1. News installation

The *getdate(3)* routine must be installed prior to installation of *news.* Consult the getdate subdirectory before proceeding.

Edit /usr/include/whoami.h and change the #define for sysname to the name chosen for your system. If you will be receiving news from other systems, add an entry for "rnews" to the Cmds array of uuxqt.c, then remake uux. (Other useful suggestions may be found in the "usenet/setup.n" document.)

News runs setuid to keep its own files secure. You must decide on the uid and gid to be used by *news.* In the supplied makefile, these are both "daemon". You should edit /etc/passwd and /etc/group to include these two groups, or you can change the news uid and/or gid by changing the appropriate make variable (in makefile).

News may now be installed:

```
su root
make install
make clean
```

News starts up much faster if it is made "sticky" (as is the case with any large shared-text program). The news.1 manual page may be put in /usr/man/man1/news.1. Also, you may want to edit the ".ngfile" and ".sys" files that are described below.

2. News Files

2.1. /usr/bin/news

This is the executable news program. The program "rnews" is a link to *news* which, when invoked, reads a network-formatted article from its standard input.

2.2. /usr/spool/news

This directory contains the files used by the news program.

2.3. /usr/spool/news/.bitfile

This file is a bit map maintained for the benefit of *login(1).* The following routine can be embedded in login to check if a user has news:

```
#define BITFILE "/usr/spool/news/.bitfile"
/* return 1 if user has news, else 0. */
newscheck(uid)
register int uid;
{
        register int fd;
        static char c;
        fd = open(BITFILE, 0);
        lseek(fd, (long)(uid>>3), 0);
        read(fd, &c, 1);
        close(fd);
        return((c >> (uid&07)) & 01);
}
```

The bit map is recreated when an article is inserted or cancelled, and it is updated when each user reads his new news. Each user's bit is one if he has news, and zero if not.

The bit map may also be interrogated using the supplied *nchk* program.

2.4. /usr/spool/news/.ngfile

This file lists all of the accepted news groups on the system. Each newsgroup to which an article is submitted or to which a user subscribes must appear in .ngfile. Newsgroups are listed in the .ngfile one per line. The local administration may modify this file as desired. A typical .ngfile for system "xyz" would be:

```
general
sys
dept
NET.general
test
to_duke
```

The "general" newsgroup should be used for most articles. The "sys" newsgroup might be for news of interest to systems programmers, while the "dept" newsgroup would be for news of departmental interest only (e.g. parties). Articles submitted to "NET.general" are put on the net (see below). The newsgroup "test" may be used for local tests of news. The newsgroup "to_duke" sends news only to Duke and may be used for network news tests.

2.5. /usr/spool/news/.sys

This file lists the newsgroups which the local system and connecting remote systems receive from the net. The file for system xyz might be

```
xyz:NET.ALL,to_xyz
duke:NET.ALL,to_duke
```

Each entry consists of the system name, a colon, and a list of comma-separated newsgroups. (The line may also contain two additional fields which are described below under "Transmission of network news.") The systems xyz and duke subscribe to all "NET" articles, xyz alone subscribes to "to_xyz" (a test newsgroup), and duke alone subscribes to "to_duke" (another test newsgroup). The local administration may edit the .sys file as desired. For example, most of the systems in the Research Triangle, NC, subscribe to the sub-network newsgroup "triangle".

2.6. /usr/spool/news/sys.nnn

Files with names of this form (e.g. "duke.122") are news articles. According to the protocol, every network guarantees to give each article it generates a unique file name. This is done by appending a unique integer to the local system name.

2.7. /usr/spool/news/.uindex

This file contains one line per news user. Each line has three colon separated fields. The first field is the user id. The second field is the time (in seconds since 1970) when the user last read news. The third field is the subscription list, which consists of newsgroups separated by commas. A missing third field is taken to be ":general".

2.8. /usr/spool/news/.nindex

This file contains one line per news article. The three colon separated fields on each line are for the file name, the date of submission (locally), and the list of comma-separated newsgroups to which the article belongs. As with .uindex, a missing third field is taken to be ":general".

2.9. /usr/spool/news/.seq

This file contains the sequence number given to the last article generated locally.

2.10. /usr/spool/news/.history

In order to prevent redundant traffic and spurious resubmission of news, the file name of every news item ever received by this system is kept in .history.

3. Network news

3.1. News article transfer format

The first character of the transferred file identifies its format and will be used to simplify inevitable changes in article formats. Currently, this character is ’A’. The rest of the first line is a unique network-wide name, which also identifies the originating node. The article name is used to prevent the unlimited duplication of news articles that might otherwise occur. The second line is a comma-separated list of newsgroups to which the article belongs. Newsgroup names may not contain colons. The third line identifies the contributor of the article. It is a system-pathname sequence suitable for mailing a reply via *mail(1).* The fourth line is the contribution date in *ctime(3)* format. The fifth line is the article title. The remaining lines are the text of the article. The article is ending by ’.’ alone on a line, or end of file.

Figure 1 - news transmission format.

```
Aduke.405            . Format id (’A’) followed by
                       the article name.
test                 . newsgroups list.
duke!swd             . path name of author.
Fri May 16 10:29:40 1980. date of original submission.
test                 . article title.
testing one two three      . text of article.
```

3.2. Transmission of network news.

When a news item is entered into the local news system, the .sys file is scanned. For each system in the file whose name does not appear in the author-pathname line of the item, and whose subscription list matches the item’s newsgroup list, *news* attempts to run "rnews" on the remote system with the article as standard input. If the last (fourth) field in the ".sys" file is empty, then *news* uses uux to execute rnews on the remote system. Otherwise, the fourth field in the file is a command line which which is invoked with the article on standard input. This transmission program (command line) assumes responsibility for passing the article unchanged to "rnews" on the remote system. Frequently, the program invoked will be a network mail program. Setting up *news* to use *mail(1)* for transmission of articles is described in detail in the "Examples" section below.

3.3. Reception of network news.

The rnews program reads an article from standard input and processes it for inclusion in the local news. *Rnews* throws away all articles mentioned in the history, and removes all newsgroups from the newsgroup line to which the local system does not subscribe (as specified in .sys). Articles that the local system does not subscribe to at all are thrown away. As *rnews* copies an article into /usr/spool/news, the local system name is prepended to the system-pathname line. After the article has been copied the article is retransmitted, as modified by the reception process, to all systems which subscribe to the article. Note that if system xyz does not subscribe to NET.test (for example), it will not receive or retransmit articles submitted only to NET.test, and will remove NET.test from the newsgroup line of any article passing through xyz.

4. Examples and little documented features.

4.1. Catching rnews diagnostics.

Some of the diagnostics from *rnews* may be interesting. You can arrange to have these diagnostics mailed to you by "mv"ing the rnews link to rnewsx and making rnews a shell script: "rnewsx 2>&1 | mail root". Be sure to make the rnews script executable. (The news makefile does not know about rnewsx, so "make install" will undo this change.)

4.2. Users who should not get news.

News assumes that all users should subscribe to "general" so that they will receive administrative notices of system downtime &etc. If an article is submitted to general, then users will be greeted at login time with "You have news." Some users (e.g. uucp) should not be bothered with such messages. Fortunately it is possible, though inconvenient, for a user to subscribe to nothing. Suppose uid 253 should never receive news. Manually change the line in .uindex that begins "253:" to contain only a subscription to "None"

```
253:328208485:None
```

Since "None" is a non-existent newsgroup, uid 253 will never "have news."

Such users will still receive the message of the day and "You have mail" notices, so eliminating the news notice may be irrelevant. Duke's login suppresses all messages when the user has a non-standard shell (e.g. uucico).

4.3. Adding a new system to the news network

To tell news about a new system with which it is to exchange news, first add a line to the .sys file:

```
newsys:NET.ALL,to_newsys
```

If you cannot obtain *uux(1)* permission to run *rnews* on newsys, then you will have use some other means of transmitting articles, and modify this .sys entry accordingly. Use of *mail(1)* to transmit articles is explained below. Now add the test newsgroup to your .ngfile:

```
echo "to_newsys" >>.ngfile
```

and welcome the new system:

```
   echo "welcome to USENET" | news -i hello -n to_newsys
```

4.4. Subnetworks

The newsgroup matching scheme, together with the system subscription lists (the .sys file), are designed to allow for the easy creation of subnetworks. To create the research triangle subnetwork which consists of only the newsgroup "triangle," each of the triangle area systems added "triangle" to the list of newsgroups received, as specified by their local entry in the .sys file, and added "triangle" to the list of newsgroups sent to participating systems.

For example, the .sys file at U.N.C. might look like

```
unc:NET.ALL,triangle,to_unc
duke:NET.ALL,triangle,to_duke
```

If the triangle network needed more than one newsgroup, each system would put "TRI.ALL" in their .sys file instead of "triangle". Since newsgroups ending in "ALL" subscribe to any newsgroup with the same prefix, all newsgroups beginning with "TRI." would be transmitted throughout the triangle subnetwork; however, each newsgroup in the triangle network should have a separate entry in the .ngfile since newsgroups containing "ALL" are not recommended for .ngfile.

4.5. Setting up news to use mail

Suppose systems Able and Charlie wish to exchange news, and that Able and Charlie talk to each other only through Baker, a system which, alas, does not run news. Able can still exchange news with Charlie using the transmission field in the .sys file to transmit news via *mail(1),* which understands indirection. System Able would take the following steps (and Charlie would take parallel action): First, Able's .sys line for Charlie should be

```
Charlie:NET.ALL,to_Charlie::sed -e "s/^/N/"|mail Baker!Charlie!news
```

The fourth field on the line is run to transmit news to system Charlie by putting an 'N' on the front of every line of the article and mailing it to Charlie!news. Able sets up to receive news mailed from Charlie by adding an entry in /etc/passwd for the user "news" and arranges to run the following shell script once an hour or so.

```
(while true
do
        echo d
done) 2>&1 | mail -f /usr/spool/mail/news | uurec
```

This script pipes all the mail that "news" has received into *uurec,* a utility program distributed with *news. Uurec* reads a series of articles, splits them apart, removes the "From" lines, edits the author’s path line to reflect the systems through which the article was mailed, strips off the ’N’ on the front of all the article lines, and passes the result to *rnews.* If news is mailed across a non uucp network, then uurec.c may have to be modified to correctly process the header information supplied by the mail programs on other networks.

5. Security considerations.

It is easy to fake the origin of an article, since *rnews* believes any syntactically valid article presented as standard input. The only cure for this is a public key encryption scheme.

Since *news* runs set-user-id, it should not use a powerful uid. If *news* ran as "bin" and a security hole were found in news, the entire system would be compromised.

6. Unimplemented features.

7. Read and write permissions on newsgroups.

Newsgroups should be protected from excessive junk news by a permission scheme. However, junk news has yet to be a problem, and no elegant and general architecture for newsgroup permissions has been proposed.

8. Per-user bit map.

News should have a bit map which indicates which articles each user has yet to read. This would allow *news* to permit the user to leave articles around in an undisposed state.

9. Activation and cancellation dates.

News articles should have activation and cancellation dates. This would relieve users of the need to submit and cancel articles at specific times. Use of *at(1)* and *calendar(1)* is recommended until such time as *news* supports these features.

10. Remote cancellation of articles.

When an article is cancelled by it author, he or she should have to option of having news send out a cancellation message which would cause remote systems to cancel the article. However, they may not want it cancelled, which is a problem.