USENET Version B Installation

Matt Glickman
Computer Science Division
Department of Electrical Engineering and Computer Sciences
University of California
Berkeley, California 94720

Revised by Mark Horton for version 2.10
Revised by Rick Adams for version 2.10.3

1. Introduction

This document is intended to help a USENET site install and maintain the network news software. Please ask questions of Rick Adams†; such questions will help to point out areas that need to be addressed here.

The overall order of things to do is:

(a) Find somebody to link up with. You need a network connection of some kind, for example, ARPANET or UUCP. If you must use UUCP and have no connections, you must have at least a dialup and preferably a dialer, and find someone willing to call your machine. The USENET directory may be helpful in finding some other site geographically near yours to hook up to.

(b) Create a *localize.sh* script to make local changes to the makefile and *defs.h* files. (Section 2 gives more details about creating *localize.sh*.) Once you're finished editing *localize.sh*, create a *defs.h* and *Makefile* tailored for your site with the command

sh localize.sh

Inspect *defs.h* and *Makefile* to ensure that all your local customizations got into your final versions. If you saw a ''?'' when you ran *localize.sh*, one or both of the files is certainly wrong. It's a good idea to anchor the patterns in *localize.sh*'s *ed*(1) scripts, especially in its *Makefile*-editing lines. For instance, use **/^UUXFLAGS/** instead of **/UUXFLAGS/**.

(c) Compile the software using the *make*(1) command.

(d) *Su*(1) and type ''make install''. This will copy the files out to the right place and make directories containing most of the important files. It will configure you in with a connection to *oopsvax* via UUCP links. This is undoubtedly wrong, so you will have to configure links as needed. If you are upgrading from a version older than 2.10.3, do ''make update''. This will cause various checks to be performed on important files in **LIBDIR**. The results will be reported to you. If you are not sure if you should do ''make update'', do it. It will not hurt anything if you have already done it.

(e) After editing the configuration table, get your contact at the other end of the link to add you to their netnews *sys* file.

(f) Post a message to the **to.*sysname*** newsgroup which should be set up to go only to the site you are linked to, as a test. Have the other person send a message to your system using the same mechanism. If this doesn't work, find the problem and fix it. (Please don't use **net.test** unless there is no alternative. It is almost always possible to use **test**, or **to.*sysname*** or some ***local*.test** group, instead of **net.test**.)

† ARPANET: rick@seismo.CSS.GOV, UUCP: seismo!rick

(g) Fill out a USENET directory form (the file *dirform* in the *misc* directory). Post a copy to the USENET newsgroup **net.news.newsite** and mail a copy to *cbosgd!uucpmap*.

(h) Format the document *‘‘How to Read the Network News’’* (the file *howto.mn* in the *doc* directory), the document *‘‘How to Use USENET Effectively’’* (the file *manner.mn* in the *doc* directory) and the document *‘‘Copyright Law’’* (the file *copyright.mn* in the *doc* directory) and post them to your **general** newsgroup with a long expiration date. You can use *inews*(1) or *postnews*(1) to do this.

(i) It will probably be necessary to fix your uucp commands to allow *rnews* and to support the **–z** and **–n** options (if you are lucky enought to have the source).

2. Installation

2.1. Configuration

Local configuration of the USENET version B software requires you to edit a few files. Most importantly, the *defs.h* and *Makefile* files must be created from their templates *defs.dist* and *Makefile.dst*. You should create a shell script called *localize.sh* which copies the files and makes local changes to the copies. Even for a completely vanilla site, some changes will be necessary. For example, your script should start with *localize.v7* or *localize.usg*. You should include the name of the local organization (**MYORG**) and the uid of the local news super user (**ROOTID**). You should also choose how your hostname will be determined. If you are a USG site, define **UNAME** in *defs.h*. If you are running 4.[23] BSD, define **GHNAME** in *defs.h*. If you have your UUCP name in */etc/uucpname*, define **UUNAME** in *defs.h*. Otherwise, news will look in the file */usr/include/whoami.h* for a line of the form

#define sysname your-sysname

If you are running System 3 or System 5, you are a USG site. Otherwise, unless you are in AT&T, you are probably a V7 site. The previously mentioned defines are the only modifications that are *necessary* to install news at your site. However, you will probably want to change some of the ones listed below. If your compiler does not accept ‘‘(void)’’, the simplest thing to do is add ‘‘–Dvoid=int’’ to the **CFLAGS** line in the *Makefile*.

A sample localize shell script can be found in *localize.sample*. The most important parameters are:

2.1.1. ROOTID

The numerical uid of the person who is the news super user. This should not be set to 0. Normally it is set to the uid of the news contact person for the site. If it is not defined, the uid of **NOTIFY** will be looked up in */etc/passwd* and used instead.

2.1.2. N_UMASK

Mask for *umask*(2) system call. Set it to something like 022 for a secure system. Unsecure systems might want 002 or 000. This mask controls the mode of news files created by the software. Insecure modes would allow people to edit the files directly.

2.1.3. DFLTEXP

The default number of seconds after which an article will expire. Two weeks (1,209,600 seconds) is the default choice. If you wish to expire articles faster than two weeks, it is recommended that you use the **–e** flag to expire instead of decreasing **DFLTEXP**.

2.1.4. HISTEXP

Articles which were posted more than **HISTEXP** ago are considered too old and are moved into the junk directory. This is because they are too old to be in the history file, so it is impossible to tell if they really should be accepted or are endlessly looping around the network. (This was theoretically possible before this feature was added.) The articles are removed after **DFLTEXP** seconds, but a copy of their ‘‘Message-ID’’ is kept in the history file for **HISTEXP** seconds (the default is 4 weeks).

2.1.5. DFLTSUB

The default subscription list. If a user does not specify any list of newsgroups, this will be used. Popular choices are **all** and **general**,**all.general**.

2.1.6. TMAIL

This is the version of the Berkeley *Mail*(1) program that has the −**T** option. If left undefined, the −**M** option to *readnews*(1) will be disabled.

2.1.7. ADMSUB

This newsgroup (or newsgroup list) will always be selected unless the user specifies a newsgroup list that doesn't include **ADMSUB** on the command line. That is, as long as the user doesn't use the −**n** flag to *readnews* on the command line, **ADMSUB** will always be selected. This is usually set to **general**. (The intent of this parameter is to have certain newsgroups which users are required to subscribe to. A typical site might require **general**.)

2.1.8. PAGE

The default program to which articles should be piped for paging. This can be disabled or changed by the environment variable **PAGER**. If you have it, the Berkeley *more*(1) command should be used, since the **+** option allows the headers to be skipped.

2.1.9. NOTIFY

If defined, this character string will be used as a user name to send mail to in the event of certain control messages of interest. (Currently these are **newgroup**, **rmgroup**, **sendsys**, **checkgroups**, and **senduuname**.) As distributed, mail will be sent to user *usenet*. It is recommended you create such a mailbox (have it forwarded to yourself) if possible, since this makes it easier for another site to contact the site administrator for your site. If you are unable to do this (*e.g*., you are not the super user) you should change this name to yourself. Also, messages about missing or extra newsgroups are mailed to this user by the **checkgroups** control message.

2.1.10. DFTXMIT

This is the default command to use to transmit news if no explicit command is given in the fourth field of the *sys* file. It normally includes *uux*(1) with the −**z** option. You should install this modification to UUCP at once; otherwise your users will start being bombarded with annoying *uux* completion messages. However, you can turn this off to get news installed.

2.1.11. UXMIT

This is the default command used if the **U** flag is present in the flags portion of a *sys* file line. In this case, the second ‘‘%s’’ refers to the name of a file in the news spool area, not a temporary file. It can usually only be used when local modifications are made to the uucp system, such as the −**c** option to *uux*.

2.1.12. DFTEDITOR

This is the full path name of the default editor to use during followups and replies. It should be set to the most popular text editor on your system. As distributed, *vi*(1) is used.

2.1.13. UUPROG

If this is defined, it will be used as a command to run when the **senduuname** control message is sent around. Otherwise the command *uuname*(1) will be run. Normally, this program should be placed in **LIBDIR**.

2.1.14. MANUALLY

If this is defined, incoming **rmgroup** messages will not automatically remove the group. News will instead mail a message to **NOTIFY** advising that the group should be removed. If you define **MANUALLY**, you should have **NOTIFY** defined. **MANUALLY** is defined by default to protect you against accidental or malicious removal of an important newsgroup.

2.1.15. NONEWGROUPS

If this is defined, incoming **newgroup** messages will not automatically create the group. News will instead mail a message to **NOTIFY** advising that the group should be created. If you define **NONEWGROUPS**, you should have **NOTIFY** defined. **NONEWGROUPS** is undefined by default to make it easier to automatically maintain the news system.

2.1.16. BATCH

If set, this is the name of a program that will be used to unpack batched articles (those beginning with the character ‘‘#’’.) Batched articles normally are files reading

```
#! rnews 1234
article containing 1234 characters
#! rnews 4321
article containing 4321 characters
. . .
```

Batching is *strongly* recommended for increased efficiency on both sides.

2.1.17. LOCALNAME

Most systems have a full name database on line somewhere, showing for each user what their full name is. Most often this is in the gecos field of */etc/passwd*. If your system has such a database, **LOCALNAME** should be left undefined. If not, define **LOCALNAME**, and articles posted will only receive full names from local user information specified in *NAME* or ***$HOME***/*.name* by the user. If you have a nonstandard gcos format (not *finger*(1) or RJE) it will be necessary to make local changes to *fullname.c* as appropriate on your system.

2.1.18. INTERNET

If your system has a mailer that understands ARPA Internet syntax addresses (‘‘user@site.domain’’) turn this on, and replies will use the ‘‘From’’ or ‘‘Reply-To’’ headers. Otherwise, leave it disabled and replies will use the ‘‘Path’’ header.

2.1.19. MYDOMAIN

When generating internet addresses, this domain will be appended to the local site name to form mailing address domains. For example, on system *ucbvax* with user *root*, if **MYDOMAIN** is set to ‘‘.UUCP’’, addresses generated will read ‘‘root@ucbvax.UUCP’’. If **MYDOMAIN** is ‘‘.Berkeley.EDU’’, the address would be ‘‘root@ucbvax.Berkeley.EDU’’. If your site is in more than one domain, use your primary domain. The domain always begins with a period, unless the local site name contains the domain; in this case **MYDOMAIN** should be the null string.

2.1.20. CHEAP

Do not *chown*(1) spool files to *news*. This will cause the owner of the file to be the person that started the *inews* process. This is used for obscure accounting reasons on some systems.

2.1.21. OLD

Define this if any of your USENET neighbors run 2.9 or earlier versions of B news. It will cause all headers written to contain two extra lines, ‘‘Article-I.D.’’ and ‘‘Posted’’, for downward compatibility. Once all your neighbors have converted, you can save disk space and transmission costs by turning this off. It is strongly encouraged that they convert. 2.10.3 is *much* faster than 2.9. The performance difference is dramatic.

2.1.22. UNAME

Define this if the *uname*(2) system call is available locally, even though you are not a USG system. USG systems always have *uname*(2) available and ignore this setting.

2.1.23. GHNAME

Define this if the 4.[23] BSD *gethostname*(2) system call is available. If neither **UNAME** or **GHNAME** is defined, *inews* will determine the name of the local system by reading */usr/include/whoami.h*.

2.1.24. UUNAME

Define this if you keep your UUCP name in */etc/uucpname*.

2.1.25. V7MAIL

Define this if your system uses V7 mail conventions. The V7 mail convention is that a mailbox contains several messages concatenated, each message beginning with a line reading ‘‘From *user date*’’ and ending in a blank line. If this is defined, articles saved will have these lines added so that mail can be used to look at saved news.

2.1.26. SORTACTIVE

Define this if you want the news groups presented in the order of each person’s *.newsrc*(5) instead of the active file.

2.1.27. ZAPNOTES

Define this if you want old style notesfile id’s in the body of the article to be converted into ‘‘Nf-Id’’ fields in the header.

2.1.28. DIGPAGE

If this is defined, *vnews*(1) will attempt to process the subarticles of a digest instead of treating the article as one big file.

2.1.29. DOXREFS

Define this if you are using *rn*(1). *Rn* uses this option to keep from showing the same article twice.

2.1.30. MULTICAST

If your transport mechanism supports multi-casting of messages, define this. Currently ACSNET is the only network that can handle this.

2.1.31. BSD4_2

Define this if you are running 4.2 or 4.3 BSD UNIX†.

2.1.32. BSD4_1C

Define this if you are running 4.1C BSD UNIX.

2.1.33. SENDMAIL

Use this program instead of *recmail*(8) for sending mail.

2.1.34. MMDF

Use MMDF instead of *recmail* for sending mail.

†UNIX is a trademark of AT&T Bell Laboratories.

2.1.35. MYORG

This should be set to the name of your organization. Please keep the name short, because it will be printed, along with the electronic address and full name of the author of each message. Forty characters is probably a good upper bound on the length. If the city and state or country of your organization are not obvious, please try to include them. If the organization name begins with a ‘‘/’’, it will be taken as the name of a file. The first line in that file will be used as the organization. This permits the same binary to be used on many different machines. A good file name would be */usr/lib/news/organization.* For example, an organization might read ‘‘AT&T Bell Labs, Murray Hill’’, ‘‘U.C. Berkeley’’, ‘‘MIT’’, or ‘‘Computer Corp. of America, Cambridge, Mass’’.

2.1.36. HIDDENNET

If you want all your news to look like it came from a single machine instead of from every machine on your local network, define **HIDDENNET** to be the name of the machine you wish to pretend to be. Make sure that you have you own machine defined as *ME* in the sysfile or you may get some unnecessary article retransmission.

2.1.37. NICENESS

If **NICENESS** is defined, *rnews* does a *nice*(2) to priority **NICENESS** before processing news.

2.1.38. FASCIST

If this is defined, *inews* checks to see if the posting user is allowed to post to the given newsgroup. If the username is not in the file **LIBDIR**/*authorized* then the default newsgroup pattern in the symbol **FASCIST** is used.

The format of the file *authorized* is:
user:allowed groups

For example:

root:net.all,mod.all
naughty_person:junk,net.politics
operator:!net.all,general,test,mod.unix

An open environment could have **FASCIST** set to **all** and then individual entries could be made in the authorized file to prevent certain individuals from posting to such a wide area.

Note that a distribution of **all** does *not* mean to allow postings only to local groups – **all** includes **all.all**. Use **all**,!**all.all** to get that behavior

2.1.39. SMALL_ADDRESS_SPACE

Define this if your machine has 16 bit (or smaller) pointers. If you are on a PDP-11†, this is automatically defined.

2.2. Makefile

There are also a few parameters in the *Makefile* as well. These are:

2.2.1. OSTYPE

This is the type of UNIX system you are using. It should be either **v7** or **USG**. Any BSD system is v7. Any System 3 or System 5 system is USG. This is normally set by *localize.sh*.

2.2.2. NEWSUSR

This is the owner (user name) of *inews*. If you are a superuser, you should probably create a new user id (traditionally *news*) and use this id. If you are not a superuser, you can use your own user id. If you are able to, you should create a mail alias *usenet* and have mail to this alias forwarded to you. This will make it easier for other sites to find the right person in the presence of changing jobs and out of date or nonexistent directory pages. **NEWSUSR** and **ROOTID** do not need to represent the same user.

---

†PDP-11 is a trademark of Digital Equipment Corporation.

2.2.3. NEWSGRP

This is the group (name) to which *inews* belongs. The same considerations as **NEWSUSR** apply.

2.2.4. SPOOLDIR

This directory contains subdirectories in which news articles will be stored. It is normally */usr/spool/news.*

Briefly, for each newsgroup (say **net.general**) there will be a subdirectory */usr/spool/news/net/general* containing articles, whose file names are sequential numbers, *e.g.*, */usr/spool/news/net/general/1,* etc.

Each article file is in a mail-compatible format. It begins with a number of header lines, followed by a blank line, followed by the body of the article. The format has deliberately been chosen to be compatible with the ARPANET standard for mail documented in RFC 822.

You should place news in an area of the disk with enough free space to hold the news you intend to keep on line. The total volume of news in **net.all** currently runs about 1 Mbyte per day. If you expire news after the default 2 weeks, you will need about 14 Mbytes of disk space (plus some extra as a safety margin and to allow for increased traffic in the future.) If you only receive some of the newsgroups, or expire news after a different interval, these figures can be adjusted accordingly.

2.2.5. BATCHDIR

This directory will contain the list of articles to send to each system. It is normally */usr/spool/batch.*

2.2.6. LIBDIR

This directory will contain various system files. It is normally */usr/lib/news.*

2.2.7. BINDIR

This is the directory in which *readnews*, *postnews*, *vnews*, and *checknews*(1) are to be installed. This is normally */usr/bin.* If you decide to set **BINDIR** to a local binary directory, you should consider that the *rnews* and *cunbatch* commands must be in a directory that can be found by *uuxqt*, which normally only searches */bin* and */usr/bin.*

2.2.8. UUXFLAGS

These are the flags *uux* will be called with.

2.2.9. LNRNEWS

This is the program used to link *rnews* and *inews*. If you have symbolic links, you can replace the ‘‘ln’’ with ‘‘ln −s’’.

2.2.10. SCCSID

If this is defined, sccs ids will be included in each file. If you are short on address space, don’t define this.

3. FILES

This section lists the files in **LIBDIR** and comments briefly what they do.

3.1. active

A list of active newsgroups. It is automatically updated as new newsgroups come in. The order here is the order news is initially presented by *readnews*, so you can edit this file to put important newsgroups first. If you have **SORTACTIVE** defined, after the first time the user invokes *readnews*, it will be presented in the order of his *.newsrc*. Each line of the active file contains four fields, separated by a space: the newsgroup name, the highest local article number (for the most recently received article), the lowest local article number that has not yet expired, and a single character used to determine if the user can post to that newsgroup. If the character is ‘‘y’’ the user is permitted to post articles to that group. If the character is ‘‘n’’ the user is not permitted to post articles to that groups. (This field takes the place of the *ngfile* in earlier versions of news. Local article numbers begin at 1 and count sequentially within the newsgroup as articles are received. They do not usually correspond to local article numbers on other sites. The article numbers are always stored as a five digit number (with leading zeros) to allow updating of the file in place.

The active file should contain **all** active net-wide active newsgroups (**net.all**and **mod.all**). It is important that they all be present, as they are used as a check for valid newsgroup names and invalid newsgroup names are removed from any articles processed by *inews*. You should use the *sys* file to keep out unwanted newsgroups.

3.2. aliases

This file is used to map bad newsgroup names to the correct ones. (For example, **net.unix.wizards** is mapped into **net.unix-wizards**). Each line consists of two fields separated by a space. If the first field is found in the newsgroup list of the incoming article, it is changed to the second field. This change takes place in the article before it is passed on to other systems, not just locally.

3.3. batch

This program reads a list of filenames of articles and outputs the articles themselves. It is typically used by the shell script *sendbatch*.

3.4. c7unbatch

This is used to decompress news that has been **encoded** for transmission over a network that only supports 7-bit transfers (e.g X.25.)

3.5. caesar

This is a program to do Caesar decoding of rotated text, on a line by line basis. The standard input is copied to the standard output, rotating each line according to a static single letter frequency table. If an integer argument is given (*e.g*., 13), every line is rotated by that argument, without regard to letter frequencies. This program is invoked by the **D** *readnews* command. It is also used by *postnews* with the ‘‘13’’ argument to encode selected material for posting.

3.6. checkgroups

Checkgroups is a shell file to aid in automatically checking the accuracy of your active file. It is executed by the **checkgroups** control message and mails a list of out of date newsgroups to the person defined by **NOTIFY** It also updates the *newsgroups* file that is used by *postnews* as a helpfile for newsgroup selection.

3.7. compress

This program does a modified Lempel-Ziv data compression. It is used by the compressed batching scheme. It averages 50% compression on a typical batch of news.

3.8. distributions

This is a list of distributions that are valid for your site. Each line has two fields separated by the first space on the line. The first field is the name of the distribution (*e.g*., **usa**, **na**, etc.). The second field is text describing the distribution. As distributed, this file is only correct for sites in the USA. You should examine this file and add or delete the appropriate distributions.

3.9. encode

This program transforms an 8-bit binary file into a file suitable for sending over a link that only allows 7-bit characters. It is used by **sendbatch -c7.**

3.10. errlog

This file contains the ‘‘important’’ error messages found in the log file. These errors usually indicate that something was wrong with an article. This file should be watched closely. The *log* file contains much more verbose information and it is often difficult to detect errors in it.

3.11. expire

This program expires old articles and archives them if archiving is selected. It is typically run once a day from *cron*(8).

3.12. help

This contains a list of commands printed when an illegal command is typed to *readnews*.

3.13. history

A list of every article that has come in to your system. It is used to reject articles that come in for the second time (presumably via a different path). This file will grow but is cleaned out by the *expire*(8) command.

3.14. history.d

On USG systems, this directory contains 10 files (history.[0-9]) which are used as part of a simple hashing algorithm to speed up history searches. Since V7 systems have DBM, this is not used on V7 systems.

3.15. history.dir,history.pag

These two files are used on V7 systems as a hashed version of *history*, containing the message id's of all articles in history. They are only used if **–DDBM** and **–ldbm** appear in *Makefile*.

3.16. inews

This is the program that actually sends and receives news. All other programs interface eventually with it. It is not intended to be used directly by a human, so it is no longer in */usr/bin*.

3.17. log

If present, a log of articles processed and error conditions is kept here. This file grows without limit unless cleaned out periodically. The *trimlib* script in *misc* can be invoked from *cron* daily or weekly to keep the log short.

3.18. moderators

This file contains a list of the moderators and their mailing addresses for each moderated newsgroup. Each line consists of two fields. the first is the name of the moderated group. The second is the mailing address of the group's moderator. As distributed, they are almost certainly wrong. You will need to modify the paths so they work from your site.

3.19. newsgroups

This file is displayed by *postnews* when a user hits **?** in response to its request for newsgroups. It is also used by *vnews* when it displays the newsgroup name. It is updated automatically by the **checkgroups** control message.

3.20. notify

If this file is present, its contents will be taken as the name of the user to notify in case of a problem. If the file is empty, nobody will be notified. (This overrides the **NOTIFY** option in *defs.h*). Having a null file is useful if one person administers several systems and does not want multiple copies of control message notifications.

3.21. oactive, ohistory, ohistory.dir, ohistory.pag

These are copies of the corresponding *active*, *history*, *history.dir*, and *history.pag* files before *expire* ran. They are kept in case something happens to the originals.

3.22. recmail

This program can serve as a link between news and your local mailer. If you have *sendmail*(8), don't use *recmail*. *Sendmail* is much more useful.

3.23. recnews

A program which allows you to send mail to get news posted. You usually need to run *sendmail* or *delivermail*(8) to be able to use this.

3.24. recording

A list of newsgroup classes and filenames to display recordings for. The recording feature is analogous to the recordings played in some areas when you dial directory assistance, trying to be annoying and make you think twice. Recordings on certain newsgroups are intended to remind the user of the rules for the newsgroup, or, in the case of a company worried about letting proprietary information out, reminding authors that anything they say is seen outside the company and so proprietary information should not be included.

The file contains one line per recording. The line contains two fields, separated by a space. The first field is the newsgroup class (*e.g.*, **net.all**), the second field is the name of the file containing the recorded message. If the file name does not begin with a slash, it will be searched for in **LIBDIR**. Sample recording files can be found in the *misc* directory.

3.25. rmgroup

This shell file should be used to remove any groups that are no longer used.

3.26. sendbatch

This shell file is used to send batched articles to other systems. It is typically run from *cron*. See the manual page for more details.

3.27. sendnews

A program to send news internally from one computer to another. It is useful if you must use mail links to transmit articles.

3.28. seq

This file contains the current sequence number for your system. It is used to generate unique article id's.

3.29. sys

This file contains a list of all your neighbors, which newsgroups they get, and how to send news to them. The format is documented below.

3.30. unbatch

This program is used to unbatch the incoming batched news and feed each article to *inews*. It's horrible and will go away in the future.

3.31. users

A list of users that have read news on your system.

3.32. uurec

A program to receive news sent by *sendnews*(8).

3.33. vnews.help

This is the helpfile used by *vnews*.

4. Setting Up Links

There are two basic types of links for exchanging news: those that use mail and those that don't. The ones that use mail are more indirect, yet more versatile, while the ones that don't are simpler. The default method does not use mail, so that is discussed first.

4.1. Non-mail Links

The basic theory behind a non-mail link is that the *rnews* program is invoked on the remote system with the article being transmitted as the standard input. This is possible on several networks, but the most common implementation is via the UUCP network. Using the *uux* command, the command which is forked to the shell looks like:

uux – –r –z remotesys!rnews < article

This is the default transmission method. In order to set up such a link, obviously a UUCP link with the remote system must be in effect. In addition, *rnews* must be available and executable by *uuxqt* on the remote machine. In most cases, this means that *rnews* must be in */usr/bin* so *uux* can find it. Also, the list of allowed UUCP commands (in */usr/src/usr.bin/uucp/uuxqt.c* or */usr/lib/uucp/L.cmds*, depending on the version of UUCP) should be checked to make sure that *rnews* is an allowed command.

Other networks that allow remote execution include the BERKNET, BLICN (*usend*(1)), many Ethernets, and the NSC hyperchannel (*nusend*(1)). It is important, however, that a spooling mechanism be available. Otherwise, if system *A* tries to send an article to system *B* via a remote execution command, and *B* is down, the article could be lost. Spooling arranges that the system will try again when *B* comes back up.

4.2. Mail Links

When using mail to transmit articles, two intermediary programs are necessary. These are *sendnews* and *uurec*(8). The idea is that when system *A* wants to send an article to system *B*, the *sys* file on system *A* has an entry for system *B* such as:

/usr/lib/news/sendnews –a rnews@B

which runs *sendnews* on the article. The **–a** option specifies that the mail should be formatted for the ARPANET. *Sendnews* packages the article and mails it to ‘‘rnews@B’’. Somehow, the B system is expected to make sure that all mail to user ‘‘rnews’’ is fed as input to the program *uurec*. This program unpackages it and invokes *rnews*.

The best way to get mail to ‘‘rnews’’ fed into *uurec* is to use *sendmail* or *delivermail*, if you are on a system running them. Create an alias in */usr/lib/aliases* as follows:

rnews: "|/usr/lib/news/uurec"

and *sendmail* will handle it. If you do not have a facility for forwarding mail to a program, you can gimmick your mailer to watch for it (using *popen*(3S), this is easy) or, if you don’t want to do any programming, you can have *cron* invoke *uurec* every hour with */usr/spool/mail/rnews* as standard input. This solution is messier because *uurec* must potentially deal with multiple messages, something that has never been tested.

5. Format of the *sys* file

To set up a link to another site, edit the *sys* file in **LIBDIR**. This file is similar to the *L.sys* file of UUCP. Each line contains four fields, separated by colons:

(1) The system name of a site to which you forward news. Normally all systems you have links to will be included. You should also have a line for your own system. If this field is *ME*, it will be used as if it were your local system name. If the system name is followed by a ‘‘/’’, the article will not be forwarded to this system if it has already passwd through any of the (comma separated) list of sites immediately following the ‘‘/’’. For example, if the sysline was:

yoursite/sitea,siteb,sitec:net,mod,na,usa,to.yoursite::

the incoming article would only be forwarded to *yoursite* if it had not already been to any of *sitea*, *siteb*, or *sitec*. This is normally used to reduce the number of duplicate articles received at a site that has multiple main newsfeeds.

(2) The newsgroups to be forwarded to them. This is a pattern of the same kind as a subscription list. Generally, you will list classes of newsgroups, that is, using **all** for everything. A typical forwarding list for a new site would be

net,mod,na,usa,to.*sysname*

where *sysname* is the name of the remote system. (Of course, if you are not in the USA or North America, you would remove those distributions and replace them with the ones appropriate for you). In particular, you don't want to forward **all** since local newsgroups (those without dots) should not be sent. For the line describing your own system, this field describes the newsgroups your site will accept from remote sites. Thus, if another site insists on sending you a newsgroup you don't want, for example **net.jokes**, include **!net.jokes** here.

(3) This field contains flags describing the connection. An **A** will indicate that the other site is running an A version of netnews. A **B** indicates a B version. Leaving it empty defaults to **B**. If you are reading this document, you have a B version. Some existing sites run A versions. If you aren't sure, ask your contact at the other site, with whom you should be talking to set this up anyway. The **F** flag indicates that the fourth field is the name of a file. The full path name of a file containing the article in **SPOOL** will be appended to this file. The **L** flag prevents transmission unless the article was created on this site. If a number follows the **L** (*e.g.*, **L3**), sites less than that number of hops away will be considered local. (It is recommended that you feed an **L** link to a backbone site, to ensure that your submissions will be more likely to get to the entire network, even in the event of a local problem. Please make sure that a mail link exists too, so you can get replies.) The **N** flag can also be included here, indicating that mail should be sent using the *ihave/sendme* protocol described below. The **H** flag can be used to interpolate the history file into the command. The **S** flags says to execute the transmission command directly instead of forking a shell. The **U** field arranges that the parameter to the optional ''%s'' in the command field to be filled in with a permanent file name from **SPOOL** instead of a temporary customized file name. The **M** flag says to use multi-casting. Multi-casting is described in an appendix.

(4) This field is the command to be run to send news to the remote site. The article will be on the standard input. Leaving this field blank means an ordinary UUCP link is being used, that is, the command defaults to

uux − −r −z sysname!rnews

The − option tells *uux* to expect input from the standard input. The **−z** option is nonstandard − you should add it (see the *minus.z** files in the uucp source directory.) It shuts off the annoying message you would otherwise get mailed to you telling you that your article was broadcast successfully. To avoid using the **−z** option, change the source or put the *uux* command in the fourth field. The **−r** option tells *uux* not to call the other system once the job is queued. This turns out to ease the load on the system, at the expense of making news be transmitted a bit slower. The news will be sent when the next call is made; usually this means the next time mail is sent to or from your system. If this turns out to be unreasonably long, put a line in *crontab* to run

/usr/lib/uucp/uucico −r1 −ssystem

every hour or so.

Here is a sample *sys* file for a site *myvax* with connections to *yourvax* where *myvax* also passes news on to *downstream*. We assume that *myvax* and *downstream* exchange a local newsgroup class **lng.all** as well as the network wide newsgroups. News to *downstream* is batched. We also assume that *myvax* and *yourvax* are in the USA, while *downstream* is in Canada.

```
myvax:net,mod,na,usa,lng,to::
yourvax:net,mod,na,usa,to.yourvax::
downstream:net,mod,na,lng,to.downstream:F:/usr/spool/batch/downstream
```

6. Posting Methods

The basic method is *postnews*. This program will prompt you for the title, newsgroups, and distribution, then place you in the editor. (The system default **EDITOR** is used unless the environment variable **EDITOR** is set, overriding the system default.) The text should be typed after the blank line. The title and newsgroups are available for editing at the top of the buffer. Other header lines can be added, such as an expiration date or a distribution. When you write out the file and exit from the editor, you will be prompted for what to do next. Your choices are: **w**rite the message to a file, **s**end the message, **l**ist the message or **e**dit it again.

Another method is to use mail. This can only be done on systems that allow mail to a given name to be fed into an arbitrary program as input. This is easily done with the Berkeley *delivermail* or *sendmail* program, and not with any other mailer the author is familiar with. (It may be possible to painfully set this up with MMDF, provided the newsgroup name is no more than 8 characters long.) To use mail, set up an alias such as the following:

net.general: "|/usr/lib/news/recnews net.general"

Whenever a user sends mail to **net.general**, this starts up the given shell command which calls **recnews** with one argument, the name of the newsgroup. You need to create one alias for each newsgroup, and to keep the list up to date as new newsgroups are created. *Recnews*(8) will in turn invoke *inews*.

Note that there are problems with *recnews*. There is no way to use it to post to multiple newsgroups without creating separate articles (something frowned upon because it forces people to read the same thing more than once.) Also, there is no way to make the recording feature (to remind people to not accidently divulge proprietary information) work when recnews is used.

7. Various considerations

7.1. Setuid bits

The current intended state of affairs is that *inews* runs setuid to **NEWSUSR**. The *readnews* program does not need to be setuid. This makes it possible to write your own interface to read news instead of using *readnews*. (As distributed, *inews* is also setgid. I know of no good reason for this.)

7.2. Modes of Spool Directories

All the files should be writable by **NEWSUSR**. However, due to a glitch, you will probably have to make the **SPOOLDIR** and its subdirectories mode 777. It could be 755 except for one problem. When a new newsgroup comes in, *inews* will attempt to *mkdir*(1) a new subdirectory of **SPOOLDIR** for the newsgroup. Since both *inews* and *mkdir* are setuid, *mkdir* will use the uid of the person who ran *inews* instead of **NEWSUSR** when checking for permissions. If the directory mode isn't 777 the check will fail. Here are several alternatives if you don't want a 777 directory around:

7.2.1. Fix Real Uid

If *inews* is always run by *cron* or as *root*, the real uid can be arranged to be *root* or **NEWSUSR**. This is a poor solution since it makes the local creation of new newsgroups require super user permissions, and is a potential security hole. If this approach is taken, care must be taken to insure that the owner of the created directory is **NEWSUSR**.

7.2.2. Change the Kernel

Inews will do: **setuid(geteuid())** (see *setuid*(2) and *geteuid*(2)) before it forks the *mkdir*. If your system permits this call, there will be no problem. In particular, Berkeley 4.0 UNIX and later systems allow this. An alternative change to the kernel is to automatically stack uids: when a setuid program is run, set the new real uid to the old effective uid.

7.2.3. Groups

You could have *inews* be setgid to **NEWSGRP** and all files writable by the group. This approach has been tested and the problem turns out to be that the *mkdir* command uses the *access*(2) system call to check permissions. Since *access* uses the real gid, you run into the same problem.

7.2.4. Another *Mkdir*

You could create a version of *mkdir* that does less checking and put it in a directory that can only be accessed by **NEWSUSR** (mode 700, owned by **NEWSUSR**). Have *inews* fork this *mkdir*.

7.3. Expiration dates

To get articles to expire automatically, put a line in *crontab* to run

/usr/lib/news/expire

every night. This command deletes all expired news. The **−a** *newsgroups* option causes all expired news to be archived under */usr/spool/oldnews* depending on which newsgroups are selected. (See *expire*(8) for details.)

Sometimes news is not expired when it should be. Be sure to check that *expire* has permissions to unlink files, and that it is properly setuid to **NEWSUSR**. You can manually invoke *expire* with the **−v** (verbose) option to find out what it's doing. Adding levels of verbosity (*e.g.*, **−v6**) will get more and more output.

7.4. Version to Version

Version B will understand incoming news in either version A or B format, automatically (presuming **OLD** is defined in defs.h.) Version B will generate either format, depending on the flag in the third field of the *sys* line. Version A will not understand version B format. Thus, it is possible for two version B sites to communicate using version A format. This will work but is not a good idea, since the translation from B to A loses information (such as the expiration date) which will not be there when translated back to version B.

News from versions A and 2.9 B do not conform to the USENET interchange standard. 2.10 B supports the standard and will communicate with either A or 2.9 B news. A news is written (losing other header information) if A is in the flags for the system. If **OLD** is defined, 2.10 will write out headers with both standard (‘‘Date’’ ‘‘Message-ID’’) and 2.9 (‘‘Posted’’ ‘‘Article-I.D.’’) lines so that either B system will properly handle the article. Incoming news is recognized by the first letter (**A** for A news), or the lack of an ‘‘@’’ in the ‘‘From’’ line (2.9). Missing fields are constructed as well as possible from the available information.

7.5. Presentation Order

The order of the newsgroups listed in ***LIBDIR/active*** is the order the newsgroups will be presented in initially. If **SORTACTIVE** is defined in *defs.h*, after the first time news will be presented in the order of the person's *.newsrc*. Initially this will be directory order, but you can edit important newsgroups like **general** to the top.

A recommended order to maintain your active file in is this:

net.announce.newusers
general
local.general
net.announce
local *newsgroups in alphabetical order*
mod.all *newsgroups in alphabetical order*
net.all *newsgroups in alphabetical order*
test
all.test
to.all
control
junk

8. Control Messages

Some news systems will send you articles that are not for human consumption. They are messages to your news system called *control messages*. Such messages contain the ‘‘Control’’ header. Older systems use newsgroups matching **all.all.ctl**, and this will still work, although the ‘‘Control’’ header is preferred. Since the newsgroup name is used for distribution only, and is not checked to ensure it's in the active file, such newsgroup names can still be used. This makes it possible to post network wide control messages with **net.msg.ctl** (or restricted broadcast such as **btl.msg.ctl**) or messages for a particular system: **to.ucbvax.ctl**. Messages are canceled, however, with a ‘‘Control’’ line in a message to the same newsgroup(s) as the original message.

A control message contains a command and zero or more arguments (much like a UNIX program). The subject of the article contains the command and arguments. The body of the article is usually ignored, although some messages can use it for additional text information. Control messages are not stored in **SPOOL**; rather, they are acted on and discarded at once.

8.1. ihave/sendme

Two control messages are **ihave** and **sendme**. These messages allow two participating sites to set up a link so that one site will tell the other site it has a given article and wait for a request before it actually sends it. The normal case is to send an entire article to a system, which consults the history file to see if the article has already been seen, and then throws it away if it has been seen before.

Note that, since most messages are short anyway, experience has indicated that for ordinary UUCP unbatched communication, all *ihave/sendme* does is triple the load and slow down forwarding. We hope future code will allow **ihave**'s with multiple message id's in the body, and existing code in 2.10 understands such messages, but does not generate them. So we advise that you don't use *ihave/sendme* for now.

Use of these control messages can cut down on this wasted transmission, but if you have a polled UUCP connection, they can slow down receipt of news due to polling delays. It is up to each connected pair of sites whether they want to use this protocol. The choice is controlled by the **N** flag in the *sys* file. In the case of a leaf node (one with only one neighbor) there is no advantage to this protocol. Even if both sites are able to initiate a connection (have dialers or the link is hardwired) the **−r** option on the *uux* can cause 2 hour or more delays in propagating news. Since this protocol can triple the number of messages generated, you should carefully evaluate your situation when deciding whether to use it. If transmission time and phone bills dominate your costs, and you are sending news to several sites, and large article bodies dominate the costs (rather than the headers and the time spent by UUCP negotiating transmission) it is probably worthwhile to use *ihave/sendme*. If your costs are dominated by CPU load from UUCP, or if you send news to a site that cannot get it from anywhere else, you probably do not want to use this protocol. The decision can be made independently for each site in your *sys* file.

This pair works as follows: Site *mysite* receives article ‘‘<123@abc.UUCP>’’. It enters it locally and then broadcasts it to its neighbors. One of its neighbors is site *yoursite* which has the **N** flag in the *sys* file. So *mysite* sends an article on newsgroup **to.*yoursite*.ctl** with title ‘‘ihave <123@abc.UUCP> mysite’’. This control message has two arguments − the first (‘‘<123@abc.UUCP>’’) is the article id of the article in question, the second (‘‘mysite’’) is the name of the site sending the article. The name of the newsgroup and the *sys* file control transmission of the article. Normally the *sys* file will read something like

yoursite:net.all,fa.all,to.yoursite:BN:

which will cause an article on **to.yoursite.ctl** to be transmitted.

Yoursite receives the message and looks to see if it has seen it before. If it has, it throws the message away and stops. If it hasn't, it sends a message on **to.*mysite*.ctl** with title ‘‘sendme <123@abc.UUCP> yoursite’’ which is transmitted to *mysite*. (The two arguments to *sendme* are the article id requested and the site to send it to.) Then *mysite* gets this message and actually transmits the article to *yoursite*.

8.2. newgroup

This message has one argument, the name of a newsgroup to be created. This allows special action to be taken locally when a new newsgroup is created. It is generated by the **−C** option to *inews*. By default, the newsgroup is added to the active file, and mail is sent to the local contact advising that this has happened. The directory will be created when a message for that newsgroup arrives. See the routine ‘‘c_newgroup’’ in *control.c* if you want something different to happen. (Note that, although the body of the message contains a brief description of the purpose of the group, this body is usually thrown away by existing software.)

8.3. rmgroup

This message has one argument, the name of a newsgroup to be removed. It is used for network-wide cancellation of a newsgroup. If **MANUALLY** is not defined, it will remove the articles, directory, and active file line for the group. There is a shell script *rmgroup* that does essentially the same thing as this message, but the shell script only removes the group locally. We recommend that you leave **MANUALLY** defined, and when you receive mail advising you of the demise of the newsgroup, you run *rmgroup* by hand. This will prevent accidental or malicious removal of a good newsgroup.

8.4. cancel

This message cancels a given article. It takes one argument, the message id of the article to cancel. It should be broadcast to the same newsgroup as the original article. If the article to be canceled is not present, the control message will not be propagated to downstream sites.

8.5. sendsys

The *sys* file is mailed to the originator of the message. There are no arguments. This is used for making maps. Since your *sys* file is public information, you should not remove or change this control message.

8.6. senduuname

The *uuname* program is run and the output is mailed to the originator of the message. There are no arguments. This is used for making UUCP maps. If you do not run UUCP or have sites in your *L.sys* which are a secret, you may wish to edit this. Note that only the output of *uuname* is mailed, not the contents of *L.sys* (which news does not have access to anyway). If you do make a change, you should arrange that some mail still is sent out to the originator of the message, so he will know your site received it. See the code in routine ‘‘c_senduuname’’ in *control.c*.

8.7. version

The local version name/number of the netnews software is mailed back to the author of the control message.

8.8. checkgroups

This control message is an attempt at semi-automatic maintenance of the list of active news groups. This control messages takes the body of the article and pipes it into ***LIB****/checkgroups*. As mentioned previously, ***LIB****/checkgroups* will update the newsgroups file, add any missing newsgroups, and mail a message to **NOTIFY** about any old newsgroups that should be removed. It is expected that the person who maintains the list of active newsgroups will broadcast this control message on a regular basis.

8.9. Other Messages

Any unrecognized message will cause an error message to be mailed to the local site administrator. Additional messages may be defined as time goes on, such as messages to automatically update directories or maps. You should be willing to go into the code (*control.c*) and add messages as they become standardized.

9. Maintenance

There are some things you should do periodically to keep your news system running smoothly. We hope to eventually automate all or most of this, but right now some of it must be done by hand.

The *history* and *log* files in your **LIB** directory will grow. You should make sure that they are cleaned up periodically. The ***LIB****/expire* program will remove lines from history corresponding to deleted articles, but it is a good idea to check the file every few months to make sure it is not going wild. Be sure not to completely lose your history file when you clean it up, in case another neighbor tries to send you an article you recently got. (If you only get news from one site it is safe to clean it out completely.)

The log file is not automatically cleaned out by any netnews software, and will grow quickly. The *misc/trimlib* script can be installed in ***LIB****/trimlib*, and invoked weekly by *cron*.

You should also clean out old newsgroups that are no longer active. To remove a newsgroup **net.foo**, you should run the shell script *rmgroup* with **net.foo** as the argument. That is,

/usr/lib/news/rmgroup net.foo

Note that clearing up UUCP constipation is another thing you’ll have to do if you have flaky hardware or phone lines. If you have more than one connection, chances are that UUCP will get clogged up when one of your neighbors goes down for more than a few hours. Various spooling schemes are being worked on to help make the news/uucp system more robust, but one thing you can and should do, if you find your */usr/spool/uucp* directory getting too big, is to install a subdirectory fix to UUCP. A quick and dirty version of this is available from Duke, which traps the file-oriented system calls at the assembly language level and maps, for example, D.fooA1234 into D.foo/D.fooA1234. Since the C. and D.*local* directories still get big, in practice this can still create some big directories, but the directories tend to be a factor of 5 smaller, resulting in a factor of 25 improvement to speed (since a directory traversal for all files is quadratic on UNIX). Right now, UUCP is the weak link in netnews distribution, and you should certainly keep an eye on it.

10. Creating New Newsgroups

As system news administrator, you are able to create newsgroups. To create a newsgroup, first make sure this is the right thing to do. Normally a suggestion is first posted to **net.news.group**,**net.relatedgroup** for a net newsgroup **net.relatedgroup**(should be the group which you are proposing to sub-divide. For instance, to propose creating **net.tv.soaps**, post the original article to **net.tv**,**net.news.group**). Followups are made to **net.news.group** *only*. (You can force this by putting the line:

Followup-To: net.news.group

in the headers of your original posting). If it is established that there is general interest in such a group, and a name is agreed on, then someone creates it by typing the command

inews –C *newsgroup*

This will create the active entry locally. The directory will be created automatically when the first article for that newsgroup is received. It will also prompt you for a paragraph describing the group and start up an *inews* to post a newgroup control message announcing the group. This control message will be sent out on **net.msg.ctl** and other sites may have configured their systems to do something with these messages. A human readable announcement is not made – you can post this to **net.news.group** if necessary.

You must be the super user to use the **–C** option to *inews*. (That is, your uid must match **ROOTID**. It is recommended that you change **ROOTID** to your own uid so you don't have to *su* to create newsgroups.)

11. Conversion from A to B

If you are currently running version A on your system, note that B is incompatible with A. The files are stored in a different format (headers have mail like field names now). The directory organization is different (each newsgroup has a subdirectory of its own, and the file names are numbers rather than *site.id* pairs). There are no *bitmap*, *uindex*, or *nindex* files to be trashed (which articles have been read is stored in each users *.newsrc* file). The user interface is slightly different (news/*netnews*(1) is now called *readnews*, news is posted using *inews*, subscription is done by editing *.newsrc*, the sense of the **–c** option is reversed, news is presented in newsgroup order, the **–a** and **–t** options now probably need **–x** as well, and there are many minor changes).

We decided not to provide a program to convert from version A to version B. Rather, the following strategy was adopted for conversion:

(1) Install the new news in a different spool directory from the old one. For example, you can use */usr/spool/newnews*. You can change to the standard name later if you want. Get it to work for local messages.

(2) Post an article to newsgroup **general** with the old news announcing the change. Make available documentation such as the accompanying paper *How to Read the Network News* to the users. This article will be the last one in the old news.

(3) *Chmod* the old news directory to 555 to prevent any more news from being posted. (Actually, this will prevent the bitfile from being updated, so it may not be a good idea.)

(4) Replace the old *rnews* program with the new *rnews* program.

(5) Test it by having your neighbor send you a message.

(6) Wait a reasonable period for everyone to have read the final article with the old news. Perhaps a few weeks is right.

(7) Uninstall the old news.

Users will have to invoke *readnews* instead of *netnews* to read news. Depending on your old method of posting, this could be changed too. (If you were using mail, it does not need to be changed.) They will also have to fix their subscriptions. In general, they can type

netnews –s

to see what they subscribe to on the old system, and then create a file in their home directory called *.newsrc* containing options −n *their subscription*

The format of the subscription pattern matching is the same as in A except that **ALL** is replaced by **all** (change to lower case). Something along the lines of this could be used to automate this:

(echo −n "options −s" ; netnews −s | sed s/ALL/all/) > .newsrc

12. Conversion from 2.9 to 2.10

Conversion from 2.9 to 2.10 is not nearly as involved as an A to B conversion. The user interface does not change much, and the user *.newsrc* files are not affected. However, it is recommended that you do the conversion during a time when no news is received, so that incoming news will not get lost. One way to ensure this is to make */usr/bin/rnews* be a shell script which saves the article in /usr/spool/innews/***$$*** (***$$*** is the process id of the particular shell and will be unique for each article).

The first step to conversion is to customize the sources. In the past, you had to take a fresh distribution and edit the *defs.h* file and *Makefile* to suit local preferences. If you had many local changes, or didn't record the local changes, upgrading could be annoying. 2.10 provides a mechanism to automate these changes. Create a shell script in the src directory called *localize.sh*. (You can use *localize.sample* as a template.) This shell script should copy *defs.dist* to *defs.h*, and copy either *Makefile.v7* or *Makefile.usg* to *Makefile*. It should *chmod* any files that need to be changed (often *Makefile* and *defs.h*) to a writable mode. Then it should invoke *ed*(1) on the files, making any necessary local changes.

The next step is to compile the software, with *make*(1). It may be necessary to update the *localize.sh* file until you are satisfied with the compilation. Note that after any change to the *Makefile* in *localize.sh*, you should run *localize.sh* by hand. Otherwise, although make will run it for you, it will then continue to do the make with the old *Makefile*.

When the software is compiled, you should run the *cvt.active.sh* shell script, with the *lib* and *spool* directories as parameters. This will create a new active file in ***LIB/active***. Then run *cvt.links.sh* with the *lib* and *spool* directories as parameters. Then run *cvt.names.sh* with the *lib* and *spool* directories as parameters. Old news will be linked into the new hierarchy while leaving links in the old hierarchy. If you were using the default library and spool directories, you would do the following:

```
sh cvt.active.sh /usr/lib/news /usr/spool/news
sh cvt.links.sh /usr/lib/news /usr/spool/news
sh cvt.names.sh /usr/lib/news /usr/spool/news
```

The next step is to back up the old binaries:

```
mv /usr/bin/rnews /usr/bin/ornews
...
```

and to install 2.10 with

make install

Once it is installed, any incoming news will be placed into the new hierarchy but not the old one. The critical time window is between running the three shell files and installing the new software − any incoming news between these two points will appear in only the old hierarchy and be lost to the new software. If any significant time elapses here, you should divert *rnews* into a separate spool directory as described above.

It is crucial that you run *expire* before any new news arrives. *Expire* will update several key files automatically.

Finally, test things by posting articles to **to.*neighbor*** newsgroups and watching some incoming news, and announce the change to your users.

When you are satisfied that the conversion was successful, run the shell file *cvt.clean.sh* which will remove the old 2.9 news hierarchy.

Appendix A: Setting up a Compressed, Batched Newsfeed

First, **BATCH** must have been *#define*'d when you built the news system. To check, look in the file *defs.h* in the news source directory. **BATCH** should be defined as a program name (by default, *unbatch*). If it's undefined or commented out, define it, re-make the news system, and install the new software.

You'll also need a working *compress* program. Use the one shipped with this news distribution, which is based on version 4.0. Your news neighbors should be running a compatible version of compress. Versions 3.0 and 4.0 are compatible with each other, but both are incompatible with versions 2.0 and before.

Update your *sys* file. First, add the **F** flag to the other news system's line. For instance, if your compressed-and-batched news feed is named *frobozz*, and its *sys* file entry looks like: frobozz:net,mod,na,usa,ca,to.frobozz:: then add the **F** flag as the third (colon-separated) field: frobozz:net,mod,na,usa,ca,to.frobozz:F: Now the pathnames of articles to be sent will be stashed in a file. This file is named in the fourth field of the *sys* entry; add it now. Use an entry of the form ***BATCHDIR/system***, where ***BATCHDIR*** is usually */usr/spool/batch* (the actual value is defined in the news *Makefile*), and *system* is the name of the remote system, in this example *frobozz*. A name of that form is necessary: the *sendbatch* script, which sends the batched news, looks for a file name of this form to decide if there's news for the remote system.

Your completed *sys* file line should look something like:

frobozz:net,mod,na,usa,ca,to.frobozz:F:/usr/spool/batch/frobozz

In */usr/lib/crontab,* find or create at least two news lines: one that runs nightly, and one that runs every hour or so. The nightly-run script should run *expire*, trim log files, and perhaps compile weekly statistics that you post to a local-area newsgroup one day a week. The hourly-run script should complete the transmitting task with a line like:

```
sendbatch -c frobozz
```

Make sure the script knows how to get to the directory in which *sendbatch* lives. You can either mention the directory in the script's **PATH**-setting line, or replace *sendbatch* with its full pathname. *Sendbatch* reads the files mentioned in */usr/spool/batch/frobozz*, batches them, optionally compresses them, sends them to the remote system, and arranges for remote processing.

This remote processing is directed by another file in **BATCHDIR**. Make a file with a name of the form ***BATCHDIR/system***.cmd (for this example, */usr/spool/batch/frobozz.cmd*). Put a line in it specifying the command that the remote system should execute to unpack the news batches that your system will send. An example *frobozz.cmd* would be:

```
uux - -r -z -n -gd frobozz!rnews
```

Now your system will transmit compressed batches. The receiving side of the business is handled largely by a program called *rnews*, which will call other programs in **LIBDIR** to do additional processing on the incoming batches.

Make sure there is an executable file called *rnews* in the **BINDIR** directory (check the *Makefile* for its actual location). It must be reachable by UUCP or by whatever transport you'll use to transfer the netnews. If you defined **BINDIR** as */usr/bin*, you should have no problems because *uuxqt* can already get there. If you defined it as a different directory, you may have to teach *uuxqt* to look in that directory; accomplishing this varies from system to system. On 4.2BSD, add the directory to the **PATH=** line of your UUCP *L.cmds* file. On System V, on the *rnews* line of your *L.cmds* file, add a comma followed by the remote system's name on that line. If yours is in */usr/bin/news/rnews*, your *L.cmds* file will look like:

```
[For 4.2BSD]
PATH=/bin:/usr/bin:/usr/bin/news
rnews

[For System V]
/usr/bin/news/rnews,frobozz
```

Other systems have a similar file in the */usr/lib/uucp* directory by which you can specify added programs and paths different from the defaults. HP-UX, for example, has a */usr/lib/uucp/COMMANDS* file which expands *uuxqt*'s horizons. In more restrictive cases, paths are compiled into *uuxqt*. If you can't modify any UUCP files, just put *rnews* in */usr/bin.*

You must also have a *cunbatch* in **LIBDIR** (wherever your *Makefile* defines it), because *rnews* will eventually try to exec that copy.

Tell the person at the other end of your newsfeed to use *sendbatch –c* to send you news. Once that's in place, watch your UUCP *LOGFILE* and your news *log* and *errlog* files to ensure that news is being correctly received and unpacked on your system.

Older compressed batching systems will try to exec *cunbatch* instead of *rnews*. If you are still communicating with these, leave *cunbatch* in **BINDIR** until they have upgraded their software.

Appendix B: MULTICAST

If this is defined (in *defs.h*) then two new flag characters become defined in the *sys* file. The first, and most important, of these is the **M** flag.

If the **M** flag is set on some line in the *sys* file, then the fourth field (transfer command) is redefined to become a *multicast* name. That is simply another system name, expected to be found in the first field of some line in the *sys* file (textually following the line containing the **M** flag).

When a news item is being retransmitted, if it should (according to the subscription list) be sent to a system that has the **M** flag set, then instead of a command being run immediately to transmit the news, the news system remembers the system name, along with the multicast name (fourth field).

Eventually the multicast system name is found in first field of a sys file line. If its subscription list allows transmission of this news item, then its command will be executed. This command may have up to two ‘‘%s’’ substitutions in it. The second of those is replaced by the name of a file containing the news item (used with the **U** flag). The first is subjected to rather special treatment. The whole ‘‘word’’ (delimited by white space) containing that ‘‘%s’’ is duplicated as many times as there were systems with the **M** flag set that referenced this multicast name (which might be 0 times, causing that ‘‘word’’ to be omitted). In each of these duplicates, the ‘‘%s’’ is replaced by the name of a system. Note the multicast system name itself is not included in this process. Then the command is executed as usual.

The second flag available if the news system is built with **MULTICAST** defined is **O**. If this flag is set, then the sys file line will be ignored unless the system name is a multicast name from some earlier line with the **M** flag, and the news item is to be sent to that (earlier) system. This allows the subscription list for the multicast system name (which is likely to be a fake system name, invented just for this purpose) to be given a very wide subscription list (like **all**) without any unusual effects.

Here is an example. Assume that you wish to forward **net.unix** to four people by mail. You could do this as ...

```
fred:net.unix::mail fred
harry:net.unix::mail harry
jane:net.unix::mail jane
tony:net.unix::mail tony
```

however this causes the mail program to be started 4 times, once for each recipient. On some systems starting the mail program is a very expensive operation. If **MULTICAST** is defined, an alternative method is

```
fred:net.unix:M:tony
harry:net.unix:M:tony
jane:net.unix:M:tony
tony:net.unix::mail tony %s
```

This would cause just one command to be run: ‘‘mail tony fred harry jane’’. Note that ‘‘tony’’ must still be explicitly included in the argument list to the mail command; the ‘‘%s’’ does not expand to include the multicast ‘‘system name’’ itself.

A more useful way of doing this, which does not assume that all the mail readers will want to read the same newsgroups is as follows.

```
fred:net.unix:M:Mail
harry:net.physics,net.astro:M:Mail
jane:net.unix-wizards,net.women:M:Mail
tony:net.unix,net.unix-wizards,net.jokes:M:Mail
Mail:all:O:mail %s
```

Now, if a news item in group **net.unix** was received, the command

mail fred tony

would be executed. If the news were in both **net.unix** and **net.unix-wizards** then the command would be

mail fred jane tony

If a newsitem in **net.med** (which no-one gets by mail) arrives, then the ‘‘Mail’’ line will be ignored, because of the **O** flag. ‘‘Mail’’ is a fake system invented just so its ‘‘transfer command’’ can be used to send news to the other recipients.

The same kind of technique can be used for normal transfer of news to other systems if your transport network supports a facility to send to many other systems in one command. (That is, if it has a multicast facility.) SunIII (the network used in Australia) has this ability, so a typical Australian *sys* file looks like

```
emuvax:aus,net,mod,fa:M:FakeName
kremlin:aus,net,mod:M:FakeName
kanga:aus,net,!net.all,net.unix:M:FakeName
FakeName:all:OUS:/bin/sendfile -NRSareporter -d%s -x%s
```

A news item in **aus.general** causes the following command

```
/bin/sendfile -NRSareporter -demuvax -dkremlin -dkanga -x/usr/spool/...
```

to be executed. Just one command is run to send the news to three remote systems.

If a multicast system has the **F** flag set, then the name of a file containing the news is appended to the file whose name is in the fourth field, as usual. But on the same line, separated by spaces, will be appended the names of all the systems that referenced this multicast system.

For example, if the Australian site wanted to batch news, instead of sending it directly, it would simply change the last line of its *sys* file to

```
FakeName:all:F:/usr/spool/batched/allsites
```

Then a news item in **net.jobs** would cause the following line to be appended to */usr/spool/batched/allsites*

```
/usr/spool/news/net/jobs/5542 emuvax kremlin
```

This can then be processed later, in something like the normal manner. (Unfortunately no commands to do this processing are yet available).

Caution: when **MULTICAST** is defined, the first ‘‘%s’’ in all transfer commands is used for multicast, regardless of whether or not the system name is ever used as the last field of some line with the **M** flag set. To use the **U** flag in such a case, a dummy ‘‘%s’’ should be used, it will simply be omitted from the command that is executed.

As an example, if a *sys* file line were

```
foovax:net,na,usa:U:uux - foovax!foonews <%s
```

without **MULTICAST**, it would need to be changed to

```
foovax:net,na,usa:U:uux - foovax!foonews %s <%s
```

if **MULTICAST** were defined.

Additional caution: The numbers of system names that may be used in this way are quite severly restricted. Typically there may only be about 10 multicast system names, and each of those is restricted to sending to no more than about 20 systems. These limits are dynamic (that is, the numbers counted are the number of multicast systems receiving any single news item, and the number of systems that each of those will actually cause this particular news item to be sent to). These limits should easily suffice for real news sending to remote systems; however they are not likely to suffice if you want to mail news to everyone on your host.