NETLIB INTERNAL DATA FLOW

ERIC GROSSE*

Abstract. The netlib repository consists of a dozen cooperating machines, with most coordination done by cron tasks and daemons. This internal netlib memo describes the flow of files and auxiliary information. It aims to help local authors appreciate which files they may edit directly and which are derived automatically. It may be of interest to administrators of large, replicated repositories.

1. Introduction. In the beginning, netlib consisted merely of a file tree, manually created indices, and one program to reply to email. As the years have gone by, the mechanism has had to grow more complicated[3]. To cope with firewall Internet security, we have to maintain copies on internal and external machines; to spread the administrative work, we have to coordinate updates on several sites worldwide; to support a broad community, we have to build the indexing and other auxiliary files needed by email, ftp, http, and searching. A small collection of scripts running hourly or nightly achieve all this, in a manner resistant to network outages, disk crashes, and human errors.

The system runs itself pretty well, and is fairly simple considering what it must cope with. But learning how it works by discovering and reading through shell scripts scattered around the world would be daunting. This little memo contains data flow diagrams and pointers that should assist anyone interested in enhancing the system or, heaven forfend, recovering from a disaster.

2. Beating one's head against the firewall. Local editors and authors need to have the master copy of netlib reside in a file system that they can directly edit, which in my environment means a Unix system inside the firewall. This inside machine, named pyxis, runs the email reply service directly. (See [1] for a description of the firewall and why it exists.)

A copy of the files (preferably in compressed form) must reside on a second machine outside the firewall, in order to serve ftp and http users. In principle, the firewall machines could run ftp and http servers that access pyxis' file system, but we do not have that facility. The inescapable minimum is some means of running, via cron, a small checksum program on the outside machine and a way of transferring files from the inside to the outside.

We use the Plan9 operating system[5] for the outside machine. Since it straddles the firewall, copying files is trivial, and we gain Plan9's substantial advantage of convenient nightly backup. More specifically, we copy files, compressed, from pyxis to the Plan9 file server, named bootes, and run ftp and http servers on the outside cpu server, named 9net. For various reasons (such as long file names in some parts of netlib), the master copy can not reside on Plan9. Some of the mirroring issues would arise anyway, to keep compressed and uncompressed files in sync.

Figure 1 charts the flow of files from pyxis to bootes and the auxiliary data that help determine what files need copying. In this and following figures, ellipses denote processes, open rectangles denote databases, and arrows denote temporary files or pipelines. Databases with no incoming arrows are updated manually; the rest are maintained automatically. Table 1 lists the times and machines on which these update processes execute.

* AT&T Bell Laboratories, Murray Hill NJ 07974 USA. `ehg@research.att.com`.

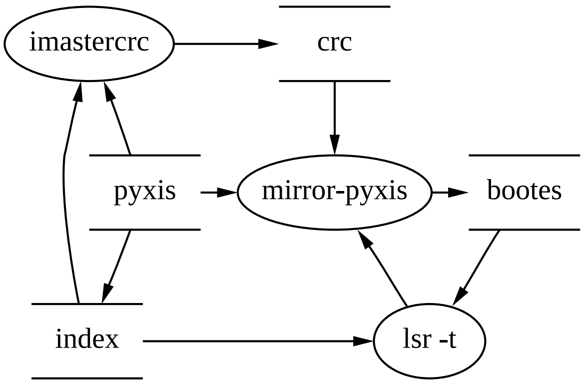

FIG. 1. *The master copy of netlib files reside on pyxis, a Unix machine inside the firewall; slave copies reside in compressed form on the Plan9 file server, bootes, and are made visible to the outside world by netlib.att.com, a Plan9 cpu server outside the firewall.*

The basic strategy is to compute, locally on pyxis and on bootes, lists of checksums for each file. Comparison of checksums establishes what files to copy, and a table of file name suffix rules governs what to compress. For more details on this mechanism, see [4].

A complication is that some netlib directories are restricted to AT&T internal distribution only, because of licensing or business issues. The list of directories to mirror is deduced from the index files, covered in the next section, which separately describe public and restricted directories.

3. Generating auxiliary files. The ultimate source of information about netlib files resides in `index` files in each directory, maintained by the person responsible for that directory. The syntax of these files was optimized for manual updates and for searching, but has proven to be quickly adaptable to new uses. A nightly process combines all the index files into a global keyword search database.

Another nightly process constructs index.html files, needed to support World Wide Web access[2], by a simple reformatting of the index files. The only subtlety is that index.html files must refer to compressed files, while index files describe the uncompressed tree. This is accomplished by following the same file name suffix table described above. Netlib may contain other HTML files besides the index.html tree, and these need similar adjustment for compression during the mirroring to bootes. Finally, because many existing Web browsers are deficient in handling compressed input and because many (but by no means all) HTML files are small, bootes holds HTML files in both compressed and uncompressed form.

A small example may make this clearer. Imagine we have the files

```
rg.f
index
example/driver.f
example/eigen.gif
example/index
```

on pyxis; then after mirroring there will be files

```
rg.f.Z
index.Z
index.html
```

```
index.html.Z
example/driver.f.Z
example/eigen.gif
example/index.Z
example/index.html
example/index.html.Z
```

on bootes. The two Fortran files `rg.f` and `driver.f` are compressed during mirroring to bootes; the image file is not (because the file name suffix tables note that GIF already is a compressed format). Again, the only subtlety is with the index files. On pyxis, `index` describes `rg.f` and `doc.html`; `example/index` describes `example/driver.f` and `example/eigen.gif`. The corresponding files `index.Z` and `example/index.Z` on bootes are simply compressed copies of the files on pyxis. Remote sites mirror `index` by copying `index.Z` and uncompressing, just the way they mirror `rg.f`. The file `index.html` on bootes is mostly a copy of `index`, but is decorated by formatting commands to be easier reading in a Web broswer and is augmented by links `href=rg.f.Z` and `href=example/index.html`. (Note the link to the *uncompressed* HTML file.) Finally, `index.html.Z` is a compressed copy of `index.html`, but with the last link rewritten as `href=example/index.html.Z`. Thus .html links to .f.Z and .html, while .html.Z links to .f.Z and .html.Z. Remote sites such as netlib.no create `index.html` files themselves, in accord with their local conventions.

The observant reader will notice a potential problem. Suppose someone creates a help file `doc.html` on pyxis with the link `href=rg.f`. Then `doc.html` on bootes needs the link `href=rg.f.Z`. The file `doc.html.Z` on bootes can be just a compressed copy of `doc.html` on bootes. So far, so good. But what about a remote site that wants to mirror `doc.html`? If netlib.no retrieves `doc.html` (or, equivalently and more likely, retrieves `doc.html.Z` and uncompresses) and chooses to offer all files in uncompressed form from its ftp and http servers, then netlib.no would need to rewrite the link `href=rg.f.Z` into `href=rg.f`. Fortunately, we do not have such a `doc.html` in netlib yet, at least not in the replicated part, and we may optimistically hope that the Web infrastructure will soon be able to deal transparently with compression, and much of this complication can fade away. (I shudder at the possibility that it will prove cheaper to double network capacity than to fix the software architecture!)

TABLE 1

Several cron tasks are responsible for copying files among the various repositories and generating auxiliary files.

time	machine	process
2:52	pyxis	mirror-utah; bibexpand
3:31	pyxis	`cat */index > DEFAULT`
3:34	pyxis	`cat */.master > master/index`
3:35, 10:30	pyxis	imastercrc
4:08, 11:08	helix	lsr; mirror-pyxis; index → index.html
4:20	pyxis	ls -lR /n/bootes
4:30	cs.utk	imastercrc
≈8:30	pyxis, cs.utk	mirror NJ↔TN
21:30	netlib.no	mirror; imastercrc

The top level `master/index` is not created manually but rather by concatenating `.master` files in each top directory. In this way, the editorial work for each directory stays localized.

Bibnet files, which are mirrored into pyxis from Utah and thence to other netlib sites, are also combined nightly into a keyword search database. Individual index files in the bibnet/authors and bibnet/bib directories are automatically created, since the file name itself provides sufficient information in this restricted category.

Two additional keyword search databases, for approximation and for computational geometry, are also rebuilt as needed. These search databases are moderately large (on the order of a megabyte) with frequent small, scattered changes. We store these on a special file system /n/bootesother rather than the normal Plan9 file system, to avoid wasting optical platter space (since these files could easily be recreated anyway).

In order to accommodate popular tools like archie and ftpmirror, another process constructs an updated `ls-lR` listing. This is, strictly speaking, redundant because of the checksum listings but it is a convenience for some outsiders.

REFERENCES

[1] S. M. Bellovin and W. R. Cheswick, *Firewalls and Internet Security*, Addison-Wesley, 1994.

[2] T. Berners-Lee and D. W. Connolly, *HyperText Markup Language specification - 2.0*, tech. rep., IETF, 1994. work in progress.

[3] S. Browne, J. Dongarra, S. Green, E. Grosse, K. Moore, T. Rowan, and R. Wade, *Netlib services and resources (revised)*, Tech. Rep. CS-94-222, University of Tennessee, August 1994.

[4] E. Grosse, *Repository mirroring*, ACM Trans. Math. Software, (1994). accepted.

[5] R. Pike, D. Presotto, K. Thompson, and H. Trickey, *Plan 9 from Bell Labs*, in Proceedings of the Summer 1990 UKUUG Conference, London, July 1990, pp. 1–9.