Installing and Operating "C News" Network News Software

Geoff Collyer

Software Tool & Die
Brookline, MA

Henry Spencer

Zoology Computer Systems
University of Toronto

ABSTRACT

How to set up and maintain the netnews software.

Section 1: **Routine Installation**

Chapter 1.1: **Installing "C News" Network News Software**

Introduction

Network news (or *netnews* for short) consists of a collection of messages formatted similarly to ARPAnet mail (see DARPA RFC 1036 for details), widely spread. The logical network, imposed on top of various real networks, formed by the set of all interconnected sites exchanging network news is called "Usenet" and was formed in 1979, radiating out from Duke University. Netnews is propagated between cooperating machines by a flooding algorithm, with some loop prevention heuristics: each machine sends its neighbours news that the neighbours have (probably) not yet seen.

Flow of netnews between machines may be achieved by use of any network (or other medium) which can transmit an arbitrary stream of (at least 7-bit, preferably 8-bit) ASCII code, unmodified. If a network cannot transmit ASCII intact (e.g. BITNET), it is possible to encode netnews before transmission across the network and decode it upon reception from the network. Since one cannot be certain that one's network neighbours will be up and reachable at all times, outgoing netnews must be queued, at least in the case of network trouble. To date, at least these networks, protocols and media have been used to transmit netnews correctly: UUCP, RS232, NNTP, Ethernet®, the ARPA Internet, Datakit®, ACSnet, magnetic tape, SMTP, and BITNET, though at least the last two require some form of encapsulation to avoid corruption of articles; SMTP because some common implementations get the newline-CRLF mappings wrong, thus throwing off byte counts in batches, and BITNET because of its Procrustean chopping, expanding, mapping, bashing and smashing of all data sent through it (sending lines of 80 or fewer characters of letters of either case and digits and plus and minus appears to be safe).

Netnews arrives via some network, which causes a command to be executed upon arrival (e.g. *rnews*). *rnews* typically spools the inbound netnews for later processing. Eventually (often within the hour or at the end of the business day), the input queue is run and the netnews is stored locally, typically under */usr/spool/news*, and queued for transmission to netnews neighbours. Once stored on disk, netnews may be read by any of a collection of unprivileged news readers, including *cat* (1). *Expire* is run typically each night to remove old netnews from disk.

C News was originally written to provide a much faster and smaller, more robust, reliable and correct implementation of netnews software than B 2.11 news. (There was once a B 3.0 news under development; it seems to be still-born.)

News System Overview

News is stored under some directory, referred to as NEWSARTS, which is typically */usr/spool/news* . Each newsgroup is stored in its own directory, whose name is formed by replacing all dots in the newsgroup name with slashes. Each article is stored in a file with a numeric name. There are pseudo-newsgroups called *junk* and *control* which respectively hold articles that your subscription list in the *sys* file permitted but which aren't in your *active* file, and control messages. There are also directories called *in.coming* , *out.going* , and *out.master* ; they hold incoming news, outgoing news batch files and outgoing master batch files. *in.coming* contains temporary files with various names and spooled input files with numeric names. *out.going* contains one directory per outgoing news feed (more for ihave/sendme sites).

Indices and related data are stored in another directory, referred to as NEWSCTL, usually */usr/lib/news* . Some of the interesting files here include *active* , *active.times* , *explist* , *history* , *sys* and *bin/config* . There are also log files kept here, notably *log* and *errlog* , which need to be aged or trimmed frequently (our standard *cron* entries do this automatically). *newsgroups* contains a list of group names and descriptions, one per line.

C News Components.

Rnews invokes the input subsystem, which spools incoming netnews in its original form, as received (usually from *uucp* or *NNTP*), typically in compressed batches. Periodically, the input queue should be run by invoking *newsrun* , thus uncompressing any compressed input and feeding it to *relaynews* .

inews is a front-end for *relaynews* which implements much of the per-site policy on news posting. Local postings are given to *inews* .

Relaynews files incoming netnews as articles on disk and initiates spooled transmission to other machines, often by simply writing the names of the disk files containing the articles on the ends of 'batch' files, which are in turn read by batching programs. Quite a bit of policy from RFC 1036 is embedded in *relaynews* . Actually, *relaynews* writes 'master batch files', which the batch file exploder, *explode* , turns into normal batch files. *explode* should be run periodically from *cron* or *newsrun* .

The output *batcher* reads lists of file names and generates news batches (see RFC 1036 or *news* (5) for the format) of prescribed sizes and queues the batches for transmission to other sites. The batcher is run asynchronously with *relaynews* , typically once an hour outside of business hours. NNTP sites will also need to run an NNTP transmitter periodically (e.g. *nntpxmit* or *nntplink*).

Expire is generally run once per night to remove from disk news articles older than a few days. *Expire* can use different expiry criterion for different newsgroups and can archive articles instead of removing them. *Expire* also runs asynchronously with *relaynews* .

Some people start their news processes from *newsxd* rather than *cron* ; we don't, and so can't offer much advice.

There are many news readers. C News comes with a limited news reader (*readnews* by Michael Rourke) sufficient to replace long-winded */etc/motd* s but you will want a heavy-duty news reader if you plan to read more than local news groups. We recommend *rn* by Larry Wall or *nn* by Kim Storm, available from your netnews neighbours or your nearby **comp.sources.unix** archive site. There are many others: *trn* and *vnews* are two. News can be read from the local disk, via a network file system such as NFS, or via NNTP (soon to be NNRP).

Preparation for Installation

Netnews consumes a lot of disk space and often a lot of transmission time. Here are some relevant measurements regarding a full netnews feed as of the time of writing (February 1992), taken from *news.lists*: a day's netnews is about 30MB and 10,000 articles in 3,000 newsgroups. Groups cover every imaginable topic, and there are for-pay hierarchies such as **clari**, which provide the UPI news wire. Years ago, sites often kept 14 days of netnews on disk, but now many sites keep news for 3 to 5 days, thus allowing for the occasional long weekend. Thus a full news feed expired after 4 days will consume about 120MB. Some people feel that news volume is doubling roughly every 12 months. If this is true, we can expect volume to increase by a factor of 10 in about 3 years to 300MB per day or 1.2GB for 4 days. It is thus wise planning to set aside a lot of disk space for netnews. There are two ways to cope with ever-increasing volumes of netnews: refuse to accept more newsgroups, or expire news after shorter intervals on disk. A current full feed takes about 16 hours to transmit at 9600 bps (assuming only 500 bytes per second actually sent), so for dial-up connections, 2400 baud and below is right out. In 4 years, a 56kbps ISDN channel should be saturdated around the clock by a full feed. One clearly wants the fastest networks one can afford.

Clearly, transmitting a full news feed is a non-trivial commitment of resources, so you may have some difficulty in finding a site willing to supply one. Such a site may in turn expect you to feed yet other sites. You will need to agree with your prospective netnews neighbour(s) upon transfer media, protocols and networks. You can always pay for a news feed from service providers such as UUNET and PSI.

Before proceeding to install C News, you should read this document through to the end, probably read the companion document *The Interface Between C News And The Outside World*, and possibly read selected items in this guide and the manual pages.

You will need to assign a user id and group id to netnews (often new ids called "news"); initialise these directories: NEWSCTL (typically **/usr/lib/news**), NEWSBIN (**/usr/lib/newsbin**), and NEWSARTS (**/usr/spool/news**); and install each subsystem (each subdirectory of NEWSBIN contains the programs for a given subsystem). NEWSCTL and NEWSARTS are logically one subtree, defining a news data base, but are split for backward compatibility with older news software, particularly old news readers. NEWSBIN contains programs and shell scripts which might be common amongst machines sharing a common architecture (e.g. Sun 3's); NEWSCTL**/bin** may override these. The goal is to install the subsystems, integrate them into a working news system, and configure the news system to communicate with other news systems.

There are a few key files that must exist before any serious attempt may be made to operate the news software. They all live in NEWSCTL. **active** is the list of newsgroups that this site knows (is willing to accept or individually reject), and must be owned by the NEWS userid (the userid that owns NEWSBIN**/relay/relaynews**, typically *news*). You will probably want to get your initial **active** file from your upstream feed and edit it to suit the set of groups you wish to receive. Be sure to make the second field more than five digits wide, by adding leading zeroes (ten digits is a conservative width). **sys** defines the newsgroups that this site is willing to accept and describes how they are to be forwarded to its neighbours. **Note:** Your **sys** file is required, as a cost of membership in Usenet, to be public information. It will be sent to anyone who requests it via a *sendsys* control message. Don't put secrets in your **sys** file. **server** contains the hostname of your file server, if you have multiple machines sharing news via a network file system. **whoami** similarly contains the name by which a cluster of machines sharing news is to be known for purposes of news. **mailname** is optional and contains the full (possibly dotted) name by which your cluster is known for purposes of mail. **organisation** (or **organization** if you insist) contains the name of your organisation, which will be copied into the **Organization:** header of articles posted locally, by default. **mailpaths** defines mail routes to machines which contain aliases for postings to moderated newsgroups. **log**, **errlog**, **history**, **history.dir**, and **history.pag** must exist and be owned by the NEWS userid. **explist** describes local expiration policy; N.B.: the order of lines is significant and articles with **Expires:** headers can interfere: read *expire*(8) for the full story. Brian Reid's *arbitron* script can be run to get some idea of what groups your users actually read. *expire* can be made to archive articles before removing them. Tentative versions of all these files are built by the installation procedures, but it is quite likely that you'll have to edit some of them.

C News Installation

Proceed to the **conf** directory of the distribution. There is a major shell file here, named **build**, that will interrogate you at length and construct shell files to do the real work. You may need to do ‘‘chmod +x build’’ before running it, to make it executable.

You will probably need your system’s manuals on hand to answer **build**’s questions. Another terminal (or another window, on a bitmap display) would also be useful. You’d best be prepared to take notes, also, as **build** will occasionally suggest that something be checked when you’re done.

Build itself does not alter any files or perform any installation chores: all it does is create shell files in the **conf** directory. If you already know something about news software, or are merely suspicious that **build** hasn’t done everything right, you should probably read the shell files before running them. There are four of them: **doit.root**, **doit.bin**, **doit.news**, and **again.root**.

Doit.root sets up the major directories for news, and sets their ownerships correctly. It typically will have to be run as *root* to have adequate permissions for all of this. It is brief.

Doit.bin does most of the work of building and installing the programs. It should be run as the user who owns the distribution directories and will own the executable programs under NEWSBIN. **doit.bin -i** suppresses installation. **doit.bin** adds creates prototype additions to *crontab* and */etc/rc* which run *newsrun* , *sendbatches* , *doexpire* , *newsdaily* and *newswatch* periodically, and clear out the news system at boot time, respectively.

Doit.news does some other small chores and installs control files. If any of the control files already exist, it will complain and refuse to overwrite them, as a safety precaution. It should be run as the owner of the news files. Since many of the files it is installing are built by **doit.bin**, that should be run first.

Finally, **again.root** tends to ownership and permission changes on a few programs that need to run set-userid. It requires the ability to change ownerships and to set permissions on the files afterwards, which usually means running it as *root*. It too is brief.

There are undoubtedly strange systems out there that **build** and friends are not smart enough to cope with. In such cases it will be necessary to edit the shell files before running them, or to use them as guides and do the work by hand. In particular, systems that require strange options in **Makefile**s will need to have those inserted by hand.

If you want to test pieces of C News without installing them, some (not all) of the subsystems have a ‘‘make r’’ feature that runs a regression test. Note: almost all of these require that NEWSCTL**/bin/config**, or its local equivalent, be in place already so that shell files can find out what PATH (etc.) they should be using.

Note that it is easy to build a test news system which is completely independent of other existing news systems on the same machine, or to build one which shares its NEWSBIN with another C news system, or shares input of articles (e.g. running an old B news system and a new C news system off the same stream of input until you are confident that your new C news system is operating to your satisfaction). See *subst* (1) for the mechanism which permits quickly changing all the places that know the location of NEWSCTL**/bin/config.** After that, edit this news system’s NEWSCTL**/bin/config** and things should be set up for separate existence.

Local policy

You may want to edit the *newgroup* and *rmgroup* scripts to change group creation and deletion policy. By default, all *newgroup* control messages are honoured, but *rmgroup* messages merely cause mail to be sent to NEWSMASTER. Some sites prefer to not honour *newgroup* messages automatically, which can cause articles to be filed in the *junk* pseudo-group until the group is created by hand. Very few sites honour *rmgroup* messages automatically, since one is then at the mercy of pranksters. Groups can be created and deleted manually with the *addgroup* and *delgroup* commands.

First News

When you arrange to get a news feed from your neighbor, you should also ask him to send you the current set of articles in the newsgroup **news.announce.newusers**. Several of these are very important reading for people who are new to the net.

Unusual Systems

We believe that C News runs fine on 16-bit machines, but it hasn't been tested very thoroughly on them lately. It will not perform quite as well with the more limited space.

Machines with very old compilers can be a headache. There are some hooks in **h/news.h** for doing without ''void'' and ''unsigned long'', two particular problem areas, but they have to be arranged manually; **build** does not know about them.

Some very old systems cannot do *setuid(geteuid())*, which makes it impossible for a daemon to make directories and get the ownership right. We provide a small program, **setnewsids**, to run setuid-root. **Relaynews** knows about it and invokes it if *setuid(geteuid())* fails; it then sets permissions correctly and re-invokes **relaynews**. The code is short enough to be read and understood in full, so that the suspicious system administrator can be sure that this setuid-root program is not going to do something awful.

Problems

In general, you should consult *Troubleshooting C News* if you have problems, and then *Known Porting Problems With C News* if that isn't sufficient. Serious errors in relaying are reported in NEWSCTL**/errlog** and will be mailed to NEWSMASTER eventually. C News generally communicates with the news administrator (NEWSMASTER) by mail. One common complaint is that there is a space shortage and incoming news has been discarded; the usual fix for this is to run *doexpire* immediately. One could modify *newsrun* to start a *doexpire* when space gets low.

It may be necessary to cancel a message posted in error by a user who can't figure out how to cancel it himself. Posting a message like this:

```
inews -h <<'!'
Newsgroups: what.ever
Distribution: where.ever
Control: cancel <user's-article's-message-id>
Subject: cancel <user's-article's-message-id>

text.
!
```

should do the job.

You may see complaints in NEWSCTL**/log** that articles have been rejected for bad headers. These will usually be from other sites, as *inews* will correct most such errors for your users. The usual errors are (a) text immediately following the colon after the header keyword, with no intervening space, (b) a new line immediately following the colon, or (c) no blank line separating the message header from the message body.

Routine Maintenance

You can add an outgoing news feed by adding a line to your NEWSCTL**/sys** file for the new site and creating its batch directory (NEWSARTS**/out.going/**site).

As news volume grows, you may need to ask your upstream feed to cut back on the groups it sends you, both to conserve network bandwidth and disk space. You will likely also need to either cut back the length of time you keep news on disk before expiry, or get in the habit of buying new disks periodically.

See the article *How to set up the mailpaths file* by Gene Spafford in *news.lists* for advice on same.

You can turn off news processing by running NEWSBIN**/input/newsrunning off** and restore it by running NEWSBIN**/input/newsrunning on**. Neither takes effect instantly; you have to wait for news processing to stop and wait for *newsrun* to be run for it to resume. You might want to turn off news processing before locking the news system to perform lengthy work on it; turning off processing both reduces the time until you get the lock, and reduces disk load due to lock contention while you have the lock.

You will get mail due to a *checkgroups* control message periodically, with a subject of *Problems with your active file*; don't worry about it. Some day *checkgroups* will get fixed and then you can worry.

You will probably want to read the *news.admin*, *news.software.b*, *news.lists* and possibly *news.groups* newsgroups to keep current.

Gatewaying mail to news and vice versa is not yet a solved problem. The most severe problem here is loop prevention. If you reject articles that lack valid RFC 1036 message-ids, just running RFC 822 mail messages into *inews -h* should suffice to gateway into news. Gatewaying news back to mail should just require a NEWSCTL/**sys** file entry like

mail:sci.space/all::mail victims

Sample build run

```
:; cd conf
:; ./build
This interactive command will build shell files named doit.root,
doit.bin, doit.news, and again.root to do all the work.  It will not
actually do anything itself, so feel free to abort and start again.

You probably need your system manuals handy.

When a question is asked in the form 'How are you [okay]? ', the
answer in brackets is what you will get if you just hit RETURN.
(If you want give an empty string as the answer, type a single
'-' instead.)

Do you want to use your previous answers as defaults [no]?

C News wants to keep most of its files under a uid which preferably
should be all its own.  Its programs, however, can and probably should
be owned by another user, typically the same one who owns most of the
rest of the system.  (Note that on a system running NFS, any program
not owned by "root" is a gaping security hole.)
What user id should be used for news files [news]?
What group id should be used for news files [news]?
What user id should be used for news programs [bin]?
What group id should be used for news programs [bin]?
Do the C News sources belong to bin [yes]?

It would appear that your system is among the victims of the
4.4BSD / SVR4 directory reorganization, with (e.g.) shared
data in /usr/share.  Is this correct [yes]?
This will affect where C News directories go.  We recommend
making the directories wherever they have to go and then making
symbolic links to them under the standard names that are used
as defaults in the following questions.  Should such links
be made [yes]?
Our 4.4ish friends suggest putting articles in /var/spool/news
and control files in /usr/share/news, with programs left where
they are in /usr/lib/newsbin.

C News lives primarily under three directories:  one for articles (and
incoming and outgoing spooling), one for control files, and one for
programs.
Where should articles live [/usr/spool/news]?
Where should control files live [/usr/lib/news]?
Where should programs live [/usr/lib/newsbin]?

C News by default assumes that all normal Unix programs can be
found in /bin or /usr/bin.  This is naive, especially on
Berkeley-derived systems where some standard programs
inexplicably moved to /usr/ucb.
It appears that some standard programs live in /usr/ucb
on your system.  Is that right [yes]?
Should /usr/ucb be searched after /bin and /usr/bin
(as opposed to before) [yes]?
```

**Is there any other directory which should be searched
to find standard programs on your system [no]?**

**C News normally uses a umask of 002, turning off only the others-write
bit in the permissions of files used. (The correspondence between bits
and number is: rwx = 421, so turning off group-write bits and all
others-access bits would be a mask of 027, for example.) Usually
a umask of 002 or 022 is appropriate.
What umask should C News use [002]?**

**C News wants to mail some forms of trouble reports to an administrator.
You probably want to make this a system mailbox, rather than that of a
specific user, so you won't have to change the software when you get a
new administrator.
Where should C News mail trouble reports [usenet]?**

**The shell files that are everywhere in C News want to pick up their
configuration parameters (mostly, the last few questions you have
answered) from a file at a known location; this is very hard to avoid
unless you play tricks with environment variables (see documentation).
Where should the shell configuration file be
located [/usr/lib/news/bin/config]?**

**What is the full pathname of the chown command [/etc/chown]?
Can I say '/etc/chown news.news file' to change both the user id
and group id of a file [yes]?**

**building doit.root...
done**

C News has libraries for several kinds of Unix:
bsd42 4.2BSD and successors
usg AT&T System V
v7 Version 7 (4.1BSD is pretty close, ditto Xenix)
v8 Version 8, aka Eighth Edition
Which best describes your system [v7]? bsd42

**C News has libraries for small address spaces (16 bits) and big
ones (preferably 32 bits, but anything rather bigger than 16).
Which best describes your system [big]?**

**Systems vary in whether certain library functions and system calls
are present. C News contains reasonably-portable versions of the
possibly-missing library functions, and fake versions of the
possibly-missing system calls, but it needs to know which are missing.
Does your system have fsync() [yes]?
Does your system have getopt() [yes]?
Does your system have memcpy() [yes]?
Does your system have memcmp() [yes]?
Does your system have memchr() [yes]?
Does your system have memset() [yes]?
Does your system have mkdir() [yes]?
Does your system have putenv() [yes]?
Does your system have strchr() [yes]?**

Does your system have strrchr() [yes]?
Does your system have strpbrk() [yes]?
Does your system have strspn() [yes]?
Does your system have strcspn() [yes]?
Does your system have strtok() [yes]?
Does your system have symlink() [yes]?
Does your system have strerror() [yes]? no

The news system uses a database package, typically the old "dbm" library from Version 7 or a lookalike, as an indexing system. We supply a version of the "dbz" library, which is faster than "dbm", uses much less disk space, and is program-compatible (although it is *not* file-compatible, so anything else using the database [notably NNTP, if applicable] has to be relinked with it). Dbz is usually preferable to dbm, barring major backward-compatibility problems. Do you want to use our "dbz" library [yes]?

Many systems, notably older ones, have implementations of the Standard I/O library ("stdio") in which fgets, fputs, fread, and fwrite are quite slow. We supply versions of these functions which are faster than those in any stdio we know; they are compatible with most old AT&T-derived stdios. (They tend not to work on modern System V, but the modern System V stdio is respectably fast.) They can be a major performance win for C News. There is a fairly thorough compatibility check run after the library is built; as far as we know, if the test works, the functions do (even on SunOS 4.0).
Do you want to use our fast stdio library [yes]?
Beware that the compatibility check will work best if the output of doit.bin is NOT redirected into a file or a pipe.

The strchr() function is usually slower than in-line C code when small strings are involved, unless your compiler is very clever and can generate in-line code for strchr(). Is your compiler that good (okay to guess) [no]?

Modern Unixes can generally use the setuid() system call to set the real and effective user ids to the current effective user id. In old Unixes, only "root" can change the real user id. This causes various problems for C News. C News provides a small program named "setnewsids" to run setuserid-root; all it does is change user and group ids and then execute C News "relaynews". It is needed only on uncooperative systems. Relaynews invokes it automatically if needed (and it then invokes relaynews in return). Can this system do setuid(geteuid()) to change the real uid/gid [yes]?

Some systems have header files that others lack, and C News is prepared to fake missing ones.
Does your system have an ANSI-C-conforming <string.h> [yes]? no
Does your system have <sys/timeb.h> [yes]?

Very old Unix systems needed the order of object modules in a library chosen very carefully. V7 introduced "ranlib" which removes the need for this. Recent System Vs have had the same facility built into "ar"

(look for the "symdef" feature in the "ar" manual page) so "ranlib" is not needed. Does your system use ranlib [no]? yes

Historically the C compiler is named "cc", but this is not true on some systems, and on others there are several different C compilers. What is the name of the C compiler to be used [cc]?

Historically the only normal compilation option needed for most programs is -O, but again compilers, especially newer ones, differ. (NOTE: many 386 compilers miscompile dbz if -O is used!) What options should be given to the compiler [-O]?

The final linking ("ld") step of compiling might need an option, such as -n or -i, to produce the preferred form of executable file. On most modern systems the default is right. What options, if any, should be given for linking []?

On unusual systems it may be necessary to link C News programs with libraries other than the usual C library. These can be specified as either full pathnames or -l... options. What libraries, in addition to the one(s) picked up automatically by the compiler, should be used when linking C News []?

Does your system have a "hostname" command [yes]?

C News tries to limit the backlog of news batches spooled up for transmission to a site, to control use of disk space. To do this, it needs to be able to determine the length of the queue of news batches for a particular site. This is UUCP-version-dependent. There is a good chance that you will have to customize the "queuelen" program. C News knows about several versions:

hdb	**Honey DanBer, aka Basic Networking Utilities**
sub	**old uucp with subdirectories (e.g. /usr/spool/uucp/C.)**
old	**very old uucp, no subdirectories**
pre	**prehistoric uucp, no subdirectories, no -g option on uux**
null	**don’t run uucp or don’t care about queue lengths**

Which one is most appropriate [hdb]? sub
Beware -- test "queuelen" to make sure it works.

C News often wants to ask how much disk space is available. The format of output from the "df" command unfortunately varies a lot, as does the availability of a system call to get the same information. C News knows about several different versions (the first three are preferred):

statfs	**system with standard statfs() (SunOS, 4.4BSD, not System V)**
ustat	**system with ustat() (most System Vs)**
ultrix	**DEC Ultrix with DEC’s own bizarre statfs()**
bsd	**4.2/4.3BSD**
sysv	**old System Vs**
xenix	**some (all?) Xenixes; some System Vs, e.g. Microport, HP?**
sgi	**Silicon Graphics Iris systems**
v7	**plain old style: no headers or fluff, just name and number**
null	**don’t know or don’t care how much space is available**

Which one is most appropriate [bsd]? statfs

Some "df" commands, especially on old systems, must be given the name of a device. Modern ones can be given any directory name and the system handles the details of figuring out what device is meant. A few will take a directory only if it is the "top" of a filesystem. Will "df" accept any directory name as an argument [yes]?

Are you planning to use expire to archive news on disk [no]?

Are you particularly short of disk space [no]?
You may want to inspect "spacefor" to make sure its defaults for things like desired free space are appropriate for your system, although the defaults are fairly conservative.

It is very difficult to do anything useful with incoming news when there is no space for it. Normally, C News simply discards it and mails a trouble report. On a single-user system, it may be better to just have the news reception stall until more space becomes available. Warning: this may stall processing of other incoming traffic, e.g. mail, as well, and the queue of unprocessed traffic may well grow until your disk fills up. Should news reception stall if space gets short [no]?

News processing is much more efficient when done in bulk, so C News normally just saves incoming news and processes it once an hour. If you have ample resources and are wildly impatient to make news available the instant it arrives, that is expensive but possible. Do you want immediate processing [no]? yes

Are you running C News on a group of machines hooked together with NFS, with articles filed on one "server" machine [no]? yes
What is the "hostname" name of the server [newsie]? world

Several programs need to know an overall name for the system news is being run on, where "system" may include multiple machines if they share a common set of control files and articles; this is used in article headers and related places. For uucp sites, this usually should be the uucp name. It is VITAL that you and your neighboring sites agree on this name -- if their news systems know you by a different name, or even a slightly-different variation of the same overall name, there will be trouble. What is the name of the overall system for news purposes [nowhere]? world

The "From:" lines of news postings, on the other hand, should carry a mailing address, which in particular should be a domain address for sites that have one. What is the mailing-address name of this system, preferably a domain address [world.uucp]? world.std.com

What is the name of the organization, for insertion into articles posted from here [Godcorp]? The World @ Software Tool & Die

Manual pages are normally stored in a tree structure under /usr/man. Local practices vary a great deal, however, and System V has also introduced some bizarre distortions into this once-simple structure. What is the top-level manual-page directory [/usr/man]?

C News adds manual pages to chapters 1 (programs), 5 (files), and 8 (administrative programs). These chapter numbers have changed in some variants of Unix. Also, originally pages from chapter 5 (for example) were stored in /usr/man/man5. This has also changed in some variants. Has your system made such changes [no]?

The "rnews" and "cunbatch" commands (which are identical, the latter being purely for backward compatibility with seriously-old systems) have to be installed somewhere where uucp can find them to execute them. It is not normally necessary for users to be able to run them, so they need not go in the directories searched for normal commands... although uucp often searches only those directories. What directory should "rnews" and "cunbatch" go in [/bin]?

Our "postnews", "readnews", and "checknews" are included mostly for completeness. They are very simple and crude compared to the user interface many users are accustomed to. As far as we know, B News (or other) versions should run fine with C News. If you are already running such user-interface software, you may not want to change. Do you want to install our user-interface programs [yes]?

The "inews", "postnews", "readnews", and "checknews" command(s) should go in one of the directories searched for normal commands, so users can run them without special arrangements. What directory should these commands go in [/bin]?

For replies to control messages, C News invokes "mail" (typically /bin/mail unless you make special arrangements) with either an Internet-style "@" address or a uucp-style "!" address. Internet style is probably better... if your mailer supports it at all. Will "mail" handle "@" addresses [no]? yes

Postnews can supply a default newsgroup, to aid naive users in getting this right for simple postings. What should the default newsgroup be [none]? wstd.general

Postnews can supply a default distribution, to limit news to a local area unless the user specifically changes it. This is probably wise. What should the default postnews distribution be [world]? wstd

Readnews has a default subscription list, for users who have not specified what newsgroups they wish to see. What groups should be in that list (comma-separated with no spaces, please, as they would be in a .newsrc) [general]? wstd.general

For administrative use, readnews has one newsgroup that users must subscribe to, even if they ask not to. What group should that be [general]? wstd.general

The ihave/sendme protocol, although marginally useful in some cases, is a security hole -- it lets another site ask for any article by Message-ID, and if your Message-IDs are predictable enough (which C News's generally are not, mind you), that site can get any article

currently on your system. Do you have any newsgroups containing confidential or proprietary material [no]?

building doit.bin...
done

building doit.news...
done

building again.root...
done

saving defaults...
done

You should now run doit.root as root, doit.bin as bin, doit.news as news, and again.root as root, in that order. (This assumes that the source directories are owned by bin. If you need to do installation work by hand, run 'doit.bin -i' as the owner; this will create the programs but won't install them.) (It is not necessary to log in as these users; use of 'su' suffices.) Finally, you will want to add the contents of 'cron', or something similar, to your cron's work-to-be-done file(s), and the contents of 'rc', or something similar, to /etc/rc or whatever your system executes when booting.

"make gclean" will clean up everything afterwards. "make lclean" does a less drastic cleanup affecting only the library directories. "make spotless" does "make gclean" and also removes the doit files.

Good luck and happy news reading.
:; ^D

Chapter 1.2: **News Flow**

The flow of news articles through C news is as follows: a batch is received by some network service and passed to *rnews*, which invokes *$NEWSBIN/input/newsspool* to queue the batch in **$NEWSARTS/in.coming**; eventually *$NEWSBIN/input/newsrun* will be run by *cron*, *newsrun* uncompresses any files in **in.coming** and either hands the batch to *$NEWBIN/relay/relaynews* locally or invokes *relaynews* over a network connection on its file server.

relaynews writes the name of each spool file onto the ends of batch files for neighbours. Eventually *cron* runs *$NEWSBIN/batch/sendbatches*, which batches up the outgoing articles and transmits them over some network.

Chapter 1.3: The Interface Between C News And The Outside World

Intro and Generalities

C News relates to the ‘‘outside world’’, the system it is installed on, in a number of ways. This document attempts to enumerate them and explain what goes on.

In general, C News attempts to rely only on ‘‘common basic Unix’’, and in particular it is not particularly BSD-specific or System-V-specific. Specifically, it makes no use of ornate locking mechanisms, silly interprocess-communication schemes, peculiar networking primitives, extensions to C, or other annoyances.

Directories And Userids

Most of the components of C News live in */usr/lib/newsbin*, with control files in */usr/lib/news* and spooling areas (for current news, inbound traffic, and outbound traffic) in */usr/spool/news*. See the document *Directory Layout and PATH in C News* for elaboration on the structure, *Files in /usr/lib/news (aka NEWSCTL)* for a summary of control files, and *Configuration Mechanisms in C News* for how to change the locations of the directories.

There is an extensive subdirectory structure under */usr/lib/newsbin*, with only a few heavily-used utilities in the top-level directory. In the following, programs not explicitly described as going elsewhere are all under there somewhere.

C News's spool areas and major control files need to be owned by a specific userid, normally *news*. (We believe that nothing except some of the Makefiles knows this name.) We suggest that this userid should not own anything else on the system. The programs in */usr/lib/newsbin* can be owned by *bin* (or anyone else) except for those that need to be setuid. (On our systems the non-setuid ones are owned by *bin*.)

Unix Utilities

C News requires a fairly complete Unix or equivalent. (We take no position on whether 4BSD, or System V, is Unix or not; our private opinion is that neither truly deserves the name any more, although we occasionally change our minds about which is less deserving.) (Note also that ‘‘Unix’’ is used here as an abbreviation for **The UNIX Operating System**®, a registered trademark of AT&T; we acquired our bad habits and incorrect capitalization from Unix [sic] documentation supplied by the Bell System in the mid-1970s.)

In particular, C News relies heavily on shell files. ‘‘Shell’’ here means, of course, the standard shell, written by Steve Bourne. If your */bin/sh* is not a Bourne shell or *very* good imitation, you're in trouble. Note, in particular, that some old versions of the Korn shell differ from the Bourne shell in some important details of quoting behavior, and we *know* this causes problems with C News. You need a standard shell.

To the best of our ability, all our shell files begin with ‘‘#! /bin/sh’’. As far as we know, nothing actually relies on being able to *exec* a shell file; that is, if ‘‘#! /bin/sh’’ is just a comment to your shell, that should be okay. If your shell misinterprets it as a request to run the C shell, however, you're in trouble. Running the following might help:

```
for f in `find . -type f -print`
do
	if test " `sed 1q $f`" = "#! /bin/sh"
	then
		ed $f <<'!'
1s/#!/: use/
w
!
	fi
done
```

If your shell doesn't know about "#" comments at all, again you're in trouble. We hope that no modern shell makes either of these mistakes.

We believe that none of our stuff relies on relatively modern shell features like shell functions or *getopts* (as distinct from *getopt*). If *test* and *echo* are not built-in commands in your shell, efficiency will suffer but everything should still run. It's possible that a few obscure things will break if your shell does not support "[" as a synonym for *test*, although we try to avoid this usage.

Within the shell files, C News makes heavy use of a wide variety of Unix utilities, notably *sed* and *awk*. Any *awk* should do; in particular, nothing needs the "new *awk*". Although we have tried to avoid it, it is possible that some things depend on *awk* recognizing "\t" inside strings, which very old *awk*s didn't.

If your Unix does not put all standard Unix programs in the standard directories—*/bin* and */usr/bin*—you will need to modify C News's default PATH to include whatever bizarre directories are involved. See the *Configuration Mechanisms* document for details. In particular, for some insane reason, 4BSD relocated a few standard utilities like *wc* to */usr/ucb*, which causes trouble (and not just to C News!). We simply put some links in */usr/bin* to fix this on our systems, and this is what we recommend you do, but if you can't, you'll have to change the PATH.

Several parts of C News rely on being able to send mail to an administrative account (the name of which can be chosen; see *Configuration Mechanisms*). The assumption is that a message, without any RFC822 headers, can be piped into */bin/mail* (or whatever *mail* is first in the PATH) invoked with a single argument which is the addressee, and be delivered to the addressee's mailbox intact, possibly embellished with headers. That is, with news's standard PATH, if

```
echo "relaynews: hi there Joe" | mail joe
```

does not result in the message "relaynews: hi there Joe" arriving in the mailbox "joe", you're going to have to fake it. (Note that some BSD mailers run into trouble with the colon in the example, interpreting such a line as a header.) See *Directory Layout* for insight on where to put a fake *mail* so that C News components will use it rather than */bin/mail*.

Similarly, several parts of C News rely on being able to invoke *hostname* without arguments, and have it return a single word which is the name of the CPU the code is running on. Again, you'll have to fake it if you don't have *hostname* or if it outputs something strange.

Input reception and output batching both want to use the *compress* data-compression program. *Compress* has been published on Usenet and is shipped with many Unix systems these days, but if you don't have it, we recommend that you get it from your neighbors or the *comp.sources.unix* archives. It is possible to configure C News so that it doesn't use *compress* to compress news being transmitted, but you don't want to. *Compress* is good and it is fast.

Libraries

C News is mostly self-contained as far as libraries go. It does need some things that are not yet universal, like a full set of string functions (e.g. *strtok*); we provide reasonably portable versions of these for places that lack them.

Networking

(We're talking here about networking in the sense of NFS and all those fun things, not *uucp* or TCP/IP.) C News is designed to run on systems which consist of a conglomeration of machines on a local network, with only one of them acting as news server. Be warned, though, that if you're doing this, you need to have *rsh* or some reasonably equivalent way of executing a program on another CPU. If you've only got one machine, just make sure you *don't* have a */usr/lib/news/server* file and forget the whole issue.

Highly System-Specific Things

There are two small utilities within C News that are inevitably highly system-specific and have a high probability of needing changes to match your system. Both normally live in */usr/lib/newsbin* proper.

Spacefor is invoked by various components of C News to find out how much disk space is available. The space margins in it are ones we find reasonable, but you may need to adjust them. On an old System V system in particular (one where *df* can't be applied to any directory name, but needs to be given a name in */dev*), you will also probably need to modify the locations to be *df*ed. You will also need to fix *spacefor* if your system's *df* yields results in some strange format or in some strange units. We believe that we get it right for stock 4BSD, many (but probably not all) System Vs, modern Irix, and real Unix (Version 7). Beware introducing major inefficiencies: *spacefor* is called a lot. Beware, also, that *spacefor* is not intended to permit routine operation using filesystems that are on the brink of being full; the limits it imposes are only approximate, and no attempt has been made to tune its clients to exploit every last available block.

Queuelen is invoked by the batcher to determine how long the current *uucp* queues are, so it can judge whether it should suspend batching of output to a given site. There is too much diversity in *uucp*s for us to try to do this for all possible versions. We think we get it right for the two most common flavors (HDB, aka BNU, and old subdirectory versions). Our versions measure queue length in batches, not bytes, but it would not be hard to change this: *queuelen*, the *batchparms* control file, and the *sendbatches* how-many-batches-should-I-try-to-add logic need to agree on the units of measurement, but (we think) nothing else cares.

It is just possible that you might have to modify *newshostname*, which also lives in */usr/lib/newsbin* itself. *Newshostname* delivers the name which should be applied to the whole system (not just the particular CPU) for news purposes. It tries to be fairly clever about different ways of getting the name, and in particular will take it out of */usr/lib/news/whoami* if that file exists, but if you're doing something odd on a strange system, changes may be needed.

Input

Input from the net via *uucp* (or equivalent) shows up as batches of news to be fed to *rnews* or its obsolete synonym *cunbatch*. These two programs normally live in */bin*, although nothing in the code knows about this and they can go elsewhere (one of our systems does this). Both are simple front ends that invoke *input/newsspool* to store the batch, while taking precautions to report trouble and not to overflow disks. Neither *rnews* nor *cunbatch* needs to be setuid.

Input via NNTP over the Internet (or equivalent) uses rather different machinery but ends up creating a saved batch in much the same way as *input/newsspool* does.

Input/newsspool is a small C program that saves a batch, writing into a file in */usr/spool/news/in.coming*. It must be able to create files there, and *input/newsrun* (see below) needs to be able to read them and delete them. This gets a little tricky because *newsspool* will usually be run by *uuxqt* as userid *uucp* (or something like that), not as *news*, which *newsrun* needs to run as. The recommended solution is to have *newsspool* owned by *news* and setuid. An alternative is to give the *in.coming* directory the userid of *news* and the groupid of *uucp*, or vice versa, and set permissions so that either can access it. One of our systems ran that way for a while.

To actually process incoming news, *input/newsrun* gets invoked to decompress the spooled batches and feed them to *relay/relaynews* (see below). There is an option for *newsspool* to invoke *newsrun* when a batch is spooled, but a (usually) preferable method is to have *cron* invoke *newsrun* once an hour. *Newsrun* does its own locking to prevent multiple occurrences running simultaneously. There is a related program, *input/newsrunning*, that can be used to set or clear a flag that stops *newsrun*; this may be a useful tactic if *newsrun* should not run at certain times. Both *newsrun* and *newsrunning* must be run as *news*.

When a user posts news, he (or his news reader) does it by feeding the article to */bin/inews*. In C News, *inews* is a complex shell file that attends to preliminaries and then invokes *relay/relaynews*. *Inews* does not need to be setuid (indeed, we make no use of setuid shell files at all, since they are grossly insecure). *Relaynews* is the heart of C News, the program that actually pulls batches apart and places articles into the database. If users are to be able to post news, *relaynews* should be owned by *news* and setuid.

News Readers

C News is fully compatible with B News to any news-reader program that does not inspect the middle field of */usr/lib/news/history* too closely. Standard B News news readers work fine. We supply a simple news reader (written by, and included with permission of, Michael Rourke) as a naive-user replacement for the B News *readnews*. More complex programs are preferable for serious news enthusiasts. We recommend Larry Wall's *rn* or Kim Storm's *nn* (which we use, unmodified), but there are others.

Output

Relay/relaynews normally queues up news for transmission to other systems by appending article names and sizes to batch files in subdirectories under */usr/spool/news/out.going*. These are then processed by *batch/sendbatches*, which should be run regularly, as *news*, by *cron*. *Sendbatches* can be configured to use a variety of transmission mechanisms, the usual one being *uux*.

Expiry And Related

News articles are removed, possibly with archiving to an archive area, by the expiry subsystem. *Expire/doexpire* should be invoked now and then, as *news*, by *cron*. We suggest nightly. *Doexpire* actually invokes *expire/expire* to do the dirty work.

C News *expire* does not have an option to rebuild the */usr/lib/news/history* file from scratch, since that has nothing to do with expiry. To rebuild *history*, e.g. if it has been destroyed, use *expire/mkhistory*.

Some news readers (notably the NNTP-based ones) need to have the third field of */usr/lib/news/active* updated occasionally to show the lowest article number still present in each newsgroup. Frankly, we think such news readers (and NNTP) simply need to be fixed. C News *expire* does not do this updating. For those who use such news readers, however, *expire/upact* will do such an update. It should be run as *news*. A much faster, but somewhat less portable, C implementation is supplied as *expire/updatemin*.

Reboots and Administration

If the system crashes, things like locks must be cleaned out if C News is to function properly after reboot. */etc/rc*, or equivalent, should run *maint/newsboot* during reboot, as *news*.

Certain log files can grow without bounds if not renamed/removed now and then. We recommend running *maint/newsdaily* once a day. It tends to logs, keeping a generation or two for use in trouble tracking, and also sends mail to the news administrator in the event that something funny has happened.

In general, C News does not attempt to break locks, on the philosophy that a stale lock may mean something is badly wrong. (See *Locking in C News* for details on locking methods.) The various programs will either give up, to return later, or wait patiently for the lock to go away. If one doesn't keep an eye on things, a problem of some kind can hang up the news system for quite a while. Running *maint/newswatch* once in a while—we recommend a few times a day—will alert administrators to signs of trouble. Except on grossly slow systems, C News locks should never hang around for any great length of time.

Chapter 1.4:
Directory Layout and PATH in C News

Intro

C News is constrained by historical compatibility with B News, but we also want to provide more flexibility for local news administrators. Accordingly, our directory organization is a little different from that of B News. We also make a lot of use of subordinate programs rather than lumping everything into a few giant lumps, and this means we need a notion of search paths.

See also ‘‘Configuration Mechanisms in C News’’, which talks about how to alter the defaults for these paths and such.

Directory organization

We retain the notion that a single directory (usually /usr/spool/news) is the top of the news-article database. We also use subdirectories of this directory, using names including ‘.’ to ensure that they cannot collide with newsgroup names, to hold incoming batches and outgoing batch control files. One can debate whether this is the right place for these activities, but in practice /usr/spool/news tends to be where people want to put potentially-big traffic-handling directories, and it’s not worth providing for separate variation of the location.

We do split the former /usr/lib/news into two, however. We reserve /usr/lib/news itself for control files that are logically part of the database. (It would make sense to put those under some subdirectory of /usr/spool/news, but that would break a lot of programs that think these files live in /usr/lib/news.) The programs—those which don’t need to be directly executable by users or *uucp*—live in /usr/lib/newsbin. Actually, they usually live in subdirectories thereof, with each significant subsystem having its own subdirectory to keep the individual directories manageable in complexity. There are occasional general-purpose utilities at the top level; there aren’t enough of them to be worth a separate directory.

PATH

In general, things are organized to permit sharing of /usr/lib/newsbin among multiple databases. It doesn’t make sense to share /usr/lib/news among multiple databases, as much of the stuff that lives there is logically part of the database.

There is a possibility that an individual database will want to override specific decisions made by the programs, i.e. will want its own version of some programs. Accordingly, provision is made for a ‘bin’ directory under /usr/lib/news. News software should set its PATH to something on the order of

/usr/lib/news/bin:/usr/lib/newsbin/xxx:/usr/lib/newsbin:/bin:/usr/bin

(See the ‘‘Configuration Mechanisms’’ document for how this should *actually* be written, to facilitate configuration changes and local customization.) That is, first look in the database’s bin directory for overrides, then in some subdirectory of /usr/lib/newsbin for the subsystem’s programs, then in newsbin itself for news-wide utilities, then in the standard system directories for standard Unix utilities.

Chapter 1.5: **Configuration Mechanisms in C News**

Intro

There is an overall problem with news stuff in that some pathnames and such are site-specific. In C News this hits both shell files and C programs, the former a particular inconvenience because they are not compiled and hence can't easily pick it up from a library at compile time. It's easy to edit shell files, but there are too many of them for this to be very convenient.

Several mechanisms are needed to solve the entire problem..

Configuration Parameters

Although there are many things that theoretically could be site-specific, there are only a few that really crop up constantly, are highly likely to change, and have to be known to running code. Here's a tentative list, with internal names and common defaults:

NEWSARTS	pathname of the article database	/usr/spool/news
NEWSCTL	pathname of the control-file directory	/usr/lib/news
NEWSBIN	pathname of the program directory	/usr/lib/newsbin
NEWSPATH	shell PATH setting for news	/bin:/usr/bin
NEWSUMASK	default umask for file creation	002
NEWSMASTER	where to send mail about trouble	usenet
NEWSCONFIG	see "Subst" section	/usr/lib/news/bin/config

Environment Variables

All C News programs that care about configuration parameters are expected to look at environment variables with the same names as the parameters, and to use the environment values to override internal defaults. This is primarily aimed at testing and special purposes rather than production use, as there is a fundamental problem: the environment is insecure. The environment variables are nevertheless very useful.

There is a need for a canned interface to the environment variables, to simplify their use. There is also a requirement for centrally-set defaults for normal production use. These requirements are addressed differently for C and shell programs.

C Interface

Libcnews provides a set of inquiry functions:

char *fullartfile(char *name);
: Returns a full pathname for the NEWSARTS-relative file *name*; if *name* is NULL, returns the value of NEWSARTS.

char *ctlfile(char *name);
: Returns a full pathname for the NEWSCTL-relative file *name*; if *name* is NULL, returns the value of NEWSCTL.

char *binfile(char *name);
: Returns a full pathname for the NEWSBIN-relative file *name*; if *name* is NULL, returns the value of NEWSBIN.

char *newspath(void);
: Returns the value of the NEWSPATH parameter.

int newsumask(void);
: Returns the value of the NEWSUMASK parameter.

char *newsmaster(void);
: Returns the value of NEWSMASTER.

Functions which return string values return pointers to areas which they may later re-use, so the values should be copied if they are to persist past later calls. There is also a function *artfile* which returns a relative pathname, as a special-purpose optimization, and a function *cd* which does a *chdir* and notes the location for future use by *artfile*.

The first inquiry function to be invoked automatically initializes their internal housekeeping. Against the possibility of malicious setting of environment variables, if any environment variables are in fact present to override default values, and at least one value in fact differs from the default, this initialization includes a call to a function the user must supply:

void unprivileged(char *reason);
: Drop any special powers that should not be available to a normal user program, e.g. those obtained by set-uid bit. *Reason* is the name of the first environment variable that caused trouble.

Programs that do not use the set-uid bit can normally have a null implementation of *unprivileged*, but to encourage thought about the matter this is *not* provided as a default.

Suitable external declarations for all these functions can be found in the include file *config.h* in the C News include directory.

Shell interface

All shell files that want to know configuration parameters should execute NEWSCTL/bin/config using the '.' command; it sets shell variables with the same names as the parameters. Note that it does not export these variables, so subordinate shell files need to pick up *config* themselves. This is to preserve the property that subordinate programs see environment variables set only if the user set them.

Subst

The alert mind will have noticed a circularity in the previous section: it's not possible to find NEWSCTL/bin/config until one knows where NEWSCTL is. There are also occasional annoyances in that documentation and such wants to know the local defaults, and can't get them from either the C or shell interface. Hence the use of *subst*.

Subst does substitutions into source files (including shell files) in such a way that it is not necessary to maintain two versions of the file. In the top-level directory of the C News sources are four files: *subst* itself, the *substitutions* file defining the values of the configuration parameters, and *subst.hs* and *subst.gc*, which are lists of files (relative to the top-level source directory) that need substitutions done. (The use of two 'list' files reflects C News's dual authorship.) See the *subst*(1) manual page for the gory details of the syntax. The only C program affected is the configuration-inquiry part of the library, but all shell files that need any of the configuration parameters need to be in one of the lists.

Program startup

As a result of all this, a C program which needs to know a configuration parameter simply calls the appropriate inquiry function, and is prepared for a return call to *unprivileged*. A shell program has to do a bit more work; it should start with something on the order of:

```
#! /bin/sh
# foobar – does foo, bar, and bletch

# =()<. ${NEWSCONFIG–@<NEWSCONFIG>@}>()=
. ${NEWSCONFIG–/usr/lib/news/bin/config}

PATH=$NEWSCTL/bin:$NEWSBIN/xxx:$NEWSBIN:$NEWSPATH ; export PATH
umask $NEWSUMASK
```

(A prototype for this is in conf/proto.sh.) (See an accompanying document for the logic of the PATH setting.) The NEWSCONFIG configuration parameter specifies the location of the configurer shell program, which is then executed to set up the rest of the parameters. The reason for doing it this way is to ensure that new parameters can be added without having to change every shell file.

Section 2:
Further Reading

Chapter 2.1:
Files in /usr/lib/news (aka NEWSCTL)

The following files can appear in */usr/lib/news* (or wherever the site's "NEWSCTL" directory—see *Directory Layout and PATH in C News*—is). There may be others for the sake of local news readers, etc., but these are the ones C News knows about.

active	Major control file: list of newsgroups recognized at this site, including current maximum and minimum article numbers and moderation status. Updated by *relaynews* and *upact*.
active.old	Previous *active*. Created by *upact*.
active.times	List of created newsgroups and when they were created, aimed at making it possible for news readers to be smarter about knowing when a group is new. Only supported by some standard news readers.
batchlog	Latest batcher log, created by *sendbatches*, showing backlogs.
batchlog.o*	Previous batcher logs.
batchparms	Control file for *sendbatches*, specifying how to feed other sites.
bin	Master override directory for programs, searched before any other by all C News software. Normally contains only the *config* shell file specifying where to find everything else.
errlog	Error log from *relaynews*.
errlog.o*	Previous *errlog*s.
explist	Control file for *expire*, specifying what gets expired, and when, and what archiving is done.
history	List of articles currently known at this site, with reception dates and pathnames. Updated by *relaynews* and *expire*.
history.dir	Part of the *dbm* index for *history*.
history.pag	Other part of the *dbm* index for *history*.
history.n	Temporary version of *history*, while *expire* is running; can be left behind if *expire* terminates abnormally.
history.n.dir	Part of the *dbm* index for *history.n*.
history.n.pag	Other part of the *dbm* index for *history.n*.
history.o	Previous *history*. Created by *expire*.
localgroups	List of local groups and descriptions, for use by *checkgroups* control-message handler (which uses it to decide what groups are legitimate even though the *checkgroups* message did not mention them). Each line is a group name, a tab, and a terse description of the group. The descriptions are read only by humans.
log	Log file from *relaynews*, reporting what was received and when.
log.o*	Previous *log*s.
mailname	Name of the site for purposes of mail, typically a domainized name. Used in building "From:" lines in newly-posted news. If not present, "*hostname*.**uucp**" is assumed.

mailpaths	Mailing routes for submissions to moderated newsgroups.
newsgroups	File created by *checkgroups* processing, with names and descriptions of newsgroups. For human reading only. Format identical to that of *localgroups*.
newsgroups.bac	Previous *newsgroups* file. Created by *checkgroups* processing.
organization	Name of the organization, for *inews*'s use in creating the *Organization:* header for a posted article.
postdefltdist	Default distribution (if any; default ‘‘world’’) for *postnews*.
postdefltgroup	Default newsgroup (if any; default is to insist on the user supplying one) for *postnews*.
replyusepath	This file should exist if and only if the local ‘‘mail’’ command is unable to handle ‘@’ addresses, meaning that replies to control messages (etc.) must use the ‘‘Path:’’ line instead of ‘‘From:’’. The contents are ignored.
server	Host name of the news server, where all operations (posting, etc.) should be done. If file does not exist, current host is assumed.
setnewsids	Set-uid-root program to get user and group ids correct for *relaynews*. Needed only if the system is old enough to refuse to do *setuid(geteuid())*. Included in NEWSCTL rather than NEWSBIN because its presence is a significant policy issue for the administrator, i.e. it is, sort of, a control file.
sys	Major control file specifying what groups are legitimate here, and what groups are fed to other sites.
watchtime	Last time *newswatch* was run.
whoami	Name of the system for news purposes, for *newshostname*. If file does not exist, other sources (*hostname* etc.) are consulted.

Chapter 2.2:
Log File Formats in C News

Introduction

The two main log files in C news are *NEWSCTL*/**log** and *NEWSCTL*/**errlog**, which are the standard output and standard error streams of *relaynews*. **errlog** should be empty; if not, something is seriously wrong, probably with your configuration or with unencapsulated news arriving mangled courtesy of some network.

Formats

errlog is a copy of any error messages written on standard error by *relaynews* or its auxiliaries (including control messages programs). There are no timestamps and the contents obey no special format. A non-empty **errlog** is a sign of trouble, which should be fixed promptly.

log is written in a stylised format, with whitespace separating the fields: date and time, to millisecond resolution though possibly not to millisecond accuracy; the machine that sent us this article; a single-character classification code; the article's message-id; and code-specific information. For example,

```
Jun 30 03:32:18.960 utgpu + <1312@sunset.MATH.UCLA.EDU> mailrus dptcdc me
Jun 30 03:32:19.600 utgpu j <1312@sunset.MATH.UCLA.EDU> junked due to groups ‘alt.drugs’
Jul  3 18:55:35.492 utstat s <470@lexicon.com> utzoo-real
Jul  3 18:55:35.912 utstat i <8907031824.AA09129@ucbvax.Berkeley.EDU> utzoo-send-ids
Jul  4 06:40:22.395 jarvis.csri.toronto.edu + <0541.AA0541@worsel> utgpu
Jul  4 06:40:22.595 jarvis.csri.toronto.edu + <0549.AA0549@worsel> utgpu
Jul  4 07:00:26.565 jarvis.csri.toronto.edu + <295@lancelot> utgpu
Jul  4 07:17:51.537 utgpu - <8258@saturn.ucsc.edu> duplicate
Jul  4 07:17:51.697 utgpu - <1675@neoucom.UUCP> duplicate
Jul  4 07:17:51.757 utgpu - <89Jul4.043358edt.10369@neat.ai.toronto.edu> duplicate
Jul  4 17:23:28.234 utgpu - <2537@quanta.eng.ohio-state.edu> no subscribed groups in ‘rec.arts.sf-lovers,rec.arts.startrek,rec.arts.drwho’
Jul  4 19:17:15.785 utgpu - <89Jul4.190330edt.5559@gpu.utcs.utoronto.ca> all groups ‘list.humanist’ excluded in active
```

The classification codes are:

code	article disposition	code-specific information
–	rejected	reason for rejection
+	accepted	list of sites to which this article was relayed
j	accepted but filed in the *junk* pseudo-newsgroup; preceding line will be **+**	reason for junking
i	generated in response to an *ihave* control message	list of sites to which this article was relayed
s	generated in response to a *sendme* control message	list of sites to which this article was relayed

Beware that control-message handlers inherit *relaynews*'s standard output, so if any of them natters on standard output (we believe none of ours do), the nattering will appear in **log**.

Chapter 2.3: **Annotated Bibliography on C News**

This is an informal list of books and papers that are of interest when running or looking at news systems in general and C News in particular. It might have been nice to include some of these things in the distribution, but it would have added considerable bulk to an already-large package.

Managing UUCP and Usenet, by Tim O'Reilly and Grace Todino, O'Reilly & Associates, 1989, ISBN 0-937175-48-X, inquiries to 'nuts@ora.com' or 'uunet!ora!nuts'. This is the Nutshell Handbook on UUCP and news. This latest edition covers C News as well as B News. It's not perfect, but until something better comes along, it is overwhelmingly the right place for a novice system administrator to start reading.

News Need Not Be Slow, by Geoff Collyer and Henry Spencer, in *Winter 1987 USENIX Technical Conference Proceedings*, The USENIX Association, 1987, no ISBN apparent, inquiries to 'office@usenix.org' or 'uunet!usenix!office'. This paper, although slightly dated, is the basic reference on how C News runs so much faster than B News. (We think, and others agree, that it's also a good paper on how to make things go fast in general.) It's also the closest thing so far to a general "philosophy of C News" paper, and gives a good—albeit somewhat selective—overview of the implementation.

Standard for Interchange of USENET Messages, by M. Horton and R. Adams, Internet RFC 1036, 1987, inquiries to 'nic@nic.ddn.mil' or anonymous FTP from nic.ddn.mil (pathname RFC:RFC1036.TXT). This is the current standard for Usenet article format and related issues. It is not complete, correct, or current, but it is nevertheless the basic document on the subject. (See our *notebook/rfcerrata* for some errata and one or two out-and-out disagreements.) Many issues are punted to RFC 822; see below.

Standard for the Format of ARPA Internet Text Messages, rev. by David H. Crocker, Internet RFC 822, 1982, inquiries to 'nic@nic.ddn.mil' or anonymous FTP from nic.ddn.mil (pathname RFC:RFC822.TXT). This lengthy and sometimes cryptic document defines the Internet mail format. The Usenet news format defined in RFC 1036 is a subversion (superset of a subset) of this, and legalistic debates over details of the format inevitably end up referring to 822. Not for the faint of heart or weak of stomach.

Requirements for Internet Hosts—Application and Support, ed. by R. Braden, Internet RFC 1123, 1989, inquiries to 'nic@nic.ddn.mil' or anonymous FTP from nic.ddn.mil (pathname RFC:RFC1123.TXT). This document updates and amends many older RFCs, notably including RFC 822 (and therefore RFC 1036).

news.software.b, various authors, Usenet newsgroup with ongoing discussion, inquiries to your local Usenet site(s). The forum for B-News-compatible software in general, including C News. The definitive source for up-to-date information, patches, bug reports, warning of forthcoming developments, debates, flame wars, and general chitchat on news systems in general and C News in particular. Read, and contributed to, by most authors of news software.

news.admin, various authors, Usenet newsgroup with ongoing discussion, inquiries to your local Usenet site(s). The forum for news administration in general, including that of both B and C News. Somewhat noisy at times, but a valuable source of information. Noteworthy for Gene Spafford's regular postings on topics such as "How To Construct The Mailpaths File".

news.announce.newusers, various authors, Usenet newsgroup with ongoing discussion, inquiries to your local Usenet site(s). (Moderated) newsgroup for regular postings of interest to new readers of news. Also quite relevant to novice system administrators.

Chapter 2.4:
Troubleshooting C News

Important First Step

Read the documentation.

Read the documentation!

READ THE DOCUMENTATION!!!

Reading documentation is a bit tedious, and it seems sort of peripheral when there's a problem crying out to be solved, but news processing is relatively complex and effective troubleshooting requires that you understand what's going on. The investment of time is worthwhile. In particular...

If you are having trouble getting C News installed, or it seems to be malfunctioning, you should read "Known Porting Problems With C News" carefully. Don't assume that your problem is not the same as one mentioned there just because you're on machine ABC and the one mentioned is on machine XYZ; some problems arise from generic causes that can occur on many systems.

If the software seems to be installed properly and appears to know what it's doing, but isn't doing what you want, careful reading of the manual pages is in order. We recommend particular attention to *news*(5), which documents the format of most of the control files in considerable detail.

General Approach

In general, your first priority should be to establish precisely what is going wrong and where. Read the documentation on how data flows within C News and how it interfaces to the rest of the system, and track down exactly where trouble is striking. If the software is sort of working but isn't doing things right, inspection of the *log* and *errlog* files in NEWSCTL often reveals what it thinks it's doing.

Frequently Reported Problems

It all seems to work, but it's very slow and the history.pag *file is enormous, many megabytes, much larger than the* history *file. By the way, this is on a 386.* See "Known Porting Problems With C News", specifically the section titled "386 Optimizer vs. dbz". (Note: at a site with a relatively small feed, it is normal for the *history.pag* file to be about half a megabyte for the first ten days, until *dbz* feels it has enough usage history to safely shrink the file.)

It works fine but the articles I post don't get sent to my neighbors. This is probably a mistake in your *sys* file. See *news*(5) and the sample *sys* file in the *conf* directory. Note in particular that an article is not sent to a site unless *both* its newsgroup(s) *and* its distribution match those specified in the *sys* line, and the *sys*-line distribution does *not* default to "all". The single commonest change needed to old *sys* files is to put "/all" on the ends of the newsgroup lists to specify transmission of all distributions.

It works fine for incoming news, but whenever I try to post an article myself I get a complaint about "renouncing setuid". There are two places in C News where the pathnames of things like the control-file directory are known. One is the "config" file, typically */usr/lib/news/bin/config*. The other is compiled into some of the programs. You *cannot* change one without also changing the other; this means that you basically cannot change either without rebuilding the software. The "renouncing setuid" message means that the two are inconsistent. Do not try to edit the config file without rebuilding the software; that doesn't work.

I'm seeing articles filed in surprising places and/or propagated (not propagated) to sites that I thought shouldn't (should) get them. The interaction of various policies in this area is *not* simple. It is almost certain that the software is doing what you told it to do. Close and careful reading of the *news*(5) and *relaynews*(8) manual pages will probably clear up the difference between what you told it to do and what you *thought* you told it to do.

I had trouble compiling your stdio *speedups, and/or they failed the compatibility test, but I used them anyway, and now I'm having bizarre problems.* If the speedups don't compile smoothly or failed the compatibility test, *DON'T USE THEM!* In this area, being "a little bit broken" is like being a little bit pregnant. As *build* told you, there are systems where the speedups do not work.

I'm having trouble compiling libc/datetok.c *with my ANSI C compiler.* We don't have ANSI C compilers handy for our own use, and there are a few troublesome areas where older compilers make difficulties and encourage errors. Until this particular bug gets fixed, a workaround is to move the definitions of *datetktbl* and *szdatetktbl* up to the beginning of the file, and delete the erroneous *extern* declarations.

I get a news feed from my neighbor, and it arrives fine, but my system sends him not just the things I post locally, but also everything I get from him. News-loop prevention is based on the **Path** header lines. The name your neighbor is known by in your *sys* file does not match the one he's inserting into **Path**, so your news system thinks he hasn't seen those articles yet. A common cause of this is that he's putting a full domain name in **Path** and you're using a short version in your *sys* file, or vice versa. The best fix is to agree on names. A workaround, often useful, is to alter the *sys* line to use the exclusion feature. Say he's putting ''abc.def.ghi'' in his Paths and your sys file knows him only as ''abc''. If there is some reason why you can't just agree on the name, start the *sys* line with ''abc/abc.def.ghi:'' rather than just ''abc:'', to tell the news system ''send abc only things that have not passed through abc.def.ghi''.

Will C News work if my article tree is spread over more than one filesystem? Yes, if your system supports symbolic links. Some tweaking is necessary: you will have to give *doexpire* the **–l** flag so it knows about the situation, and you will need to alter *spacefor* to check space on more than one filesystem. There is also a problem, currently, in that the *find* command used by *mkhistory* and *addmissing* will not follow symbolic links; a fix for this is coming.

How do I configure C News so that it will not automatically create any newsgroups just because some yo-yo on the net sends out a newgroup *message?* This is not provided as a configuration option at present. Most control messages are handled by shell files in NEWSBIN/*ctl*, and those can be edited to implement any local policies desired.

It ran fine for a while, but now relaynews *is complaining that it's unable to write the history file, saying "(File too large)".* This means you're on a System V or related system that implements the stupid *ulimit* feature, limiting the size of files, and the *ulimit* is too low. Note that while some sensible suppliers have their *login* raise the limit to a very high value, most of them seem to have forgotten to do the same for *cron*... and much of C News is run from *cron*.

I occasionally get a mail message saying a whole bunch of groups are invalid and I should delete them. What's going on? This is the result of a *checkgroups* control message being received. There is no clear specification for the contents of such a message, and the C News *checkgroups* code is known to be buggy. This will be fixed eventually. For now, ignore the mail.

The daily status reports from newsdaily *mention my own site as one sending bad headers! What's going on?* This is a bug in the logging code: when it wants to generate a log message for some condition that has no site name associated, it uses your own rather than something like '<no_name>'. This will be fixed eventually.

Section 3:
Specialised Subjects

Chapter 3.1:
B News to C News Transition Guide

Preface

We describe the changes in installation and administration from B News to C News. The intended audience is B News administrators switching to C News.

Introduction

C News was written by two people (Geoff Collyer and Henry Spencer) who were experienced B News administrators (since 1982). C News was intended to be a much faster and more correct B News with some of the things that irritated us in B News changed. C News is thus administratively similar to B News, though not identical. We have no personal experience running B News later than 2.10.1, and thus may have missed subtleties in later releases, notably 2.11.

The single largest visible change from B News is probably that we wanted to allow the possibility of machines of different architectures sharing a single news data base (*/usr/spool/news* plus the data files in */usr/lib/news* in B News terms) via network file systems while keeping separate directories of executables elsewhere. We did this by pushing the executables into a new directory, often called */usr/lib/newsbin*. We also took the opportunity to push most of the binaries into subdirectories to reduce the clutter at the top level, as the old B News */usr/lib/news* had become awfully cluttered.

The next most visible change is probably that we only compile in default values for directory names (and a few other configuration parameters), but they can be overridden at run-time via environment variables (subject to restrictions to prevent spoofing). *notebook/config* (relative to the C News source subtree root) describes the variables. In a nutshell, *NEWSCTL* is the equivalent of */usr/lib/news* when referring to control files, *NEWSBIN* is the equivalent of */usr/lib/news* when referring to executables (and is often */usr/lib/newsbin),* and *NEWSARTS* is the equivalent of */usr/spool/news*. This capability can be used to run B News and C News in parallel for testing, feeding them identical inputs. It can probably be used instead of the **IHCC** ifdefs in B News. It is also used for regression testing, when we run known input data through the software in test directories and verify that we get known results out; **make r** in a given source directory will generally invoke the relevant regression test and **make rs** in *conf* will invoke all of them.

Changes in control file formats

In general, *news* (5) now describes the file formats.

Sys adds a few optional features and withdraws some old and little-used ones. Many obsolete flags draw fatal diagnostics, notably **N**, the old, unbatched ihave/sendme flag. The other unsupported flags are **A** (convert outgoing messages to A News format; *relaynews* is not a protocol converter), **H** (append history entry to named file?; sorry), **S** (duplicate the work of the shell to avoid one process; just delete this flag), **M**, and **O** (multicast flags; obsoleted by the batcher, as far as we can tell). New flags are **f** (append spool file name and size in bytes to the named file) and **n** (append spool file name and Message-ID to the named file; equivalent to **FI** in B News). There have been some changes to flag semantics: **I** in B News invoked all the ihave/sendme machinery *and* wrote message-ids to the batch file; C News just writes the message-ids. See below for how to set up ihave/sendme feeds. In C News, **FI** and **I** are equivalent; use **n** if you want file names and message-ids. All file names written are relative to *NEWSARTS*.

B News allowed comments to be continued; this is at odds with all other UNIX† software, including

† UNIX is a trademark of Bell Laboratories.

C News. To comment-out an entry under C News, you need to prepend '**#**' before *each* line of the entry, not just the first.

There is now an optional *distributions* list after the newgroups pattern, separated by a slash. If the fourth *sys* field is a relative command name, *NEWSCTL/bin* and *NEWSBIN/relay* will be searched before the standard search path. If the fourth field is a relative file name, *NEWSARTS/out.going* will be prepended internally. If the fourth field is an empty file name, it will be replaced internally with *NEWSARTS/out.going/*system*/togo* where *system* is the first field of the entry. Note that although these batch files live under *NEWSARTS*, they are not news articles, and their parent directories will *not* be automatically created. The ihave/sendme kludges of B news have been expunged; one must say what one means. See the section below on ihave/sendme.

Active has four fields (there is vestigial code in some, but not all, programs for two or three fields) and any number of digits in the article-number fields (five is too few for the long term). Two new values are understood for the fourth (flags) field: **x**, meaning ''quietly discard articles for this group'', and **=**realgroup meaning ''file articles for this group under *realgroup* instead''. **=** is useful for coping with badly-run local newsgroups, often created from mailing lists. You must lock the news system with *locknews* before editing the active file.

History is in an extended B 2.10 news format: the second field consists of two subfields separated by a tilde: time received as an integer (a *time_t* in fact), and the value of the *Expires:* header, "-" if none. More subfields may appear in future. You must lock the news system with *locknews* before editing the history file, and run *dbz*(1) (found in *NEWSBIN*) afterward to rebuild the index files.

Log and *errlog* are only faintly similar to their B News counterparts. Error processing in general is due for an overhaul, which may further revise their formats. Currently, *errlog* is the unadorned standard error from *relaynews*(8) and any programs it runs. *Log* is a time-stamped log, down to the millisecond (where possible, offer void where prohibited by System V), consisting of one line per article, and always showing the name of the host that handed us this article and the Message-ID of this article. See *Log File Formats in C News* for details.

Locks are in the style of B 2.10: *NEWSCTL/LOCK* is the news system lock, but it never times out. One locks the news system by repeatedly attempting to link to the lock name until successful; one releases the lock by removing the lock name. There are other locks, all of which begin with the prefix *NEWSCTL/LOCK*. This scheme works well over network file systems, can be used from shell scripts and interactively, permits one to see trivially what locks are present, and is portable across UNIX variants. See *Locking in C News* for details.

'Missing' control files

Aliases does not exist, since we believe that header munging is to be avoided if at all possible (we do update **Path:**, regenerate **Xref:**, and delete some large obsolete headers [**Relay-Version:**, **Posting-Version:**, **Date-Received:**, **Received:**, **Posted:**, **Illegal-Object:**]). It is possible to file articles locally under different newsgroups by use of the *active* file **=** flag.

Notify is replaced by *NEWSMASTER* in *NEWSCTL/bin/config*.

Moderators, *recmail* and *recnews* are missing; we don't know what they do (yet).

Seq is gone; we don't need it.

New control files

Batchparms controls outbound batching. See *newsbatch*(8).

Explist controls expiry policies. The command-line options for *expire* are completely different from B News. See *expire*(8). The simplest possible *explist* is ''**all x 7 -**'' for 7-day expiry of all articles and history entries.

Mailname is the domain name of this machine for mail purposes; *whoami* is the news name of this machine (e.g. for **Path:** headers). If *replyusepath* exists, automated mail replies will use the **Path:** header for return addresses. If *server* exists, *inews* will *rsh* to the hostname found therein to run *relaynews*.

Setnewsids is an optional setuid-root program (in the wrong directory, it appears) to set user and group ids; it should be obsolete now, but instead POSIX and System V are making it necessary all too often.

ihave/sendme feeds

We didn't understand the implementation of ihave/sendme in B News for a long time and nearly didn't implement it, having a very dim view of ihave/sendme to start with, but we wanted to be RFC 1036 compliant. Upon discovering the tricks (including mandatory default values for various *sys* file fields) used internally by B News to implement ihave/sendme, there was immediate agreement that

(a) we should document how ihave/sendme works, and

(b) we should implement something more obvious and that would not mandate fixed values for fields that users might legitimately want to change from their default values.

We also see no use for the old unbatched ihave/sendme protocol: it is grossly wasteful of resources at today's traffic volumes and offers no clear advantage to sending lists of Message-IDs in each *ihave* or *sendme* message.

The result is that, in our second implementation, we have implemented batched ihave/sendme with *batchparms* specifying details of batching and transport. One typically needs a single *sys* entry to specify which articles are to be offered to the other system via an *ihave* message.

Note that any article requested by a *sendme* will be sent; there is no checking that the requesting site is permitted by its appropriate *sys* entry to receive this article. Such checking could be added, at some cost in performance. For now, disable ihave/sendme on systems with 'private' newsgroups (if that isn't an oxymoron).

See *Setting Up Netnews Feeds Using the Ihave/sendme Protocol* for gory details including sample *sys* files.

Differences in behaviour

The *checkgroups* control message is non-destructive; it merely mails its output to **NEWSMASTER** (e.g. **usenet**). The distributed *checkgroups* script has problems, which will be fixed some day. In the mean time, some people have reported that the latest B News *checkgroups* script works fine for them; you could try it.

There is an *Also-Control:* header recognised, of which *Supersedes:* is a special case.

Interactions with NNTP

We had erroneously assumed when writing C News that the NNTP situation was not too disastrous. Since we didn't then run NNTP ourselves, we benignly ignored NNTP. Only once we started running NNTP did we realise what a performance disaster it was. We are still working on improving the situation.

The **CNEWS** ifdef of NNTP as distributed by Stan Barber makes the situation tolerable (we supplied the original version of the code inside the ifdef) if not wonderful. In particular, it will happily accept duplicate articles, only to have *relaynews* throw them away. Paul Vixie's *msgidd* (and associated nntpd patch) is strongly recommended for sites that get more than one incoming NNTP feed; it eliminates the reception of duplicates. Sites not running *msgidd* or running the reference NNTP implementation without **CNEWS** defined are on their own; performance of the total news system is likely to be poor.

More work is called for. Several people have simultaneously invented the idea of a multi-threaded *nntpd* handling multiple inbound transfer sessions and that may be a worthwhile direction to pursue. (It permits cheap duplicate rejection with little lock contention and few processes.) It has the disadvantage of not exploiting parallelism on multiprocessor machines, which seem to be what the biggest news relay sites are becoming.

dbm & dbz

There are now several redistributable *dbm*(3) clones available. Before they appeared, we fixed and enhanced Jon Zeeff's *dbz* library; it is faster and has smaller files than any *dbm* or *dbm* clone, so we recommend using it for news. Whatever you elect to use, be sure to link NNTP and any other software that makes *history* file lookups with the same library that C News uses (picking up *libcnews.a* is usually simplest).

Migration to C News

A good first step is to read the documentation in the C News *doc* source directory. You will want to refer to the contents of the *man* directory. If you have trouble formatting any of this documentation, get the *awf* formatter and use it. *Awf* can be had from your nearby *comp.sources.unix* archive site, or by *uucp* or *ftp* from *uunet* (*uunet.uu.net*) as *comp.sources.unix/volume23/awf.Z*. For side-by-side comparisons of B and C News where this document may have missed something, one could compare the corresponding sections for B and C News in *Managing uucp and Usenet*, by Tim O'Reilly and Grace Todino, O'Reilly & Associates, 1989, ISBN 0-937175-48-X, inquiries to '**nuts@ora.com**' or '**uunet!ora!nuts**', or 800-338-NUTS (in California, 800-533-NUTS), FAX 707-829-0104. See *Annotated Bibliography on C News* for a fuller bibliography.

It is probably safest to create parallel C News trees, at least for the equivalent of */usr/lib/news*, then populate them and rename the B and C trees when you are satisfied that C News is installed and set up correctly. The C News installation procedure (*build*) will not overwrite existing control files in *NEWSCTL*, so copying your old *active*, *sys*, and possibly *history* files to *NEWSCTL* adapting them as necessary (see below), and then installing C News there normally should get you started.

One advantage of setting up a parallel tree for C News is that it is possible, though *build* doesn't know how, to run all incoming news into both news systems for a day or two to convince yourself that everything is working right. If you decide *not* to switch, you can just remove the C News tree. In any case, running both systems simultaneously avoids the desperate panic of having to cut everything over at once and get it right first time to avoid losing news; this just should not be necessary any more. The change is just to ensure that all the commands that used to invoke *rnews* (*inews*, *rnews*, *cunbatch* and probably a few others) now hand the incoming batch to both news systems. Something like this should suffice (and has worked in the past):

```
# parallel rnews
f=/tmp/rn$$
cat >$f
brnews <$f
crnews <$f
rm $f
```

Sys should only need to be scanned for unsupported flags (I think **S** can just be deleted without ill effect), and entries with **I** flags rewritten to write site**.wehave/togo** and *batchparms* updated to describe the ihave/sendme tango for this site. *Active* can optionally be edited to refile groups locally, to compensate for lack of *aliases*. C *expire* can read a correct B *history* file, but B News sometimes produced incorrectly-formatted history files. To be safe, and to pick up information that B News didn't store in the history file, run *mkhistory* to regenerate *history* and its index files. *Mkhistory* will take quite a while; it has to open every article in the news system, which can take hours on some machines.

The normal installation procedure is described in *Installing "C News" Network News Software* and really just involves running *conf/build*, answering its (many) questions, and following its instructions. (We intend that eventually *build* will be run once per site for all time (or until the hardware or OS change drastically) and that thereafter components may be built using *make*.) Do be careful to use the correct userid for each step of installation; doing the whole thing as *root* will result in incorrect file ownerships. Two major by-products of *build* are *libcnews.a* in the root of the source subtree and the directory *include*.

There are some differences in configurable options. All the length and size limits are gone. Essentially all the B News *makefile* options are now queried for by *build*. DFLTEXP and HISTEXP are now specified in *explist*. Only a trivial newsreader, a *readnews* replacement, is included with C News, so little of the reader configuration is relevant. NOTIFY is now *NEWSMASTER* and *build* queries for it. DFTXMIT and UXMIT are now CMDPFX and CMDSFX in *relay/sys.c* with no simple override (though an override is in the works); the default command, in case anyone still uses it, is **uux - -r -z** system**!rnews**. MANUALLY, NONEWGROUPS and UUPROG are replaced by policy in shell scripts; edit *relay/ctl/** to taste. BATCH is gone; the unbatcher is built into *relaynews*. OLD is gone (*relaynews* is not a protocol converter). DOXREFS is gone; **Xref:** is always generated for cross-posted articles. SENDMAIL and MMDF are gone; we just use *mail*. DEADTIME is gone; locks do not time out.

This table summarises the disposition of the remaining B News options. File names in the 'build file' column are relative to *NEWSCTL* and refer to files generated by *build*. A 'flag' in the 'note' column indicates that the presence or absence of the file *is* the flag. A 'gone' in the 'note' column indicates that the option has vanished. If the 'build file' column is empty, the option has just been absorbed into *build*.

B option	build file	note
SPOOLNEWS	rnews.newsrun	flag
INTERNET	replyusepath	flag
MYDOMAIN	mailname	
MYORG	organization	
UUNAME	whoami	
CHEAP		gone
NICENESS		gone
FASCIST		gone
ORGDISTRIB		gone
MULTICAST		gone
UNAME		
GHNAME		
BSD4_2		
BSD4_1C		
LOCKF		
HIDDENNET		
SMALL_ADDRESS_SPACE		

If you compile and install the software as two steps, it is simple, though time-consuming, to run regression tests for the major subsystems: **cd** into a source directory (e.g. *expire*, *batch*, *relay*) and type **make r**. If you get no complaints, the software is believed (after testing) to work correctly on your machine.

Sites with a lot of outgoing news feeds (over 60, say), should be aware that batch file writing will probably dominate *relaynews*'s elapsed time. We hope to fix this.

After installation, errors will generally be reported by mail (to *NEWSMASTER* [typically *usenet*]). To watch the progress of incoming news, tail *NEWSCTL/log* and *NEWSCTL/errlog*. *errlog* should be empty on a smoothly-running system.

Chapter 3.2: **Setting Up Netnews Feeds Using the Ihave/sendme Protocol**

Introduction

The ihave/sendme protocol is a means of conserving network bandwidth at the expense of some real-time delays in getting netnews. It pre-dates the NetNews Transfer Protocol (NNTP) for the TCP/IP protocol stack by several years and attempts much the same optimisation, but ihave/sendme is batched, unlike NNTP.

The above perhaps needs emphasizing: *ihave/sendme has nothing whatsoever to do with NNTP*. NNTP has ''ihave'' and ''sendme'' messages, which have somewhat similar functions to the ihave/sendme protocol (hence the similarity of names), but the implementations are completely unrelated. If you want to set up an NNTP feed, you are reading the wrong document.

Ihave/sendme is sketched in extremely vague terms in ARPA Internet RFC 1036 (nee 850), but the description therein is so lofty as to be useless as a protocol specification.

Into the Breach

The basic ihave/sendme strategy is for a site which has just received a new netnews article to send an *ihave* netnews control message containing the Message-ID of the new article to each of its ihave/sendme neighbours; the neighbour consults its netnews *history* file and if it has not seen the article, sends back a *sendme* control message containing the original Message-ID; upon receipt of the *sendme*, the first site will transmit the article via normal channels to the requesting neighbour. If you are getting exactly one news feed, ihave/sendme is of no benefit and merely slows down the reception of news. If you are getting multiple feeds, the added delay may not be worth the reduction in volume of news transferred (though sites being fed by long-distance telephone may disagree). In general, ihave/sendme should be a last resort, if only because it is more complicated to understand and set up than an ordinary news feed.

Due to the high volume of netnews, sending individual articles is always a performance disaster (see B News, NNTP and unbatched ihave/sendme), so the golden rule of netnews is ''Thou shalt batch thine articles.''. This makes the above sketch a little slower and more complicated: now individual articles are not sent, but batches of *ihave*, *sendme*, and netnews messages are transmitted, incurring further delay since netnews batchers are usually run only once per hour, say.

We will now walk through an example ihave/sendme set up between two C news sites (*utzoo* and *utstat*), with reference to the following *sys* file fragments and flow diagram. Most of the work is done by specialised batch preparers. *relaynews* merely responds to *ihave site* or *sendme site* control messages by writing the name of the file containing the control message onto site**.ihave/togo** or site**.sendme/togo**, respectively. This scheme owes some ideas to Root Boy Jim of UUNET.

utzoo's sys file

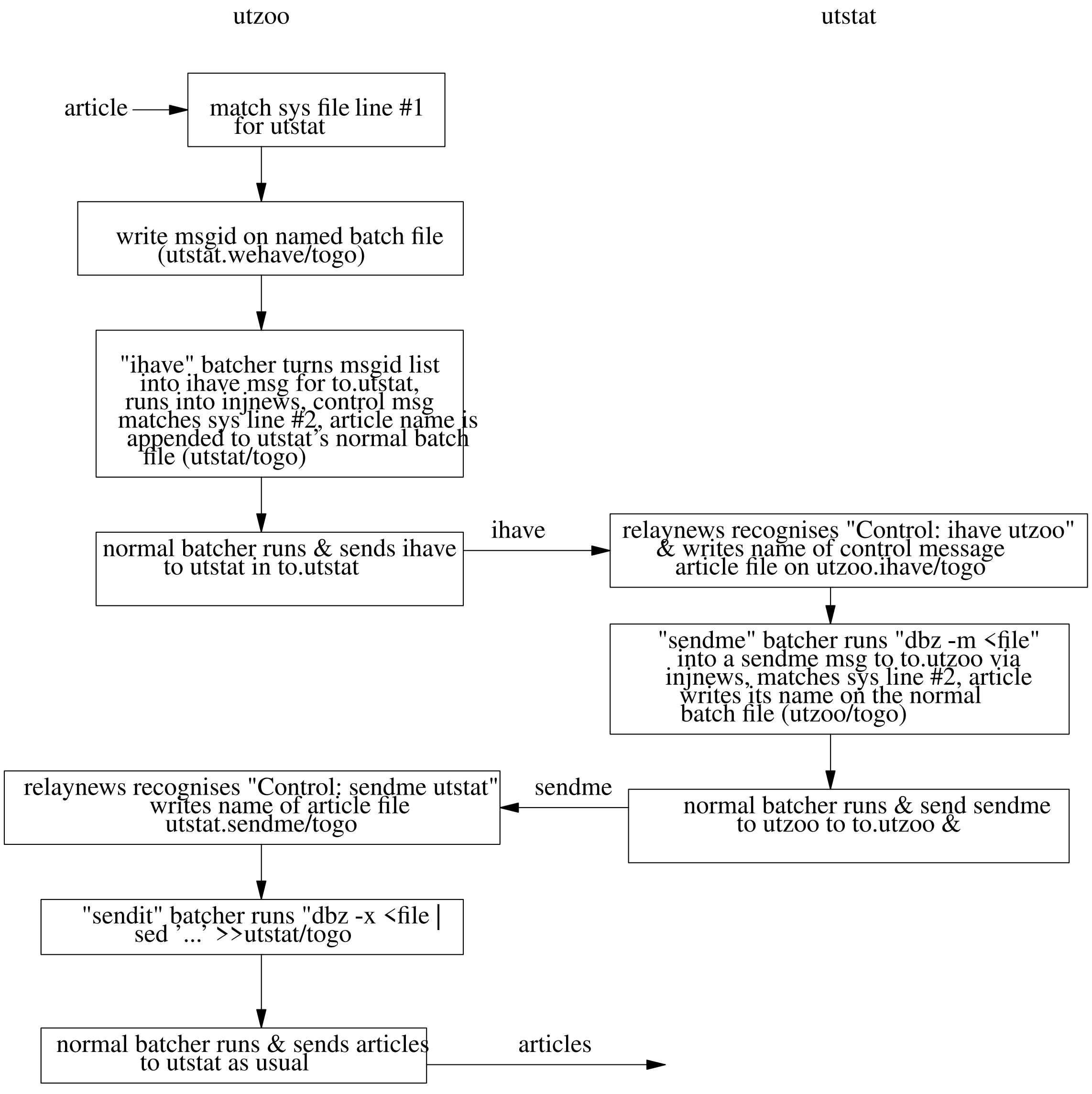
```
# Send ihave telling utstat what we have -- batcher turns the batch into a
# giant control message and posts it to "to.utstat".  (#1)
utstat.wehave/utstat:rec.music.synth,!to/all:I:
# actual transmission of control messages, via normal means (#2)
utstat:to.utstat/all:f:
```

utstat's sys file

```
# Send ihave telling utzoo what we have -- batcher turns the batch into a
# giant control message and posts it to "to.utzoo".  (#1)
utzoo.wehave/utzoo:rec.music.synth,!to/all:I:
# actual transmission of control messages, via normal means (#2)
utzoo:to.utzoo/all:f:
```

utzoo's batchparms file

```
# sample ihave/sendme setup
# site          size    queue   builder   muncher   sender
# ----          ----    -----   -------   -------   ------
utstat.wehave   40000   20      batchih   nocomp    viainews
utstat.ihave    40000   20      batchsm   nocomp    viainews
utstat.sendme   40000   20      batchra   nocomp    vianowhere
```



An Example

An article arrives on utzoo

A new article arrives on *utzoo* in newsgroup *rec.music.synth* and is matched by the first *sys* file line **on utzoo**, which writes the Message-ID of the article onto a batch file (*$NEWSARTS/out.going/utstat.wehave/togo*). There are some subtleties here: it is essential to add **,!to** to the subscription list if that list contains **all** or **to** to prevent looping messages; and it is always worthwhile to prevent leaking local newsgroups out one's full feeds, so if the subscription list contained **all** one would want to add something like **,!general** or **,!utstat**. Eventually, the batcher runs on the batch file.

The *batchih* batcher on *utzoo* prepares a batch of Message-IDs as an *ihave* control message to *utstat* and submits it to *inews −h* which matches *sys* line #2, and writes the file name of the article file containing the control message on the usual *utstat* batch file (*$NEWSARTS/out.going/utstat/togo*).

Eventually, the normal batcher will run on *utzoo* on the usual *utstat* batch file and will produce batches, including the *ihave* control message, and transmit them to *utstat* by means specified in *$NEWSCTL/batchparms*.

The resulting ihave message arrives on utstat

When the *ihave* control message arrives on *utstat*, *relaynews* will write the name of the file containing the control message on the *utzoo ihave* batch file (*$NEWSARTS/out.going/utzoo.ihave/togo*).

Eventually, the *batchsm* batcher will run on *utstat* on that batch file, which will produce a *sendme* control message and submit it to *inews −h* which matches *sys* line #2, and writes the name of the *sendme* control message on the usual *utzoo* batch file (*$NEWSARTS/out.going/utzoo/togo)*

Eventually, the normal batcher will run on *utstat* on the usual *utzoo* batch file, and will produce batches including the *sendme* and send them to *utzoo* by means specified in *$NEWSCTL/batchparms*.

The resulting sendme message arrives on utzoo

When the *sendme* control message arrives on *utzoo*, *relaynews* writes the name of the file containing the control message on the *sendme* batch file, and the *batchra* batcher eventually writes the file names of the requested articles (named by Message-ID in the *sendme* control message) on the usual *utstat* batch file.

Eventually, the normal batcher on *utzoo* will run and produce batches, including the articles requested by the *sendme*, and will send them to *utstat* by means specified in *$NEWSCTL/batchparms*.

Comparison with B News

There is a lot of activity involved in ihave/sendme, and there are at least five distinct channels and three transactions needed to send a batch of articles. B news ‘‘simplifies’’ the *sys* file by requiring certain *sys* file options (sic) to be set to fixed values, and by overloading madly. The resulting *sys* file is much more difficult to decypher, and if you should want to specify non-default options, you are out of luck.

Chapter 3.3:
Errors in RFC 1036

Introduction

RFC 1036, the standard *du jour* for the format of Usenet (netnews) messages contains significant errors, enumerated below. References are made to RFC 850, the previous netnews message format standard, and also to RFC 822, the mail message format standard (which, note, has been slightly amended by RFC 1123).

Header order insignificant

Between RFC 850 and RFC 1036, a sentence stating that the order of message headers is insignificant has fallen out of the standard. This may be a reflection of the reality that B 2.10 news did indeed care about the order of **From:** and **Path:**.

"Re:" is only three characters

One sees the contradiction "the four characters "Re:"" repeatedly; there should be a space after the colon.

cmsg incorrectly described

Similary, RFC 1036 claims that a **Subject:** prefix of "cmsg" will be interpreted as denoting a control message; the correct prefix is "cmsg " (including a space).

Xref is transmitted

RFC 1036 says that **Xref:** headers should not be transmitted, yet they are stored on disk as part of message headers, so they will be transmitted by both B and C news. The standard appears to be too strict.

cancels should propagate always

RFC 1036 says that *cancel* control messages should stop propagating if the receiving system is "unable to cancel the message as requested". It is not clear what this means, given that modern news systems hang onto cancellations for not-yet-seen articles in hopes of being able to cancel them in the future. B 2.11 interprets absence of the target article as "unable to cancel". It would improve the efficacy and reliability of *cancel*s to propagate them anyway, given that feed anomalies are widespread. There have been verified instances in which cancellations did not achieve anywhere near the propagation of the original article. In the interests of robustness, C News interprets absence of the target article as deferred cancellation rather than failure to cancel, and propagates the *cancel*.

cancel validation

RFC 1036 requires that a *cancel* message have a **Sender:** or **From:** header matching the message it is cancelling. It is not entirely clear from the text whether this restriction is supposed to be enforced at the originating site or at each receiving site, although the latter is implied.

More seriously, it is not clear what "matching" means in this context, considering that a substantial fraction of the information in such lines is typically in RFC 822 comments. There is an unfortunate tradition of news readers generating header comments in varying ways. There is also a lot of obsolete or misdesigned news software still operational, and some of it gratuitously alters the header comments (and sometimes even the non-comment parts of the headers!) in messages passing through. While theoretically these complications should affect the original and the cancellation identically, in practice this is not consistently so, and it is difficult to generate a cancellation that works dependably. This is not just speculation; there are verified cases of the originators of messages having considerable difficulty cancelling them when it was important to do so.

The value of RFC 1036 authentication is also somewhat questionable. It provides no useful security against malice, because news is so easy to forge. While there is some value in preventing accidents, there is room for doubt as to whether this is worth the interference with legitimate cancellations.

C News takes the position that the RFC 1036 approach to authentication is impossible to implement in a practical way, due to its vagueness and the prevalence of gratuitous and unpredictable header rewriting, and on balance the inability to cancel is worse than the largely-illusory security provided. C News therefore does not authenticate cancellations.

Doing something about the problem is difficult. Specification of a *precise* algorithm for header matching would help, but finding one that will disregard gratuitous header mangling is hard. A more appealing approach would be to authenticate cancellations by cryptographic means, but there are severe difficulties in key distribution on an unreliable non-real-time network like Usenet, and the cost of checking cryptographic credentials is disturbingly high. Ultimately, it may be necessary to abandon destructive control messages entirely, or reserve them for rare emergencies and route them through a trusted moderator for cryptographic authentication.

ihave/sendme not documented

The description of the ihave/sendme protocol is so vague as to be useless to an implementor. See the C news documentation for an adequate description of the protocol. The description in RFC 1036 also contains an error: *remotesys* is not optional; given that there may be multiple message-ids preceding it, there would be no way (other than ad-hocery) to tell if the final argument were a message-id or a *remotesys*.

Case-sensitivity in message-ids

RFC 1036 says nothing about whether message-ids are case-sensitive or not, thereby punting the issue to RFC 822. The RFC 822 rules are horrendously complex and no news system has ever implemented them correctly. (B 2.10 considers them fully case-sensitive, which is wrong. B 2.11 considers them fully case-insensitive, which is also wrong. C News gets the normal case right, but correct handling of certain obscure RFC 822 constructs would require a complex parsing algorithm; fortunately, the cases where this matters appear to be extremely rare.) Simplification appears necessary.

New headers

The B news **Supersedes:** header needs to be documented in the next revision of the RFC, as does the C news generalisation, **Also-Control:** (see *relaynews*(8)).

"Keywords"

Section 2 says that a header begins with a "keyword" as the header name. RFC 1036 never defines what a keyword is, and RFC 822 does not use the term. "Keyword" here must be considered an informal term with no precise meaning, imposing no additional restrictions on header syntax.

In particular, things like ">from: foo@bar", which causes B News to choke, appear to be legal RFC 822 (and hence 1036) headers. (Before quoting lexical rules, such as the requirement for balancing brackets, please note that the 822 lexical rules are context-sensitive.)

Theoretical legality notwithstanding, such bizarre header names are dubious and unwise practice. RFC 1036 probably should be tightened up to exclude them.

RFC 822 Comments

RFC 1036 section 2 implies, both in its general discussion and in its discussion of the "From:" header, that RFC 822 comments are not, in general, accepted in RFC 1036 article format. However, the point is not made loudly and explicitly, and some nit-pickers argue that RFC 1036 permits dubious practices like timezone name comments in "Date:" headers. This needs to be nailed down in black and white. C News takes a strict position on this in cases where it cares about the contents of headers.

Duplicate Headers

RFC 822 requires that at most one "Date:" header occur in a message, and likewise for "From:", although careful reading is needed to discover this. It permits more than one "Message-ID:" or "Subject:"

header, and is (of course) completely silent about ‘‘Newsgroups:’’ and ‘‘Path:’’. With the arguable exceptions of ‘‘From:’’ and ‘‘Subject:’’, duplicates of required headers are highly undesirable in news and cause difficulties for current implementations. RFC 1036 vaguely implies that the required headers are expected to be unique, but never says this. This needs to be made much more precise. C News takes a strict position and rejects articles with duplicate required headers.

Chapter 3.4:
Known Porting Problems With C News

Intro

C News in general is pretty portable. People have got it to run on a very wide range of systems with little trouble. Difficulties are usually problems in the system, not C News. Some of them, however, are widespread enough to be worth comment, for the guidance of people having problems. If you run into a novel problem, we are always interested in hearing about such things.

Unix Dependencies

The biggest portability glitch in C News is that it depends a lot on Unix utilities. The extensive use of complex shell files, *sed* and *awk* programs, and a wide range of lesser Unix utilities would make it quite difficult to move C News to a system that is seriously non-Unix-like. The actual C programs seldom depend on Unix in major ways. (An exception is the use of *read* system calls in *expire*, to avoid difficulties with stdio end-of-file behavior; we now know how to avoid this but haven't implemented the fixes yet.)

We know that *awk* and the colon (:) operator of *expr* are problem areas under Minix.

Shell Problems

C News seriously stress-tests shells. The current Minix shell is not robust enough in the face of complex inputs, botches some constructs entirely, and can run out of memory on the complex shell files. Any shell that is too old to implement comments begun by "#" is big trouble, since we use such comments everywhere.

Any system/shell combination that thinks that a shell script starting with "#! /bin/sh" should be run by the C Shell (because it starts with '#') is also big trouble: you will have to change that line to ": use /bin/sh" everywhere. We know that at least some releases of Xenix have this problem. It is not necessary that your kernel understand the "#!" feature—we believe that nothing in C News relies on it—but it is essential that it not cause invocation of the C Shell.

We know that some Hewlett-Packard Unixes have broken shells, probably the result of mistakes in HP's efforts to make the shell 8-bit-clean; the symptom is that something like:

```
x=y
if test " $x" != " y"
then
	echo oops
fi
```

prints "oops". This is, again, big trouble, because we do that a lot.

Many people using 3B1s, aka UNIX PCs, run the Korn shell as their */bin/sh*. Some other folks may do this too. Beware that *ksh* was not fully *sh*-compatible for a long time, with some subtle differences in the ill-documented behavior of backquotes and backslashes. Some of the C News shell scripts, notably *inews*, are known to hit these bugs. We are *told* that current *ksh*s have fixed them.

It is reliably reported that SunOS 4.0.*x* shells will dump core in some ill-defined circumstances, when the user environment (sum of all environment variables) is exactly the wrong size. Perhaps this has been fixed in 4.1.

It is reliably reported that the VAX 3.1 Ultrix shell is somewhat broken and gives various kinds of trouble. Switching to */bin/sh5* (note that this requires fixing the first line of a zillion shell files) is reported to banish the problems.

It is reliably reported that recent SunOS shells give the wrong exit status from the *wait* command: they give a 0 for a successful wait, rather than giving the exit status of the process waited for. This makes *inews −W* appear to always succeed even if backgrounded parts of it failed, which can be troublesome in NNTP environments where correct exit status is important.

Make Problems

There is a persistent problem on 3B2s with implementations of *make* that violate the SVID in a subtle way. They attempt to execute makefile commands directly, rather than via the shell, if the commands do not contain metacharacters. This means that if—as on many 3B2s—*test* is a shell builtin *and there is no /bin/test program*, the makefile line ‘‘test −s file’’ will cause *make* to complain about an unknown command. (The SVID says that makefile commands must be executed as if by the shell, and the shell will execute this line correctly.) We’ve added ‘;’ on the ends of such lines, which suffices to convince *make* to run a shell on the systems we’ve encountered, but AT&T is good at finding ways to break such workarounds. This problem is also known to occur in A/UX.

Another obscure problem, a bug in either *make* or the shell, appears in at least some releases of Ultrix: a construct like

ln ... || cp ...

in a shell file is seen as an error—and *make* aborts—when the *ln* fails, even though the *cp* would work.

Offsetof

ANSI C requires C compilers to supply a macro *offsetof*, which can be used to find the offset of a structure member within the structure. *Relaynews*’s header-parsing code uses it, defining it if the system has not supplied it. Unfortunately, it is really hard to write a portable version of this. The implementation we currently use is:

#define offsetof(type, mem) ((char *)&((type *)NULL)−>mem − (char *)NULL)

The table in *relay/hdrdefs.c* puts invocations of *offsetof* in initializers. This turns out to be a severe stress test for C compilers. A compilation error in *hdrdefs.c* is almost certain to be problems with this macro. Some compilers, notably the one in Microport System V for the 286, reject it. We have heard a report that System V Release 2 on the VAX silently miscompiles it! If you have trouble with *offsetof*, you might try this version instead:

#define offsetof(type, mem) ((int)&((type *)NULL)−>mem)

Fast Stdio Routines

We supply a set of fast standard-I/O routines that are compatible with most AT&T-derived implementations of *stdio*. They speed up C News quite a bit. However, they don’t work on all Unixes. The tester program we supply, which the library-build procedure runs, is thought to diagnose such problems 100% of the time. It has been reported in the past that A/UX and Microport 386 stdios flunk the test. SunOS 4.0 used to pass the test falsely, but improvements in both the test and the routines seem to have cured the problems: 4.0.3 passes the test and as far as we can tell, the routines run correctly under it.

In any case, if you are feeling nervous or are having mysterious problems, telling *build* that you don’t want to use the fast-stdio stuff is always safe. This is also the best response if you have trouble compiling those routines.

void

Old compilers that don’t understand the *void* type will choke on much of our code. There is a commented-out ‘‘#define void int’’ in *news.h* that cures most cases of this if you uncomment it. (We have a report that you might need to add ‘‘−Dvoid=int’’ to the Makefile in *libv7* if you’re using that library.) C News does not rely on the ANSI C ‘‘void *’’ type as far as we know.

Modes in fopen

Unix V7 documented only ‘‘r’’, ‘‘w’’, and ‘‘a’’ as suitable mode arguments to *fopen*. It actually implemented the read/write modes, ‘‘r+’’, ‘‘w+’’, and ‘‘a+’’, as well, and C News relies on them. Unix reimplementations based on old documentation may have trouble here; we know that at least the older versions of Minix really don’t implement these modes.

A related complication in Minix is that *ftell* reportedly doesn't give the right answer in ''a'' mode. This makes *dbz* flunk its regression test.

MAXLONG

The *relay/cpu.h* file formerly defined a constant *MAXLONG* which is the biggest positive value of a *long*. The definition was clever but failed on some odd systems (Unisys?). Current versions of C News generate the value dynamically in a less fallible way, and check the value for plausibility. (This is encountered when *relaynews* is asked to process a single article, not a batch. This happens primarily when an article is posted locally, with *inews*.) It is still barely possible that the plausibility check will fail on some bizarre system.

df Output Format

The *spacefor* utility needs to understand the output format of *df*, unless you're lucky enough to have a system that has one of the semi-standard system calls to report disk space. There are numerous variations on *df*. *Build* and the relevant makefiles know about the more common ones, but customization may be necessary. *Spacefor* is commented well enough that it should be possible to figure out the necessary changes; usually the initializations of *nr* and *nf* are all that need changing. If there are colons (:) in your *df*'s output format, you should probably start from the ''sysv'' *spacefor*, which attempts to preprocess the output to get rid of System V garbage; otherwise the ''bsd'' one is a reasonable starting point.

One constant nuisance is *df*s that are too stupid to take a directory name as an argument. The long-term solution to this is to edit a suitable variant of *spacefor* to know about the proper arguments. A short-term solution is to use the ''null'' variant, sacrificing space checking for the sake of getting something working.

We're told that HP-UX 7.0 users are best advised to choose the ''bsd'' variant of spacefor, and edit it to call *bdf* instead of *df*. Similar approaches may be useful on other hybrid SysV/BSD systems.

Floating Point

The only places in our code where floating point is used, as far as we know, are in the calculation of expiry dates in *expire* and the calculation of space in some of the variants of *spacefor*. These are not performance bottlenecks, so slow floating-point arithmetic is not a problem. Complete absence of floating point would require only minor modifications. Note, however, that we use *awk* a lot, and the typical *awk* implementation uses floating point extensively.

386 Optimizer vs. dbz

We have a reliable report that the System V 386 optimizer (invoked when *cc* is given the **−O** option) miscompiles the *dbz* package badly enough to cripple it, without producing any error messages. The only fix is to compile *dbz* without **−O**.

SCO Xenix/386 2.3 has the same problem.

nnafree and nnfree

We have a reliable report that the HP Spectrum C compiler has an optimiser bug that makes it throw up (with a ''cc: Internal error 3279: Please contact your local HP representative'' message) on the *nnafree* macro (and *nnfree*, a historical synonym) in *h/news.h*. The following revised version of the macro reportedly avoids the problem.

```
#define nnafree(mempp) do { if (*(mempp) != 0) { free((char *)*(mempp)); \
  *(mempp) = 0; }} while (0)
```

It is also reliably reported that the Microport compiler objects to these macros in large model. Whether the above fix would suffice is not known. Manual expansion [barf!] is known to work, although it would be less painful to define a function containing the right code and change the macro to call the function. Code for a suitable function can, in fact, be found in *h/news.h*, inside '#ifdef lint'.

ANSI C

Although we made an effort to be ANSI-C compatible, lack of access to a real ANSI C compiler limits our ability to do this. A secondary problem is that it's really very difficult to get code that is 100% acceptable to both ANSI C compilers and older compilers. Some issues inevitably got resolved in favor of current compilers, and may cause complaints from ANSI C compilers.

Work is in progress on moving us closer to ANSI compatibility. Beware that if **__STDC__** is defined by your compiler but it is *not* ANSI compatible, you are on your own as far as we're concerned, even if the value is specified as 0. (We can't just use "#if __STDC__" because some preprocessors choke on the appearance of an unknown identifier in #if.)

GNU C

If compiling with the GNU compiler, you may need the **−traditional** flag. Beware, also, that if you are using your system's *dbm* library, it contains functions that return structure values, and the GNU conventions for handling such values are incompatible with the ones in many AT&T-derived compilers. The **−fpcc-struct-return** option cures this.

Awk Problems

A number of problems can arise if your *awk* has bugs, since the shell files rely on *awk* fairly extensively. For example, *awk* is a prime suspect if *spacefor* doesn't work. We've fixed the worst trouble spots, but would appreciate detailed information on any more.

One known problem that is hard to avoid is line-length limits in *awk*. In particular, for several purposes in control-message handling C News wants to build a "canonical" representation of the *sys* file, with backslashed newlines removed. This is done by NEWSBIN*/relay/canonsys.awk*. Most *awk*s have limits on line length, and some of the limits are too low to cope with long multiply-continued *sys* lines. 512-byte limits, found in a number of old *awk*s, are particularly troublesome. A quick look indicates that this will interfere, to some uncertain extent, with *checkgroups* and *sendsys*. Big deal. :-) The complaint may also appear from *newgroup*, but there it should be harmless.

Bart Muyzer and Martijn Roos Lindgreen report that HP-UX 8.0 *awk* is badly broken, such that (for example) the regular expression "/[\t]/" will match backslashes and t's as well as tabs and spaces. Installing the HP-UX 7.0 *nawk* as *awk* is reportedly a workable fix.

Systems Without Hard Links

Some vaguely Unixoid systems have trouble implementing real ("hard") links. Examples are VMS in general and Eunice in particular. There are some hooks for dealing with this, but it's not trivial.

Relaynews will try to make symbolic links if real ones fail. There is one exception: if *relaynews* cannot buffer up enough of the article in memory to find the 'Newsgroups:' line, it will drop the article into a temporary file and will rely on being able to move that to the first of the 'real' locations by manipulating links. This should essentially never happen except on 16-bit machines, and should be rare even on them.

Expire has a **−l** option which tells it to consider the first filename of an article its 'leader', not expiring the article under that name until it has expired under all others. This has not been tested much recently.

The locking system (both C routines and the *newslock* program) will need revision in some system-dependent way to avoid use of links.

There is one minor use of links in installation (*inews* is found in two places, and the Makefile attempts a link before doing a copy), and substantially more in the regression tests.

16-bit Machines

C News has been tested on 16-bit machines—indeed, a good bit of the early development work was done on one—and does run on them. Nothing relies on ints being 32 bits. Nothing relies on pointers and ints being the same size, as far as we know. Nothing relies on large address spaces, although one or two modules come in separate small-space and large-space versions, and the small-space versions are slower.

However, there are some fundamental limits to consider. Both *relaynews* and *expire*—the usual trouble spots for space shortages—want to keep lots of stuff in core. There isn't any easy way around this one.

Number of File Descriptors

There is a constant, NOPENBFS, in *relay/trbatch.c*, that defines how many batch files are kept open simultaneously. If you are feeding much news to more systems than this, *relaynews* performance will suffer.

The major limit on NOPENBFS is available file descriptors (although on a 16-bit machine there might also be a shortage of memory for *stdio* buffers). Other parts of *relaynews* want perhaps 10 file descriptors for other purposes, so with the usual total supply of 20, a NOPENBFS value of 10 is the right default. If you feed many people, and your kernel provides a process with more than 20 file descriptors, you probably want to boost NOPENBFS (this can be done with -DNOPENBFS=xxx in the makefile). Remember to leave about 10 descriptors worth of headroom.

Shell Processing Order

Normally, shell variable expansion should take place before scanning for syntax elements such as "0<&1". At least one reimplementation of the shell (specifically, Bash 1.04) does things in the wrong order. This is known to affect, at least, *relay/sh/anne.jones*, which can be fixed by changing (circa line 44)

"")   USER="‘who am i <&$fd |

to

"")   USER="‘eval \"who am i <&$fd\" |

or so we are told.

Binary-Mode Fopen

In several places, the new *dbz* uses ANSI C "binary mode" fopen codes, e.g. ‘fopen(..., "r+b")’. The text/binary distinction involved is meaningless under Unix, and most Unix implementations just ignore trailing nonsense in the second argument of *fopen*, so it all works out nicely.

Unfortunately... at least one version of DEC's Ultrix objects to the ‘b’s, we are told. Sigh. DEC will have to fix this to make their systems ANSI compatible, but heaven only knows how long that will take.

Meanwhile, the fix is to delete the ‘b’s in the second arguments of the *fopen*s in three places in *dbminit()* in *dbz/dbz.c*, if your system has this particular bit of brain damage.

size_t

Some systems, notably from Microport, do not define the *size_t* type in the *<sys/types.h>* header. Worse, the *size_t* in that header doesn't necessarily bear any relationship to the ANSI C *size_t*. This causes trouble in the *dbz* library in particular. The proper type for *size_t* is whatever the C *sizeof* operator returns, nominally an unsigned type which is large enough to contain the size of any memory object. We think nothing relies too heavily on it being unsigned. Note that *size_t* must also be suitable for use in the two middle arguments of *fread* and *fwrite*, the last argument of *memcpy*, *memchr*, and *memcmp*, and the argument of *malloc*.

Background Processes vs. csh

Inews runs much of its processing in the background. We are told that this can hit problems, in some circumstances, with *csh*'s manipulations of signals, terminal modes, etc etc. We prefer a standard shell, and have made no attempt to understand the C shell's weirdnesses. We're aware that well-written programs can fail under the C shell due to bizarre problems with weird signals, etc., but we class this as the fault of the C shell and its co-conspirators and decline to contort our programs to compensate for its failings. We do sympathize with people victimized by it, but can be of no practical help.

Compress Behavior

Extremely old (pre-1985) versions of *compress* run off at the mouth with a status message on *stderr* even when nothing goes wrong in the compression. This upsets the batcher, which thinks any *stderr* output means trouble.

ulimit

Most versions of System V have the concept of *ulimit*, a per-process limit on how big an individual file can be. This limit can be lowered by anyone but raised only by the super-user; normally *init* or *login* initializes it to some suitable value. Unfortunately, many System Vs set it far too low, at 1 megabyte. This causes trouble with many things, but in particular, *relaynews*, *expire*, etc. need to be able to work with the *history* file, which can easily be several megabytes. It's necessary to deal with this on all paths by which any of these programs might be invoked: from *uucp* or other transport software bringing in news, from *cron*, and by users via *inews* for local postings. It is difficult to do this in a portable way when super-user privileges are needed.

Restricted Shells

There is an unfortunate interaction between the '#!' feature in shell files and the ''restricted shell'' feature found in some Unixes (notably System V): the restricted shell typically is just a different way of invoking */bin/sh*, and in some versions, careless code just checks the first letter of the name the shell was invoked under to see if it was 'r'. Unfortunately, if the system has the '#!' feature and there is a shell file named, say, *rnews* whose first line is '#! /bin/sh', this shell file will end up invoking the restricted shell!

Simply deleting the '#!' line might fix this; on systems which have the Korn shell, changing '#! /bin/sh' to '#! /bin/ksh' might work. Otherwise you will have to arrange to have your neighbors invoke *cunbatch* instead of *rnews*, or else write a simple *rnews* that invokes the real one under another name. It would be wise to check for other shell files whose names begin with 'r', also, as *rnews* definitely isn't the only one.

Remote Invocation vs. Nonstandard Shells

When *newsrun* is invoked on a host that is not the news server, it uses *rsh* to propagate news batches to the server. It runs */bin/sh* explicitly to avoid major difficulties with non-standard shells, but it has to rely on the invoker's login shell to run that one command. This means *newsrun* will emit spurious output if its invoker's login shell is the C shell and its invoker's *.cshrc* contains commands that generate output. A similar problem occurs with *bash* and *.bashrc*.

The simplest solution is to use */bin/sh* as the login shell for *newsrun*'s invoker. Otherwise, if you have a networked news server, check that the login shell is standard enough to invoke */bin/sh* by executing the following command as *newsrun*'s invoker.

rsh *newsserver* **exec /bin/sh –c true**

This command should output nothing.

A slightly related problem is that not everyone calls the run-remote-shell command *rsh*; on System V in particular, *rsh* means something different. For the moment we have opted to ignore this issue, as the possibilities for gratuitous differences boggle the mind. People facing this problem may wish to place an *rsh* shell file in the search path to invoke the right command in the right way, whatever that is.

Values of Logical Operators

There seem to be compilers, e.g. the Ultrix one on DEC's new RISC workstations, that go into convulsions when they see something like

***p++ = isascii(c) && isalnum(c);**

because they don't know how to generate a numeric value for '&&', or because they don't know how to turn that value into a 'char'. One or two places in C News use constructs like this. If you run into this, you might want to try replacing the right-hand side with something like ''(...) ? 1 : 0'' to get the troublesome operator back into a conditional context.

Empty Lines

Some backward operating systems (through which your C News distribution may have passed on its way to you), and perhaps some stupid text-handling software even on sane operating systems, do not recognize the notion of an empty line. They think all lines must have at least one character in them; the closest they can come to an empty line is a line consisting of a single blank. This matters because *relaynews* will tolerate white space only in certain places in the *sys* file, and in particular, it tolerates empty lines but not lines consisting solely of white space. The result will be complaints (in *errlog*) about white space in the *sys* line for a system named " ".

active-File Date

On some Bull systems, at least ones running DPX/2 B.O.S. 1.0, apparently *relaynews* updates the contents of the *active* file correctly, but the file's date remains unchanged! This appears to be a kernel bug. It reportedly upsets some news readers. A workaround, said to be effective, is to add the line

```
utime(ctlfile(actrelnm), (time_t *)NULL);
```

after the call to *nnafree* in *actfsync* in *libbig/active.big.c*.

enum Operators

Some compilers have difficulty compiling the *readnews* we supply, because they object to applying the '!' operator to an *enum* type. Changing the definition of *booltype* in *rna/defs.h* to

```
typedef int bool;
#define	false	0
#define	true	1
```

is reported to solve this.

Amiga Library Ordering

It is reliably reported that when compiling some of the programs under SVR4 on the Amiga, it is necessary to give ''−lc −lucb'' as library options—linking of the C library *must* precede linking of the Berkeley-emulation library, or the code links but will not run.

AIX and Mach vs. fsync()

The *relaynews* regression test fails under some (all?) AIXes, because the system refuses to do an *fsync* on a file descriptor open to */dev/null*. It is possible that this does not affect production use, however. Mach (at least on the NeXT) is reported to have similar problems.

AIX/370 vs. Shell Files

AIX/370 has added at least one keyword (''on'') to the shell, and this is known to cause syntax complaints in at least one shell script (*newsrunning*). Unless this is also a keyword in the final version of POSIX 1003.2, we don't plan to fix this.

Struct Conditional Expressions

Some (all?) SCO Xenix compilers take offense to expressions like

```
value = (dbzint) ? dbzfetch(key) : fetch(key);
```

where the functions return *struct* values. This occurs in three places in *dbz/dbzmain.c*, and the workaround is to expand those conditionals to statements like:

```
if (dbzint)
        value = dbzfetch(key);
else
        value = fetch(key);
```

staleness vs. Ultrix

Several Ultrix users have reported a problem with the ‘‘staleness’’ command. It seems Ultrix’s *sed* is an antique and blows up on the complex regular expression in *staleness*. A fix, at some small cost in performance, is to change the last four lines of *staleness* to something like

```
exec awk '$1 == "/expired/" { print "-o", $3 }' $NEWSCTL/explist
```

SCO Xenix string functions

Under SCO Xenix 2.3, and perhaps other recently-released SCO systems, the string functions like *strchr* exist but can be inordinately slow when dealing with long strings. This is not an academic issue: one symptom is that *relaynews* takes a long time to start up, eating 10-15 seconds of CPU time before it starts processing articles. This apparently is a combination of badly-written code and strange internationalization support. Just using our string functions, by telling *build* that your system does not have them, works much better. Telling the compiler **−nointl** may be helpful if you don’t want to go that far.

Old SCO Xenix vs. ***setvbuf***

The *dbz* package makes some use of the *setvbuf* routine. Incredible though it sounds, old versions of SCO Xenix reportedly had the order of *setvbuf*’s arguments wrong! If you have SCO Xenix version 2.2 or earlier, check the arguments to *setvbuf*: if the second and third are a type and a buffer pointer respectively (they are supposed to be a buffer pointer and a type), you’re in trouble and will have to tinker with the *dbz* sources.

SunOS 4.1.1 vs. write()

In some circumstances, a SunOS 4.1.1 *write* system call to a disk file can be interrupted by a signal. No other Unix does this, and routines like *fwrite* are not prepared to cope with it. This can result in gratuitous failures of *dbz* in particular.

It is thought not to be a problem in C News, but some other packages using *dbz* suffer, and we mention it just in case. Sun acknowledges it as a bug. The bug-id is 1052649. It is fixed in patch 100293-01.

uucp Variations

There are innumerable variations on the details of *uucp* that may require appropriate modifications to *queuelen*. For example, some versions of Honey DanBer (aka BNU) *uucp* cut all system names down to seven characters, and *queuelen* will have to be altered to do likewise.

malloc Variations

On some systems, performance is noticeably better if the **−lmalloc** library is used, rather than relying on the *malloc* in the standard C library. A/UX is reportedly an example.

Slow egrep

At least some System V suppliers (including, reportedly, Apple in some [now obsolete?] A/UX versions) have broken *egrep* in such a way that it is inordinately slow. It may be worth substituting *grep* for *egrep* in some of the shell scripts, with appropriate caution since they do not accept quite the same pattern syntax.

Chapter 3.5:
C News vs. VMS

To run C News at all, you need a fairly good emulation of Unix. There are several such for VMS. They have various minor imperfections. The only one we specifically *know* of that is a real problem for C News is the inability to make real links for cross-postings. There is some half-hearted code in various places that tries to deal with this situation. It has *not* been tested too thoroughly.

Relaynews normally files an article under its first group and then makes hard links into further groups. If *relaynews* finds itself unable to make a hard link, it will try making a symbolic link instead. The one situation where this will fall down is if a news article's header is enormous, too big to fit in core. In this case, *relaynews* stores the article in a temporary file, makes links to it under all the appropriate names, and then unlinks the temporary name. This obviously isn't going to work if the appropriate-name links are symbolic. We believe this case essentially never happens on large-address-space machines, and is quite rare even for small address spaces.

Expire has a **–l** option that tells it to consider the first name of an article as the "leader", not to be deleted until all others have been deleted.

The one place where extra work would be necessary would be *mkhistory*, which has no notion that some links are different from others.

So far as we know, we don't get into any of the other trouble areas of Unix emulation on VMS, at least with the Eunice emulator. We don't have a VMS handy for testing, so we make no guarantees.

Section 4: **Implementation**

Chapter 4.1: **Control Message Implementation in C News**

Introduction

Netnews *control messages* are ordinary-looking netnews articles which contain the special header **Control:**. Such articles are filed in the pseudo-newsgroup *control* and cause related actions by the local news system, such as mailing a file to the poster of the control message.

Built-ins

ihave, **sendme**, and **cancel** are handled internally by *relaynews*, because processes cannot share open *dbm*(3) databases, there is no standard way to close them, and these control messages read or write the *history* files, including the *dbm* files. *Ihave* and *sendme* are also permitted to have message-id arguments containing **<** and **>**, both of which are rejected by *relaynews* in arguments to externally-implemented control messages, since they are shell metacharacters (**/** and **..** are also bounced) and could be indicative of an attempt to do something nasty.

Normal Control Messages

Most control messages are implemented by *relaynews* by executing the command line following **Control:** with a search path of *$NEWSCTL/bin:$NEWSBIN/ctl* and with standard input set to the control message article. The command inherits the standard news search path and *relaynews*'s user and group ids, typically *news*; this can be important to gain access rights to control files. The news system will be locked (by *$NEWSCTL*/lock) while the command runs, because this is often important for manipulating control files from the command, and because the news system is locked while *relaynews* runs. If that command returns non-zero exit status, mail is sent to *$NEWSMASTER* (usually **usenet**). Standard output and standard error often are redirected to *$NEWSCTL*/log and *$NEWSCTL*/errlog respectively; invocations of *relaynews* by *inews* are exceptions.

Chapter 4.2:
A Tour Through the C News Libraries and Include Files

libc and friends

libc contains routines that are sufficiently useful and general that they could profitably be added to one's C library. Indeed, on some systems they are in the C library. Notable inventions include *fgetmfs* which safely reads arbitrarily-long input lines; it has an **fgetmfs.h** over in the header directories. *ldiv* is the ANSI one, if you need it. *memlist* is a package to ease keeping track of a lot of allocated memory. *stdfdopen* is invoked by setuid programs to ensure that the standard file descriptors are indeed open, opening **/dev/null** on each closed standard descriptor.

libstdio contains new (faster) guts for the original UNIX *stdio* library; if they compile on your V7, 4BSD or System III system, you may want to use them instead of the default versions in your C library. On System V, these routines are only slightly faster than the versions in the C library.

libfake contains things that probably should be in your C library, but might not be, and a couple of fake routines for system calls you might not have.

libcnews

libcnews contains functions of general use to news software, but not the world at large. *complain* is like *warning* , but never prints the symbolic value of *errno* .

There is a locking package, containing *lockdebug* , *newslock* , *newsunlock* , *errunlock* , and *nemalloc* . *lockdebug* enables or disables lock debugging. *newslock* attempts to lock the news transport layer against read-then-write access to the **active** file, writing to the **history***, **log**, **errlog**, and batch files. It returns only when it has the lock; in the meantime it sleeps and retries until it gets the lock. There is no timeout; this is a feature. *newsunlock* removes the above-mentioned lock if this process locked the news system. *errunlock* is like *error* except that it unlocks the news system (via *newsunlock)* before exiting; it is should always be called instead of *exit*(3) or *exit*(2) if there is any chance that this process has locked the news system. *nemalloc* is like *emalloc* but it calls *errunlock* if it can't allocate memory.

ltoza converts a **long int** to a string of a given width, containing the decimal representation, zero-padding as needed on the left. *ltozan* is like *ltoza* but omits the terminating NUL, so it can be used to over-write a string without truncating it. *ngmatch* returns a truth-value resulting from comparing a list of news-group patterns and a list of newsgroups. One may substitute ‘‘distribution’’ for ‘‘newsgroup’’.

There is a package of pathname manipulators. *artfile* returns a name for its filename argument, assumed to be relative to *$NEWSARTS* ; *fullartfile* promises to return a fully-qualified path name. *ctlfile* returns a name for its filename argument, assumed to be relative to *$NEWSCTL* . *binfile* returns a name for its filename argument, assumed to be relative to *$NEWSBIN* . *cd* changes to its argument directory, check for errors, and notes the directory name, by making a private copy, for later optimisations. *newsumask* returns the value of *$NEWSUMASK* . *newspath* returns the value of *$NEWSPATH* . *newsmaster* returns the value of *$NEWSMASTER* . All these functions supply default values for the *NEWS** variables if none are found in the environment. If values are found in the environment, they are used, and a function named *unprivileged* is called.

readline is like *fgets* , but executes *newslock* upon encountering EOF and retries the read. This is used when reading growing files such as **history** or batch files. *strlower* down-cases an entire string, in place. *strsave* is like *strdup* but it calls *nemalloc* rather than *emalloc* . *str3save* takes three strings, allocates space for their concatenation via *nemalloc* , including terminating NUL, and concatenates them onto it. A *NULL* argument will be replaced by an empty string. *timestamp* writes a timestamp on a given stream, and returns the current time via an argument for later use.

Unix-variant-specific libraries

There are several libraries that provide functions for talking to specific Unix variants. These are basically functions that change from one variant to another. **libfake** (see above) contains things which simply might not be there in a particular system.

These libraries all provide the same virtual interface to operating-system-dependent services: *fclsexec*, *fopenexcl*, *getcwd*(3), and *gethostname*(3). Implementations for vanilla implementations of these UNIX variants are provided: Seventh Edition, including 4.1BSD (**libv7**); Eighth and Ninth Editions (**libv8**); 4.2BSD and later (**libbsd42**); System III and System V (**libusg**). *fclsexec* sets the close-on-exec flag for a given *stdio* stream. *fopenexcl* performs an "exclusive create" open: the open fails if the file already exists.

Address-space-size-specific libraries

These libraries provide alternate versions of the **active** and **sys** file code: **libsmall** should work on any machine, but is suboptimally fast; **libbig** has worked even on PDP-11s, and is quite fast, but consumes memory without apology.

Include files

libh contains include files unique to C news. **news.h** defines a few limits, some file names, some types (**boolean** and **statust**), some characters, some status bits, some macros for speed, and declares functions in **libcnews** or miscellaneous functions in *relaynews* (oops!). **config.h** declares the pathname functions in **libcnews**. **fgetmfs.h** declares symbolic values and macros for using **fgetmfs**. **libc.h** is a start at a V9-like declaration of all of the C library. **memlist.h** defines the interface to **memlist**.

hfake contains a few include files that your system ought to have but might not. **stdlib.h** is a degenerate ANSI **stdlib.h**. **string.h** declares the string functions. **timeb.h** declares the structure used by *ftime*.

Chapter 4.3:
Locking in C News

Several parts of C News need some way of locking parts of the news subsystem against concurrent execution. Various system-specific locking system calls exist, but none of them is truly portable, and most of them provide far more functionality than we need.

C News locking uses the *link*(2) system call and pre-agreed names. *Link* has the necessary characteristic for safe locking: it is an atomic test-and-set operation. Furthermore, it exists in all Unixes.

All locks are created in the NEWSCTL directory (see *Configuration Mechanisms in C News* for where this directory is to be found and how programs can determine this) and have names starting with 'LOCK'. To acquire a lock, first create a temporary file in NEWSCTL with a name of the form 'L.*n*', where *n* is your process id. You are urged to also write your process id, in decimal ASCII, into this file. Then attempt to link the temporary file to 'LOCK*x*', where *x* is chosen based on what sort of locking you wish to do. Existing lock names are:

LOCK	relaynews, modifications to control files
LOCKinput	input subsystem processing spooled input
LOCKbatch	batcher preparing batches
LOCKexpire	expire expiring articles

If the link fails, sleep and try again. If it succeeds, proceed. The temporary file may be removed then or at the same time as the lock is removed. Programs are expected to make a determined effort to remove lock files when they terminate, normally or as a result of signals.

Shell programs have an additional problem in that System V has broken *ln*(1) so that it removes a pre-existing destination file. C News therefore provides a pure, simple locking program under the name NEWSBIN/newslock (if the recommendations in *Directory Layout and PATH in C News* are followed, this will automatically be in the search path of shell programs). Usage is 'newslock tempfile lockfile'; exit status is 0 for success, 1 for failure, 2 for wrong number of arguments. No messages are printed for normal failure, so no redirection of output is needed.

A suitable locking procedure for a shell file using the standard configuration facilities is:

```
lock="$NEWSCTL/LOCKxxx"          # modify name as appropriate
ltemp="$NEWSCTL/L.$$"
echo $$ >$ltemp
trap "rm −f $ltemp ; exit 0" 0 1 2 15
while true
do
	if newslock $ltemp $lock
	then
		trap "rm −f $ltemp $lock ; exit 0" 0 1 2 15
		break
	fi
	sleep 30
done
```

A template of this form can be found in the file *newslock.sh*.

Although there are various thorny questions associated with breaking locks by dead programs, reboot is a time when surviving locks are definitely invalid. (Although there are problems even here if a networked group of systems are not rebooted as a unit.) For this and other reasons, a system running C News should execute NEWSCTL/bin/newsboot at reboot time (e.g. from */etc/rc*).