length limitation, `tuglib` automatically splits large files into parts which are mailed separately with distinctive headers, like `latex.tex (3 of 18)`.

File splitting is desirable even for `ftp` access, because long transfers may suffer timeouts that terminate the connection. A simple utility, `bsplit.c`, is available in the `support` directory, for splitting binary files into smaller parts. To avoid destructive padding with garbage characters on record-oriented file systems like VAX VMS, the sizes of each part (except possibly the last) are chosen to be a multiple of common file system block sizes, typically 512 bytes. For example, the command

```
bsplit -32768 fonts.tar
```

would split the file into 32KB parts named `fonts.tar-001`, `fonts.tar-002`, and so on. The original file can be recovered by appending the pieces in order; on a reasonable file system that provides alphabetically-sorted directories (or at least the illusion thereof) this can often by done by a single command using wildcard pattern matching, as in UNIX:

```
cat fonts.tar-??? >fonts.tar
```

Users on deficient file systems, like that of PC DOS, may have to work a little harder to reconstruct the original file.

`tuglib` normally refuses to mail very large files; this limitation is removed by making such files available in split parts.

Binary files are automatically recognized by `tuglib`, and are sent as *xxencoded* files. A file is regarded as 'binary' if it contains any non-printable characters other than carriage return or line feed, or if it has lines longer than 72 characters.

Because xxencoding is a new scheme developed for `tuglib`, a header is appended to the response to describe the encoding in sufficient detail to allow

the recipient to write a program to decode the message. Of course, decoding programs are available from the `tuglib support` directory, so in practice, few `tuglib` users will ever have to write their own decoder.

Xxencoding is a generalization of UNIX uuencoding, and the `xxencode` and `xxdecode` programs will handle uuencoding as well. Xxencoding was developed to deal with e-mail corruption, and to facilitate reassembly of large messages that have been sent in parts. Both encoding schemes represent three 8-bit bytes as four 6-bit bytes, and output is assembled into lines less than 65 characters in length, so as to avoid destructive truncation of long lines by anti-social mailing software.

Uuencoding biases the 6-bit bytes by 32, to move them into the range of printable ASCII characters. One variant of `uuencode` remaps the encoded blank (ASCII 32) to back accent (ASCII 96), which is the same character to `uudecode` (it only looks at the lower six bits in a character). This change removes blanks from the output encoding, and avoids damage from blank-trimming mailers.

Xxencoding instead maps a 6-bit byte into plus, minus, digits, upper-case letters, or lower-case letters, which is a 64-character set that is more likely to be immune to translation corruption. The translation table is included in the output, and will be used by `xxdecode`, so the encoding can survive one-to-one character remappings. `xxencode` also prefixes each line with a two-character sequence, 'xX', and on completion of encoding, appends a byte count and a CRC-16 checksum to the output.

Cyclic redundancy checksums are superior to simple checksums obtained by adding or exclusive-OR'ing data byte sequences. Such methods cannot detect bytes out of sequence, and can fail to detect even two single-bit errors, such as in two consecutive bytes with an inverted bit in the same position.

By contrast, the CRC-CCITT checksum used by the ANSI X.25, ADCCP, HDLC, and IBM SDLC protocols detects error bursts up to 16 bits in length, and 99 percent of error bursts greater than 12 bits. The CRC-16 checksum used by DDCMP and Bisync, and by `xxencode` and `xxdecode`, detects error bursts up to 16 bits, and 99 percent of bursts greater than 16 bits in length.

`xxdecode` ignores any line without the 'xX' prefix, allowing input to consist of a concatenation of several mail messages without the necessity of stripping mail headers and trailers. `xxdecode` also validates the byte count and checksum so as to detect corruption. Regrettably, `uudecode` has no comparable facility; it will happily produce garbage from corrupted input with no warning to the user.

Differences in end-of-line terminators on various operating systems are a minor nuisance; carriage return (Apple Macintosh), line feed (UNIX), and carriage return followed by linefeed (TOPS-20 and PC DOS) are all in use. All of these, and more, are supported on VAX VMS. As long as files can be transferred as plain text, `ftp` and e-mail handle 'lines', and line terminators appropriate to the receiving system will automatically be supplied. When files are encoded however, the line terminators are encoded with them. Thus, transfer of a file from one of the directories residing on TOPS-20 to a UNIX system will result in a non-native carriage return at the end of each line.

To deal with most of this problem, two utilities, `dos2ux` and `ux2dos`, are provided in a single shell bundle, `dos2ux.shar`, in the `support` directory. They take a list of files on the command line, and convert CR LF to LF or LF to CR LF, and also preserve the last write date of the original file. I've not yet written the `mac2ux` and `ux2mac` variants to handle conversion between CR and LF terminators, but they could be easily generated from the `dos2ux` and `ux2dos` code. Since the operations are so similar, it would probably make sense to merge them into a single utility whose operation was controlled by a command line option, or by the name of the file it was stored in.

Retrieval of a complete directory having many small files is painful because of the many `tuglib` requests needed. The solution is to make directory contents available in a single archive file, such as the `.arc` format widely used on PC DOS, or the UNIX `.tar` and compressed `.tar.Z` formats. Besides allowing a group of files to be retrieved in one request, the archive file preserves exact file names and importantly, file time stamps. We have made several large collections, including all of our public fonts, available this way.

Public-domain implementations of the `arc`, `compress`, and `tar` utilities are available for several operating systems, including PC DOS, TOPS-20, UNIX, and VAX VMS, so the use of these archive formats should not pose a problem for most `tuglib` clients.

7 TUG membership query

The TUG address data base is kept in a specially-formatted secret file in the `/tuglib` tree, inaccessible to anyone but the super-user (UNIX `root` login) or the `tuglib` daemon. A multi-line address like

> Nelson H. F. Beebe
> Center for Scientific Computing
> Department of Mathematics
> 220 South Physics Building
> University of Utah
> Salt Lake City, UT 84112

Tel: (801) 581-5254
FAX: (801) 581-4148
E-mail: Internet: beebe@science.utah.edu
TUG Board of Directors

is reformatted into a single-line entry with unprintable control characters separating the original lines.

The `whois` (or `who is`) command uses a shell script to invoke the UNIX `grep` command to find matching lines in the address file (ignoring letter case and punctuation), and then converts the magic separator characters back into normal newline characters. Thus, the above entry could be retrieved by any of several different commands:

```
whois nelson beebe
whois beebe
whois 581-5254
whois SALT laKe CiTy
```

Each word in the `whois` query is matched separately against the *entire* address entry. You need not remember people's initials, and you can use `whois` on parts of the address other than the personal name.

Of course, `tuglib` is still just a stupid computer program: if you send `who is Dave Kellerman`, it will not understand that `Dave` is short for `David` and it will fail to match `David Kellerman`. Personal names can be abbreviated to a leading prefix, however: `whois Don Knuth` will find Donald E. Knuth's address.

It is quite possible that multiple addresses match a `whois` query: `whois sweden` potentially can list the addresses of all TUG members in Sweden. However, organizations, including TUG, guard their membership lists with care, partly because such lists have commercial value, and partly out of concern for privacy. Germany, for example, has laws that severely restrict the use of address data bases. Consequently, `tuglib` will refuse to send more than a small number of addresses in response to a `whois` command, and it makes no provision for restarting a `whois` search. You cannot retrieve the entire list by a command like `whois *` (for the UNIX `grep` utility, the `*` matches zero or more of anything, that is, everything).

My original intent in setting up `tuglib` was to augment it with a mail forwarding service, such that electronic mail sent to an address like `malcolm.clark@science.utah.edu` would automatically be mapped to that member's real Internet address and forwarded. Such a forwarding list is maintained for the numerical analysis community on the machine `na-net.stanford.edu`; mail to `moler@na-net.stanford.edu` will always get to Cleve Moler, no matter where he is. This is a very convenient service, since a community of researchers can easily keep in touch, even though they may be moving often.

Consultation with the maintainers of `na-net` revealed that a surprising load is caused by this forwarding service, and at times, several CPUs are kept busy just handling the mail forwarding. Since the machines on which `tuglib` currently runs have many other duties as well, and provide `tuglib` as a free community service in their spare time, I abandoned further ideas for automatic mail forwarding.

The `whois` command eliminates mail forwarding overhead, since presumably an address will be looked up once, and then e-mail correspondence will be initiated directly by the users themselves. `whois` provides the additional service of supplying postal addresses, telephone numbers, TUG committee affiliations, and any other relevant information that happens to be recorded in the complete address entries.

There is nothing magic about the address data base handling. If you wanted to, you could easily eliminate the restriction on the number of matches returned, and then implement a recipe data base lookup to answer queries like `who is garlic`. A code change to accept the alternate request form `what has garlic` would then be desirable.

8 Future extensions

Since it built upon the lessons of `netlib`, I feel confident that `tuglib` is quite satisfactory in its current configuration. There are a few things that I would like to add, if time permits.

While automatic xxencoding of binary files avoids the e-mail problems noted earlier, for files that only happen to have tab characters and form feeds, encoding is unnecessary on Internet-to-Internet connections. On the other hand, perfectly normal TEX files sent in e-mail that goes through brace-corrupting gateways will be damaged, and may not be encoded by `tuglib`. These situations suggest that the remote user should be able to control whether encoding is applied, and if so, what form of encoding.

When `uuencode` can be used safely, it would probably be more convenient than `xxencode` for UNIX users, because `uuencode` is already available on UNIX. Other encoding schemes are available as well, including `atob` and `btoa`, and `bencode` and `bdecode`. This suggests an addition of new command verbs to augment the `send` command.

Directory listings are currently only available through the prior creation of the `00dir.lst` and `00tdir.lst` files in each directory. Perhaps it would be advisable to generate these dynamically in response to a `tuglib` command; they would then be guaranteed to be up-to-date, and disk space would not be used to store them. However, those files are also useful for anonymous `ftp` retrievals, because it is not always possible to get more than a bare list of file names from an `ftp dir` command; very often, the file sizes and time stamps are of interest too.

It might also be helpful to have a command like `sizeof tex/latex` to return the disk space requirements of a directory.

9 Acknowledgements

This work was carried out by the author with the support of facilities at the Department of Mathematics at the University of Utah.

My deepest thanks go to Eric Grosse and Jack Dongarra for having written the `netlib` system and published a paper about it [1]. I also want to thank Eric Grosse for making the `netlib` software available to TUG for modifications to create `tuglib`, and for keeping a record of everyone who has received `netlib` so they can get bug fixes.

Thanks also go to the TUG Board of Directors for helping in the testing of `tuglib`, and to Don Hosek and Joachim Lammarsch for many useful conversations and electronic exchanges.

Finally, we owe an enormous debt to the people who support and develop the Internet and make it freely available to a worldwide community connecting hundreds of thousands of machines, and perhaps over a million users. Without their foresight, many collaborative efforts the world over would be effectively impossible.

References

[1] Jack Dongarra, Eric Grosse, 1987. *Distribution of Mathematical Software via Electronic Mail*. CACM, 30(5), 403–407.