The Hacker's Guide to News 3.0 (beta level 7)

Eric S. Raymond
Thyrsus Enterprises
Malvern, PA 19355

1. Introduction

This document discusses the data structures, organization and internals used by the News 3.0 (beta level 7) version of the USENET software. It is intended for the delectation and enlightenment of netnews hackers. It does not describe the USENET communications protocols or message standards; for those details, see RFC 1036. It also passes lightly over the details of external representation of the news database files; see *news*(5) for such details.

The News 3.0 (beta level 7) distribution is a complete rewrite of the netnews software. The original aim of the modifications was simply to add volatile groups to the 4.3BSD beta release of 2.10.3, but the project expanded over a period of a year to encompass a complete overhaul, re-organization and re-documentation of the code; diffing it against 2.10.3 or 2.11 will get you nothing but dazed and confused. Comments, questions, suggestions for improvements and the like are welcome; mail to

{att,uunet,rutgers!vu-vlsi}!snark!eric = eric@snark.UUCP (Eric S. Raymond)

or write me via USnail at 22 South Warren Avenue, Malvern PA 19355. Specifics on News 3.0 (beta level 7)'s debts to earlier news software follows:

2.10.3B — the base version from which News 3.0 (beta level 7) was modified, though very little of either its substance or organization has been left intact.

2.11B — Ken Almquist's experimental rewrite contributed the 'l' (index) and 'ud' (unsubscribe-from-discussion) reader commands, a more advanced postnews and some important ideas about the inews/rnews/postnews layering.

2.11 — All the 2.11 features -- backbone and Internet gatewaying, working ihave/sendme protocol, deferred news processing etc.

rn — Larry Wall's package contributed the original of Configure and inspired the global article marking facility, rn-style regular-expression commands, % escapes and keyboard macro facility.

C news — The new history file format was inspired by Geoff Collyer and Henry Spencer's ''C news'' design. The link-definition features of Configure were suggested by their associate Christine Robertson, who also wrote the savesrc utilities included in the misc directory of this distribution.

sendbatch — Chris Lewis's sendbatch program for 4.3BSD supplied the original of the Q option test.

uucast — Shane McCarron's uucast package provided the original of the UUCP multicasting code in libuucp.a

nntp — The nntp code was mutated from Stan Barber's nntp support for 2.11B.

Additional thanks are due to Henry Spencer for his regexp(3) routines and to James Gosling for the ancestors of the edbm.c routines. Please reserve all bug flames for me as these gentlemen are not responsible for any errors I might have introduced in adapting their code, and (in the latter case) might have trouble even recognizing it through the mutations.

2. Overview of the netnews programs and files

This section is intended to help you understand why news works the way it does. If you are familiar with a previous version you can safely skip it.

Any messaging system requires at least three things: a communications link and protocol through which messages may be exchanged, a posting method and a receiving method. The receiving method has to be capable of buffering incoming messages until users can read and dispose of their messages.

In netnews, the first two roles are fulfilled by the UUCP protocol and the news posting programs. The news receiving method buffers incoming articles into a common spooling area. The reader programs provide various interfaces through which to browse through the spooled articles.

An objective of netnews is to support subject-based bulletin boarding. To this end, the article spooling area must be (at least conceptually) divided up into named topic areas or newsgroups. A single article may be posted to multiple newsgroups.

Netnews is a broadcast-based system; that is, rather than transferring articles point-to-point between a sender and a designated receiver, articles are simply broadcast and it is up to each receiving site to accept or reject them. In particular, receivers have to be able to recognize and reject articles they've already seen, and each user will typically only want to see an article once (no matter how many newsgroups it appears in).

Practical considerations (the finite size of disks!) require that the news software be able to clean out the tree periodically. This requires a means of detecting and deleting articles that are no longer useful enough to keep space tied up for.

All these facts together determine the organization of the netnews data files and programs. The article spooling area is organized as a file tree in which the directory nodes designate topic areas. The files used to identify articles already seen and expire old articles live in a private library area, and each user keeps a file containing information on articles already seen. All of these files are collectively what is referred to as ‘‘the netnews database’’ below, where more detail on the representations used for them is given.

Programs for a system like this fall into three functional categories: readers, posters and prowlers. Readers are interface programs for browsing through the article tree; they read everything in the database but write information only to the user's list of things seen. Posters modify the article tree to add (and occasionally delete) articles, and prowlers are demons that pop up occasionally to delete articles and possibly perform other housekeeping (at present, all prowler functions are modes of the expire utility). Contention between readers is seldom a problem and can inconvenience at most one user, but contention between concurrently-running posters and prowlers could leave the database in a bad state for everybody and has to be guarded against.

The news tools automatically try to use the most fine-grained interlocking available on your system. The critical data structures are the history and active files. There are three major cases:

1. No mandatory file locking (V7, SIII) -- the system tries to synthesize semaphores out of link() or similar primitives. Locking is by file only; only one instance of rnews or expire may be running at any given time. If rnews tries to run while expire is going, the news will be spooled.

2. Mandatory locking by file only (4.xBSD, SVr1, early SVr2) -- as above, but with better security and robustness in the face of signals, system crashes, etc.

3. Mandatory region locking in files (XENIX 3.0, later SVr2, SVr3, POSIX) -- Multiple copies of rnews and expire may run concurrently; accesses to the database are automatically serialized.

3. General comments on reading the code

More details on all the new and most of the old modules can be found in new comment headers styled after UNIX manual pages which have been added to them. Each contains a complete functional description of their interfaces; each can be compiled with an interactive tester with which its function can be verified. If you don't understand what a given sequence of calls should do to the news database, compile the appropriate tester and try it. Hackers are urgently requested to adapt one of these testers for any new data type they create.

4. Interface modules for the netnews database

The *news*(5) manual entry describes the formats of the netnews database files and the modules News 3.0 (beta level 7) uses to manipulate them. This section discusses the functional interfaces and data structures to these representations. More detail on each module (including a list and description of entry points) can be found in the documentation header attached to it.

Readers and posters need to be able to assign an article a unique ID (and corresponding file) within each newsgroup (subdirectory) it belongs in (by convention, this ID is the sequence number of the article in the group). Both kinds of program also need to be able to get a list of all groups that are currently active. In order to avoid lots of exhaustive searches of the article tree (and to hold data associated with each group such as whether or not it can be posted to and its expiration class) the news software relies on a file named LIB/active.

The News 3.0 (beta level 7) code provides a pair of modules called rdactive.c and wractive.c to handle interaction with this file. At startup, rdactive.c digests the active file into an array of allocated in-core structures. This makes for very fast and flexible access to the data with a minimum of disk hits, but the code has to assert a lock on the active file and go to disk to read and update it when it allocates a new article (so that multiple posters won't grab the same article number).

The only place this causes any real peculiarity is in the wractive.c code that handles group creation and deletion messages; there, we have to assert a lock on the file before re-reading it (to make sure no one else is using it) then make the change before unlocking. Expire uses a similar strategy. Note that readers never have to lock the active file, and posters never have to lock the active file except when reserving a new article number.

In rdactive.c's view of the universe there is always a current or selected group on which all field accesses and updates are done. The code provides handles for selecting groups by name or stepping through them in sequence. All the fields mentioned in the *news*(5) manual page can be got at through functional or macro handles.

4.1. The history file

The rdhistory.c/wrhistory.c module pair supplies functions to access and modify the article history file (format details are available in *news*(5)). The information is actually kept in a hashed-key database maintained by the routines in edbm.c.

4.2. The .newsrc file

Each user's .newsrc file holds state information about what the user has seen and selection options about what kind of news the user wishes to see in the future.

The rdbits.c, rdnewsrc.c and wrnewsrc.c modules support a complete set of entry points for manipulating .newsrc data. When a .newsrc file is read, the group line data is packed into some dedicated fields and an allocated area of packed bits in the active-groups array, which must have been set up by a previous active file read (the ID lines simply go into an allocated list). Thereafter, functional handles can be used to access and modify the .newsrc data of the currently selected group. In particular, functions to mark a given active message read or unread are provided.

The group data structures contain two fields that don't directly correspond to the format above. One is an index number; the other is a count of articles unread in the current group. The index number tells you where in the list of .newsrc groups this group was found; it's used by an entry point that sorts the in-core image of the active file to put the groups in the order they were found in the .newsrc, and has a special value NOTFOUND that tags groups not found there at all. The unread-article count is kept up to date by the article-marking functions and is useful for checking whether the group contains any new articles.

4.3. The feeds file

This file describes the links your system has with its USENET neighbors. Functions to read and interpret it are provided in feeds.c, part of the *libnews.a* news library shared by all news programs. Note that the function that searches for option letters in the third field recognizes numeric suffix options, and also supports a quoting syntax that permits you to follow an argument with a string of text options. It is strongly recommended that all control parameters for new transmission features be settable in this way.

4.4. The article spool directory

For each newsgroup (say **comp.misc**) there will be a subdirectory LIB/comp/misc (where LIB is the directory named by the TEXTDIR configuration attribute, typically /usr/spool/news/) containing articles, whose file names are sequential numbers, e.g. LIB/comp/misc/1, etc.

Each article file is in a mail-message-like format described in *news*(5). 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 Internet standard for mail documented in RFC 822.

4.5. Other LIB files

The news library includes a number of data files used by the code:

LOCK -- file used to lock active file for exclusive use
EXPLOCK -- file exists only when expire is running
aliases -- association list, maps group aliases to names
distributions -- information file, describes USENET distributions
errlog -- log file, significant errors only
failures -- articles that couldn't be localized (batch fmt)
feedback -- accumulated rating sweeps information
feedbits -- subscription bit file (see FEEDBITS/CACHEBITS)
followups -- association list, maps groups to followup groups
log -- log file, normal transactions included
mailpaths -- association list, maps group names to moderators
newsgroups -- information file, describes available newsgroups
readnews.help -- help file, describes readnews commands
vnews.help -- help file, describes vnews commands
xmitlog -- transmission log file (if D options are enabled)

The news library provides functions for reading, using and writing association list files, and for writing to the log files. The information and help files exist to be displayed by user-interface programs.

More details on some of these follow:

4.5.1. LOCK

The LOCK file is created and checked by the Version 7 and System III implementation of the lock() and unlock() functions.

4.5.2. mailpaths

This file is used to route submissions to moderated groups to moderators. Each line maps a newsgroup name to a mail address for the moderator.

4.5.3. attributes

This file defines runtime-configurable attributes for the news system. The RUNTIME configuration symbol must be defined for it to be evaluated. The most important attributes for a beginning configuration are 'gateways'.

The value of a gateway key is expected to be a sprintf(3) format string with %s in the user slot of a mail address, and to specify a route to the the destination site part of the address acceptable to your mailer.

There are, at present, only two recognized kinds of gateway keys. One is 'backbone', which defines a site with a full set of mail aliases mapping group names to group moderators. The other is 'smart-host', which defines a path to a site that has an autorouting mailer that will accept Internet-style @-addresses (note that this doesn't necessarily mean an ARPANET or Internet site; any site running the UUCP project's smail will do).

5. The structure of news readers

5.1. Overview

In News 3.0 (beta level 7), readers are assembled out of three kinds of service libraries that provide 1) article tree traversal logic, 2) a screen/keyboard I/O manager and 3) a command interpreter. Communication between these three parts takes place through uniform, well-documented interfaces (so that, for example, any conforming command interpreter can be linked to either a screen-oriented termcap-based I/O manager or a paging I/O manager using only cooked mode).

Note that readers may call also up a symbiotic 'newsfilter' process to do news interest scoring. Interfacing to news filters is described in the *newsreaders*(5) and *newsfilters*(5) manual pages. Code to handle the reader end of such links is automatically included in the reader service libraries, and code that handles IPC from the filter end in also included in the libfilt.a library (see below).

In this section, we describe the interfaces of the reader-support libraries.

5.2. Article-tree traversal logic

All news readers have the basic mission of walking through the article tree, using active file and .newsrc information, and presenting each new article through a given interface. Older versions of the netnews system used a lot of very complex, ad-hoc code scattered through several modules to do this. In News 3.0 (beta level 7) this logic is isolated into six modules that layer on one another. Four of these (the {rd,wr}active.c and {rd,wr}history.c modules) have already been discussed. The highest two levels are described here.

nextmsg.c − this layer supplies functions to aid in traversing the tree of articles. The major entry point is nextmsg(), which (normally) goes to the next new article, setting up information on the article file name, its directory, and its sequence number within the group. Options to traverse backwards and/or to see old articles are available, and entry points to seek to specified or remembered locations in the tree are provided.

session.c and **reader.c** − this layer provides the major ''get next article'' function for interactive readers. It maintains a trail of articles seen in this session, including their location and header information. The trail is all allocated storage, so there's no limit on it short of physical or virtual memory size. The trail is used to support the subject-thread following code, which also lives here. Backtracking into the trail and unwinding of backtracks is also supported. A function to do marking or unmarking of all of a article's cross-references is also provided.

These layers supply a complete set of calls for treating the article tree and history as abstract data types, separating these functions from the user interface code of news readers. Note that they provide primitives for navigating through the article tree in either article-id or subject-thread order.

5.3. The role of I/O managers

The generic command interpreter functions and service libraries do all their display I/O through the following functions:

```
vinit()       -- initialize the interface manager
vgetc()       -- get a char from the command queue
vungetc()     -- push a char back to the command queue
vgetline()    -- get a line from the command queue
venqueue()    -- append a character to the command queue
vclearin()    -- clear the input queue
vttymode()    -- go to tty mode
vcrtmode()    -- go to screen mode
vdelay()      -- pause (if in screen mode)
vupdate()     -- update the display
vgetcmd()     -- parse a command out of the input stream
```

We term an implementation of these calls an *I/O manager.* The modules ttyin.c and ttyout.c contain an interface manager for line oriented tty-style interfaces. For a more interesting example, see visual.c, which defines screen-oriented interface functions for vnews-like readers. Or see paged.c, which implements the same calls as

visual.c in a way that uses only cooked mode.

Interactive readers will also want to define the following:

```
vexpand()     -- expand reader-specific %-escapes
vcommand()    -- the command interpreter
```

By default, actual parsing of the returned command string is done by gparse() in gcmd.c. This function returns a standard command structure usable by interpreters.

The vexpand() and (of course) vcommand functions are specific to each command interpreter. They have standard names and interfaces so they can be called by service layers.

5.4. Special I/O-manager considerations for screen-oriented news readers

Even in the presence of a support package like termcap(3), screen-oriented display of text poses some tricky special problems. For starters, command processing is complicated by the requirement to handle and buffer single character keystrokes in raw or cbreak mode, especially if interrupts are to be handled gracefully. Long lines must be folded to the width of the terminal. The conveniences of standard I/O are lost.

The tasks described above are at least independent of the display format and command set chosen. In TMN-Netnews they are handled by the following modules:

ttyctl.c − UNIX-version-independent tty mode control. The interface is modelled on System V Release 3 curses(3).

scrnctl.c − a screen manager package, provides capabilities equivalent to a large chunk of BSD curses(3) (or a smaller piece of the AT&T version of same), given termcap(3) to work with.

vtermio.c Handles the subtleties of queued raw-mode I/O with interrupts, including the ability to hook functions to each character input and alarm signal event.

visual.c − interface manager primitives. This implementation assumes that two lines of the tty screen are reserved for messages and prompts, and that the remaining lines are available for text.

Building a custom visual news reader is easy. All you should need to change to customize your own command structure, screen layout and special actions is the equivalents of the functions presently in vnews.c (which you should probably use as a starting point).

If you are building or debugging a visual news reader and DEBUG is enabled, note that readinit.c checks at initialization time to see if the usetty[] array is nonempty, and, if so, interprets it as the name of a tty to redirect screen I/O to/from. If you don’t give a full path name it will try to fill in the missing path by prepending "/dev/" and (if usetty[] starts with a digit) "tty". The intent is to make it easy to run a vnews session under a debugger like sdb from one tty or window and do all the normal user-mode I/O through a second. The -T option of vnews invokes this feature.

Note that you should have set TERM to reflect the termcap/terminfo display type of the terminal to be used for the run; i.e., if you’re going to redirect vnews I/O under sdb to a Hewlett-Packard 2621 called tty000, you should see

```
TERM=hp2621 sdb vnews
[sdb messages]
>r -T 000
Attaching to /dev/tty000...
```

on your screen, following which the reader display will appear on tty000.

5.5. Standard interpreter parts

The database-access and article-traversal layers described above provide all the essential services for a news-reader program in a way that completely seals it off from underlying file organizations and formats. The I/O manager architecture does the same for keyboard and display use (compare, for example, vnews with edvnews -- same command interpreter, two different I/O managers).

The database service libraries and the three I/O managers in the distribution provide sufficient generality that the task of building a new reader reduces to building a new command interpreter and linking it with pre-existing

parts. The distribution also provides some useful standard parts for building the interpreters themselves.

The gcmd.c and rfuncs.c modules implement reader commands common to readnews and vnews. The former dispatches from a standard command structure such as implementations of vgetcmd() return; the latter executes the commands. Either or both may be used to provide most of the standard functions for a custom reader.

Note further that although the command names of the generic command interpreter mostly coincide with the actual corresponding commands you type to readnews and vnews, this doesn't have to be so. Interfaces can be built with a layer of indirection between user commands and the generic-command interpreter. In fact both readnews and vnews each do this in order to be able to implement a few special commands not recognized by the other.

At a deeper level, the generic command interpreter itself could be thrown away or rebuilt without losing most of the command service features in rfuncs.c or the article tree traversal layers underneath them. An rn-like multilevel command interface could be implemented this way without too much trouble.

The browse.c module provides general, I/O-manager-independent logic for page-oriented browsing of multiple input streams including the current article, the current group's subject list file, help files, and a user-formatted group data list. Sufficient user-definable hooks and escapes are provided that any kind of interpreter modification of normal browse order can be implemented easily.

5.6. The user interfaces of readers

The cumulative effect of having all these layers available should be that would-be custom reader designers can kit-build 90% of their project out of the existing service libraries, the remaining 10% being the custom command interpreter module itself.

For example, the readnews, ednews, mailnews, vnews and edvnews readers all share readinit.c which holds common code for initialization and option processing. Each reader runtime contains a main() which is the interface manager or (if non-interactive) action loop for that interface. The interface manager uses session.c and nextmsg.c to fetch articles and hands most of the command interpretation off to gcmd.c, which calls service modules such as rfuncs.c. All use one of the three I/O managers supplied.

The checknews and mailnews tools differ mainly in that, being non-interactive, they don't use or link readinit.c, gcmd.c, rfuncs.c, or any of the associated command-interpreter parts.

5.7. Front-end/server organizations

Another method of rolling your own interface is to use the ''ednews'' server. Ednews is a generic reader-interface server suitable for controlling through a pipe or named-fifo pair (the basic trick is that it always hands back file names rather than file text, because the interface front end can read files). The distribution source includes a module ednewsipc.o that provides high-level control facilities for ednews from C front ends; this module can be compiled to yield an interactive tester.

Anyone on a machine with a windowing system could use these to write a front-end that uses the local facilities to achieve Smalltalk-like sexiness. SUNTOOLS, X, Gosling's NEWS or the layer windowing on the AT&T 7300/3B1 would do fine. Ednews is also useful as a back end to EMACS; in fact, the misc directory includes a package ednews.el that provides primitives to help GNU Emacs front ends talk to ednews(1).

Finally, the distribution also includes ''edvnews'', which is like ednews except that it interprets the full vnews command set and returns cooked-mode pages of article text rather than article filenames. This could be used like ednews with an rudimentary front end.

5.8. How to build news filter processes

The distribution includes one sample newsfilter, rnkill. If you study the rnkill.c source, you will see that it defines *one* kill-command interpretation function; the details of IPC with the reader, parsing of stored kill commands and the like are all handled by modules in the libfilt.a library.

To implement a new filter, all you have to do is write your own analogue of rnkill. See the *newsreaders*(5) and *newsfilters*(5) manual pages and the description of the filter librry modules below for more details.

6. The posting programs

The top levels of the posting programs are structurally fairly simple. Postnews and inews are both front ends that call rnews through the newpost() routine in libpost.a. The motivation for this organization is to make rnews (which is executed by every system that calls in to send news and must be suid) as small and simple as possible.

The inews code consists mainly of options processing and a dispatch to one of three execution modes. It knows how to digest standard mail headers and, if necessary, combine their sender information with sender information in a news-format header. It will also automatically mung notesfiles into netnews format if that conversion has been enabled via the ZAPNOTES config switch.

Postnews is more elaborate; it does various sanity and correctness checks (besides calling an editor for article composition). For the interesting case (article origination) the important code is in originate(). Submission date fields are set here, the NAME and ORGANIZATION environment variables are interpreted here and forged names in From fields are handled.

Note that rnews's calls to hread() will bomb unless it sees a Path and From header and a submission date header such as originate() generates; this is deliberate. The rnews code does **not** necessarily need to see a Message-ID; if none is present it assumes the article is being originated and supplies one.

The rnews executable is linked together from versions of the database access code ({rd,wr}active.c, {rd,wr}history.c) and the following special modules:

rnews.c − options processing and control flow for rnews.

insert.c − these functions do local posting of submitted articles.

post.c − article-posting dispatch functions for rnews. The code for dispatching article transmission to other systems also lives here.

unbatch.c − this module knows how to detect and crack the various kinds of batching used for news transmissions.

control.c − handles the interpretation of incoming control messages by rnews.

dispatch.c,transmit.c − lower level transmission code, also used by sendbatch (see below). These may use the library modules sysmail.c and uucast.c.

The decision to digest the active file into core before use practically demanded that rnews be enhanced to handle batched articles itself, in order to avoid reading the database more often than necessary. Serendipitously this turned out to speed the process and eliminate a potential security hole.

Given all this, it seemed natural to eliminate all flavors of the old unbatching programs (unbatch, cunbatch, c7unbatch) by giving the rnews program the ability to do decompression itself using either a default or specified decompression program. This was pretty simple, mostly a matter of plugging a popen() in the right spot (eventually, for speed and security, I wrote a replacement peopen() that does its fork-exec directly instead of using a shell).

The remaining posting program is sendbatch, which replaces a collection of scripts in older versions. This program, when invoked, runs through the batch files of all or a selected subset of systems, batching and transmitting all pending messages to them. The motivation for turning this into a program was to allow it to interpret the A, C, E and N transmission options; it shares its low-level transmission code with rnews. Note that batch files list message IDs rather than (as formerly) file names; this is because the volatile groups feature made it possible that the article copy listed in a batch file might be deleted while another copy was still on the system. More generally, this gets the abstractions right; articles are not identified with a particular one of their evanescent instances, but rather by their true names, the Internet IDs.

7. News access via NNTP and other networking methods

There are two configuration switches called SHARED and NONLOCAL intended to permit netnews to be used with a news hierarchy located on a shareable file system and through NFS or RFS. To understand what these switches do, it is useful to understand that all of netnews's access to article text and other news database goes through a limited set of 'contact points', access primitives that normally fetch things off a local disk volume or write to it.

One of these contact points is rdactive(), the function that digests the current state of the active file into core.

Four more of these contact points are the history read access primitives. These initialize access to the history file, permit one to seek given article-IDs in it, permit one to get newsgroup/article locations for an article given its ID, and releases access to the history file.

Another two are artname(), which returns the name of an article text file corresponding to a newsgroup/number pair, and lstname(), which returns a subject-line list for a given group.

These seven contact points are normally supplied by getfiles.c, rdactive.c, and rdhistory.c You can replace them with primitives that snarf the requested stuff over a network by writing a network service library that implements them, say, via requests to an NNTP server. The libnntp.a library in src/D.network does exactly that.

Of course, we need to be able to post articles, doing some equivalent of localizing the article to newsgroup spool directories on disk. Our eighth contact point does that to a suitably prepared article file.

NONLOCAL is the switch to set if you want your readers to run as clients of a network news server; in July 1988 as this is written, NNTP is the only such animal. This library turns all eight contact point calls into NNTP requests, snarfing data to local tempfiles and returning the tempfile names when needed.

To use netnews with NFS, simply declare ADMDIR to be a shared filesystem. Each site can have its own LIBDIR holding executables that operate on the ADMDIR files.

Beware! The POSIX lockf(3) primitive does *not* work properly over many NFS implementations (this is, in particular, a known problem on Suns and Pyramids). If you use SHARED, make sure the LOCKF switch is on.

If you have RFS or some similar transparent stateful file-sharing, you'll be able to write to the shared volume without risking wedgedness due to a crash while the write lock is on (this is why NFS clients must post through NNTP).

Note that if you choose to go the NONLOCAL route, you'll never run expire or rnews (they take place on the server machine). The makefiles know about this and will only make readers. The InstallNews script also knows about this from your configuration files and won't create crontab entries for expire, newsclean or sendbatch as it normally would.

Also note that an NNTP client talking to NNTP 1.5 patchlevel 3 or earlier will lose a few peripheral features such as the ability to get an active newsgroups list from postnews.

8. How to hack the news transport layer

There are basically five ways you can change the transport layer of netnews.

8.1. Add another support module to bbsauto

This is appropriate if the news source you want to gate to and from is a BBS service like CompuServe, GEnie, BIX, XBBS or any micro-based BBS.

The BBS-dependent parts of bbsauto are carefully isolated from the driver code. Each service robot is implemented as a small set of capability functions implemented in terms of high-level constructs like expect/send, capture, and upload functions.

A competent C programmer should be able to bring up support for a new service in eight to ten hours; various debugging hooks have been included to ease this task. It is recommended that you copy one of the existing modules and modify it, rather than building one from scratch.

A robot prototype, with all kinds of useful info included in it in comments, is included as the file robot-proto.c. Examine it to learn more, and have fun.

8.2. Press your mailer into service.

This requires that you be able to mail to the input of a program on the target machine. This is the cheapest and easiest method for sites running the Berkeley 4.x mailer (or equivalent) over Internet or Berknet. See the *Installation Guide* for details.

8.3. Compile in a different DFLTXMIT/UXMIT command.

If your OS permits you to spawn remote jobs and feed them transmitted data on standard input a la **uux**, and all your downstream sites are alike, this is a good way to go. Remember that at the point of transmission command generation news will plug three %s parameters into DFLTXMIT/UXMIT: the remote system name, the name of news's reception agent and the name of the transmission file (in that order).

8.4. Use Ken Almquist's xfernews utilities (or equivalent).

This method makes news work over a data-only link with no remote execution capability (i.e. Kermit, XMODEM and the like). It requires no changes to netnews but involves fairly high (uucp-like) overhead; xfernews works through a whole set of enqueuers and server demons designed to be called from the command field of your feeds file. For more details, snarf the code from your nearest comp.sources.unix archive.

8.5. Handcraft your own additions to transmit.c

This may be the right thing to do if you have Sun NFS or AT&T RFS or if session-level ISO C bindings ever get really standardized and added to C libraries (to dream the impossible dream...:-)). The s_option() function has been defined in order to make feed options parsing as painless as possible. If you do something like this and it works, please wrap it in #ifdef/#endif switches and post diffs to the net so we can all use it.

Every effort has been made to layer the posting routines to make them easily understandable and modifiable, but things do get a bit tangled in the lowest levels (dispatch.c and transmit.c). The rest of this section attempts to dispell the fog so that new transport layers can be written to correctly use the existing protocol options. As described above, the netnews transmission code is broken into three separate modules. The idea of the three-part organization was to isolate the point-to-point code from the broadcast logic. The division of labor is intended to be:

post.c -- broadcasting logic
dispatch.c -- point-to-point/multicast with batch spooling
transmit.c -- plain point-to-point or multicast

The idea here was that only sendbatch(8) would call transmit() directly; everybody else (in particular, the broadcast() and control() code in rnews) would go through dispatch(), ensuring correct interpretation of the batching and broadcast options.

According to this theory, post.c should know nothing about transmission options, dispatch.c should handle the L, N and F options (which control locality, batching and ihave/sendme) and transmit.c should handle the rest (A, B, C, D, E, M, S, U, and X). Unfortunately, reality turns out to be not quite that simple.

The F option posed the major problem. In previous news versions using uux as a transmission layer, netnews called one of four remote agents (rnews, unbatch, cunbatch or c7unbatch) to do its decoding. To know which one to pick, transmit() has to be aware of the state of the F option for the target system.

The third problem is the N option, which has to be interpreted in different ways depending on whether rnews or sendbatch is doing the transmitting (with both F and N we want *ihave*s to be generated in the outgoing batch, not at the time dispatch() is called). The N logic in dispatch() is therefore enabled only when F is off, and sendbatch(8) has its own N code.

The functions *artfilter*() and *filefilter*() are called by dispatch() to apply translations on a per-article and per-transmission basis respectively. Implementations of new article protocol (as opposed to broadcast logic) options should fit in those functions in order to be uniformly applied to single sends and batches by all transport layers. See the handling of the 'E' option (within the ENCODE #ifdef/#endif in transmit.c) and the associated sevenbit.c module for a model of how to add this sort of thing.

General-case multicasting (the X option) is supported by the simple expedient of handing the list of target systems down to transmit() level along with the name of the generated batch. Any user-supplied transmission command attached to an X feed is expected to know what to do with such a list.

If there is no transmission command given for an X feed and the UUCAST configuration attribute is defined, transmit.c will call on UUCP multicast code in uucast.c (in the libuucp.a library). This feature is experimental and tricky; read the uucast.c code and use at your own risk.

Note that the H option of older versions is no longer supported. The existence of sendbatch makes it unnecessary, and the changes in history file format make it unsafe.

9. The netnews support libraries

The service layers described above have been gathered with miscellaneous other functions into eight library archives available for the implementation of new news tools. These are:

libport.a general portability functions.

libnews.a various small functions specific to news.

libread.a support modules for readers.

libscrn.a support for screen-oriented interfaces.

libpost.a helper functions for new postings.

libpriv.a functions for news database modification (for ‘privileged’ programs).

libuucp.a routines that diddle with UUCP job files.

libeipc.a routines for communication with ednews(1).

The remainder of this section describes the contents of each library.

Note that each of the libraries depends on the previous one. They correspond roughly to the interface layers decribed above. They must be linked in reverse order (i.e. libscrn.a first, libport.a last).

9.1. The portability library

The portability library is a collection of non-POSIX/X3J11 general utility functions used by the news code. Many of these emulate library or even system calls from System V or 4.2BSD UNIX that aren’t otherwise available on older versions. This library is kept separate from libnews.a in hopes that it can be used with net source distributions. The header file portlib.h contains some common definitions for this library.

alist.c -- This code handles association lists. An association list is represented by a file of lines each containing two fields separated by whitespace. Entry points exist to read in a file of associations and to do lookup on the first field and return the value of the second. It uses the dballoc.c routines.

arpadate.c -- The arpadate() function resembles UNIX ctime(3) except that it generates an ARPANET-format date suitable for use in RFC822 message headers.

backquote.c -- This function reads the input from a popen(3)ed command into a buffer, up to a given maximum number of characters.

bcopy.c -- Implements the BSD 4.2 bcopy() function on systems that don’t have it.

bzero.c -- Implements the BSD 4.2 bzero() function on systems that don’t have it.

bitbucket.c -- Returns TRUE or FALSE according as a given fp is/is not connected to /dev/null.

checksum.c -- Code for calculating a 32-bit CRC checksum of a file section.

dballoc.c -- These routines assist you in maintaining an allocated array of records of arbitrary length in core. The functions all require a pointer to a control structure that includes pointers to I/O functions for the record type, allocation chunk sizes for the record array and the record size. Entry points exist to allocate new records, find old ones by name, do I/O between the record array and corresponding file representations and the like.

df.c -- This routine returns a count of blocks free on the file system containing its directory argument. It has some unavoidable limits and bugs; see the header comment in its source for details.

edbm.c -- These routines provide database management for news history. The code is actually a rather general hashed-key-database package. If the POSIX lockf() or XENIX locking() calls for mandatory file locking are supported access will be automatically serialized, ensuring correct operation even if multiple writers are running concurrently.

environ.c -- defines functions setenv() and delenv() to set and delete environment entries. See getenv(3) and environ(5) in your UNIX manuals for background.

errmsg.c -- A function useful for reporting UNIX system errors.

fcopy.c -- A function for doing fast copy of a named file to another named file.

filestat.c -- Primitives for querying file attributes (size, last-modified time, etc). On UNIX these are implemented by using stat(2); we isolate them here to make hypothetical future ports to OS/x, AmigaDos, or whatever easier.

fullname.c -- This function, given a user's login name, recovers whatever information on the user's full name is available.

fwait.c -- Functions used for executing other functions in background.

lcase.c -- A function used to force strings to all lower case.

linecount.c -- given an open file descriptor, this counts lines to end of file. The file descriptor is unchanged.

lockfile.c -- portable routines for mandatory locking of an entire file. They use POSIX or 4.2BSD-style mandatory file locking if they can and a busy-wait loop on lockfile linking if they must.

mkdir.c -- implements emulations of the mkdir(2) and rmdir(2) system calls for systems that don't have them. The former can't be made to work very well on V7, see the documentation header.

more.c -- this function gives you a handle on your system pager (but checks the environment variable PAGER in case the user might want to specify one). If the first (filename) argument is non-NULL, the pager will see it as an argument. Otherwise, it will act as through the second argument (a file pointer open for read) were stdin.

nstrip.c -- This function strips trailing whitespace and line-feeds from a string.

peopen.c -- A popen(3)-workalike for secure programs. This spawns a bare process rather than a shell, but argument tokens beginning with > or < are interpreted as in a shell command. No other shell metacharacters are accepted.

prefix.c -- This function does prefix matching of a string against another string.

procopts.c -- This function provides command-line option processing driven by an options description structure described in procopts.h.

regexp.c -- A slightly hacked version of Henry Spencer's regexp(3) routines, used for implementing (what else?) regular expression commands.

savestr.c -- This function saves an allocated copy of its string argument.

server.c -- Assists you in controlling a child process used as a server. Presently only used to implement ed-newsipc.o.

setadd.c -- Implements set addition and subtraction for token lists.

slist.c -- These routines use dballoc.c to maintain string lists. Entry points support reading a file into a string list, checking for the presence of a string by prefix matching, and writing an in-core string list to a file.

spawn.c -- Routines for running programs under control of news code (at the moment, just a front-end to the fork/exec pair). Someday versions of this for VMS, OS/9 etc could make the upper layers of the news code more OS-independent.

strindex.c -- A basic find-substring in string routine.

vms.c -- Supplies functions to support the news code under VMS/EUNICE.

xerror.c -- Provides error exit and logging services and an open-file-or-die function. This collides (intentionally) with the log.c::xerror() of libpriv.a. When using libport.a with libpriv.a, link libpriv.a **before** libport.a in order to get the libpriv.a version.

10. The news library functions and random support modules

The libnews.a library is a group of small modules shared by many news programs. A brief description of each member follows.

articleid.c -- This module has two major entry points. One function decorates an article file with a new Message-ID including timestamp, article text hash and home site. The other, given such an ID, cracks it into a structure holding said information in program-accessible form.

artlist.c -- Functions to support parsing article locations from a blank-separated list of article references of the form <groupname>/<article number>. Used by the history file access code and by the nntp hooks.

escapes.c -- functions to expand standard C-style escape sequences in strings or the input screen. If the RNESCAPES configuration attribute is on, most of the %-escapes recognized by rn are also expanded (see the reader manual pages for details).

fascist.c -- A function used to check access permissions to newsgroups if the FASCIST (or COMMUNIST) configuration switch is set.

feeds.c -- These functions provide access keyed by system name to the contents of the feed file that describes the news communication links to neighbor sites.

getfiles.c -- Functions that seal away the operations involved in mapping article tree locations to files (so that, in particular, users may link with alternate modules that use an nntp-like server for such access).

mailbox.c -- Functions for reading and writing mailboxes and digests of various types. Includes code for saving articles with or without headers, with or without mailbox delimiters, and with or without truncating the save file (all according to flag bits in a control argument).

msgopen.c -- This module provides functions for opening and closing RFC-822 message files. They hide the difference between messages stored in clear or compressed form.

myorg.c -- Contains the organization() function to retrieve the current user's organization as set by the contents of the ORGANIZATION environment variable.

newsinit.c -- Global initialization code for netnews tools. Generates the locations of important library files for the rest of the code to use.

ngmatch.c -- Matches a newsgroup name against a list of newsgroups, implementing the standard prefix-match, !, any and all conventions.

ospawn.c -- Spawns a given command with an -o <num> option appended; the command is expected to write status data on file-descriptor <num> before exit. A pointer to said status data (in a static buffer) is returned.

rdactive.c -- The 'read' side of the active-file interface, see above.

rdhistory.c -- The 'read' side of the history-file interface, see above.

rdnewsrc.c -- The 'read' side of the newsrc-file interface, see above.

sysmail.c -- The functions in this module provide an interface to the system mailer suitable for use by netnews transmission programs.

ttyin.c-- functions for interactive fetching of user commands on a line-oriented device. Handles all the messy details of buffering, recovering from signals and re-prompting if necessary.

ttyout.c -- Primitives for output on line-oriented interfaces, see above.

10.1. The poster-support library

These modules provide low-level support for news database modification by news-privileged posting programs and prowlers.

collect.c -- this function collects standard input into a temp file with name constructed from a given template.

lock.c -- These routines are used to get exclusive access to the netnews database during critical sections of the code. They use POSIX or 4.2BSD-style mandatory file locking or System V semaphore operations where they can, and a busy-wait loop on lockfile linking if they must. See D.port/lockname.c.

log.c -- Provides error exit and logging services and an open-file-or-die function. These functions, unlike their equivalents in xerror.c, log to the netnews transaction and error log files.

mung.c -- This functions support in-place munging of article headers.

ngprep.c -- these functions are used to compile the newsgroup and distribution lines of the current header into an internal form more convenient for C processing.

textwalk.c -- The entry point in this library does a depth-first traverse of the text directory, applying at each node a pointer to a function which is passed location and node-type arguments. See the code for details.

wractive.c -- The ‘write’ side of the active-file interface, see above.

wrfeeds.c -- The ‘write’ side of the feeds-file interface, see above.

wrhistory.c -- The ‘write’ side of the history-file interface, see above.

10.2. The reader-support library

The libread.a modules provide support for the standard reader abstractions. They suffice (with a top-level command interpreter) to construct a line or screen-oriented news reader.

browse.c -- primitives for browsing through news in page-sized chunks. Uses the insrc.c facility. They manage five source objects; one for articles, a second for index page files, a third for group lists, a fourth for help information, and a fifth for raw access to article headers. Use these to construct readers that do their own page chunking.

checkinit.c -- Initialization code for news browsers (including readers and checknews).

clockdaemon.c -- defines a clock daemon routine which, called periodically from a reader, can do mail checks, get the current name, and call a trip function (usually to save the .newsrc file).

gcmd.c -- A generic command interpreter that dispatches most of the vnews command set. See the detailed discussion above.

getart.c -- The article-getter primitive for readers. This is the *only* piece of code in the reader libraries that knows where articles actually live and what their representation is. Hacks to do things like cracking mailboxes and digests will go here someday.

insrc.c -- this module allows you to switch between multiple source objects, which act like arbitrary-length text line queues with special filtering properties. Each source object can be written to (lines longer than the terminal width are folded) and read from, and seeks to given lines are supported.

macros.c-- support for the generic user-defined macro facility.

nextmsg.c -- Implements a get-next-article primitive and some related services using the facilities of rdactive.c and rdnewsrc.c. See the detailed discussion above.

reader.c -- Code used by interactive readers to navigate through discussion trees, set and return to placemarks, and do backtracking lives here. Builds on the facilities of session.c.

readinit.c -- Contains the main initialization and wrapup functions for interactive readers. Also includes a rather motley assortment of header-display, article-printing interrupt-handling functions common to all readers.

rfuncs.c -- Logic for the generic reader commands dispatched by gcmd.c (quite usable separated from it,however).

session.c -- Code for the reader-session abstraction. Article-marking and subject-following logic lives here.

vinfoline.c -- Code for generating group and article-subject information lines in a standard format. Isolated here because it’s command-set and I/O-manager independent.

wrnewsrc.c -- The ‘write’ side of the .newsrc-file interface, see above.

10.3. The newsfilter-interface library

Includes code for both ends of the pipe IPC between newsfilters and readers, and some support functions for building newsfilter processes. Details of how to use these modules are given in their header comments.

filter.c -- Implements a state-machine ‘harness‘ for filter logic that sequences five user-supplied action functions according to IPC with a reader process. Also interprets newsfilter options.

lnfilter.c -- Supplies action functions for filter.c that implement a generic newsfilter. This generic newsfilter will know how to respond to reader requests and where to read user commands from. It must be supplied with article and group interest-scoring functions and a command-processing hook.

savescore.c -- Automatically handles caching of results from newsfilters run off-line, and retrieval of cached interest scores at reader run-time.

tofilter.c -- Support for the reader side of IPC to the newsfilter.

transact.c -- Low-level routines for IPC used by both readers and filters.

10.4. The visual-interface library

Includes the visual-interface support layer described earlier.

10.5. The posting-functions library

Includes user-interface-independent routines for editing postings and replies and feeding postings to rnews.

editmsg.c -- User-interface-independent functions for article and reply composition and header munging, for postnews and readnews.

postart.c -- User-interface-independent functions for postnews and future things like it.

newpost.c -- User-interface-independent functions for feeding postings to rnews.

originate.c -- User-interface-independent functions for originating new news.

10.6. The UUCP hacks library

Includes routines that examine and diddle the UUCP control files to get functions not included in the standard UUCP entry points. This code is fragile; the UUCAST feature in particular may break if your UUCP configuration is at all nonstandard. Read and understand the source code before using it.

uuq.c -- return the size of the outgoing UUCP queue for a given system.

uucast.c -- multicast a UUCP command to multiple neighbor systems.

sevenbit.c -- code for transmitting binary data over 7-bit lines.

11. Conventions to follow when modifying the code

The netnews software is a trust of the UNIX community. If you have a neat feature to add, by all means do it and reap your share of glory. Please, though, for the sake of other netnews hackers, try to add and modify code in ways that are clean and consistent with the style of the existing stuff. Some guidelines for this follow:

When you post or otherwise disseminate a modification, do it as a set of diffs (preferably context diffs) of the old and new code *and* relevant documentation. Put the new code inside an #ifdef/#endif with the old code in the other branch, so that those applying the patch can compile either with it or without it for testing purposes. And cite the revision number!

Whenever possible, hide system dependencies behind macros and conditional includes in the system.h file (that’s what it’s there for).

The News 3.0 (beta level 7) code is mostly organized into modules that each implement an abstract data type (Someday perhaps this stuff will be recoded into C++ and each of these will become a class). Each module has one associated .h file including extern declarations for each of the associated files and any macros used with the package (explanatory commenting of these declarations is encouraged). Most module .c files have header comments in UNIX manual page format that describes their interfaces in detail. Go thou and do likewise!

The enhancements to the active, .newsrc and expire formats were intended partly to centralize all state information for given abstract data types in a single data file each. Accordingly, it is strongly suggested that

* all new per-newsgroup attributes be put in new fields of the admin file and accessed through active.c.
* all new per-user attributes which the user is supposed to be able to modify be represented by new syntax in the .newsrc file and accessible through newsrc.c.
* all new per-article attributes be represented by fields in the history file and accessible through {rd,wr}history.c.

Most of the major modules include code for an interactive test main. The small effort required to adapt one of the existing test drivers to your new module will amply repay itself when you need to test and debug.

The code uses the UNIX system naming convention for public typedefs; such names end with ‘‘_t’’, and no other identifiers do.

The symbol ‘‘private’’ is defined to expand to ‘‘static’’ and used for top-level static declarations in the news software. The intent is to make it easy to compile the source for profiling via CFLAGS="-Dprivate=/* */". Please follow this convention when adding and modifying code.

Most of the code uses Berkeley style with 4-space indentation. If this is not to your taste, try at least to use a consistent and recognizable style.

Gotos are occasionally necessary for jumping out of multiple nested loops and such, but if your hackery includes too many of them the maintainers will probably become suspicious of your sanity and throw it away. *Any* gotos that jump inwards into a scope will subject you to an ancient Gypsy curse too horrible to be described here.

The News 3.0 (beta level 7) sources lint without errors or warnings under AT&T System V Releases 2 and 3 and under (at least one version of) 4.2BSD (considerable pains were taken to achieve this). Those who diddle with them are requested to keep lint happy, as it really does assist in the detection of bad craziness.

All occurrences of NULL that normally get compiled are accompanied by a cast to the data type actually being used. This may seem like overkill but it prevents insidious bugs in situations where sizeof(char *) != sizeof(int), in particular on the 286 and other brain-damaged architectures.

Please reserve the -D command line switch on utilities for setting the ‘debug’ global and disabling database-altering code. All such code should be wrapped in #ifdef/#endif pairs contingent on the DEBUG configuration attribute. Please reserve the -v command-line switch for controlling the public variable Please try to follow existing practice in the code by referring to debug levels by manifest constant names of the form V_* declared near the module beginning (so that a simple grep of the source can teach someone reading the code what various levels of verbosity mean). Finally, please arrange that -D <num> be equivalent to -D -v <num> in new tools (so that debug at level n implies level n verbosity).

Do not assume that string literals are modifiable, i.e. by mktemp(3). This code should compile and run in environments where C constants really are.

Do not assume that auto variables are zeroed out by default. Don’t even assume that internal static variables (static variables declared within function scope) are zeroed -- some XENIX compilers have severe braindamage and don’t do this!

Try to adhere to the following discipline for handling of allocated storage. It helps the debugging process, and virtually eliminates bad-pointer errors. A pointer to allocated (dynamic) memory should never be assigned the address of pointer to static memory, nor vice-versa (violating this rule invites chaos as free() tries to interpret some region of static memory as a malloc arena). Each block of allocated memory should at all times have exactly one permanent pointer to it (a permanent pointer is a pointer whose value may travel outside of the local block). If two things need to point to the same string, malloc the space twice. If you need to modify the string using one pointer and see the result on the other, there’s probably a better way to do things. After freeing a block of memory, NULL out the pointer unless it is either immediately assigned or an automatic that will disappear on block exit. Before assigning to such a pointer, add #ifdef DEBUG code to check to see if it is non-NULL. If so, free it and log a programming error. If you feel you need to violate these guidelines, *document the reason*!

12. How to mung the Makefile, config.h and news.h files

This section describes the uses of the various preprocessor and make-macro attributes set up in Configure, in case it fails and you need to hand-hack a configuration to bring up the system.

12.1. Level 0 Configuration Attributes

The following are set by the Level 0 configuration, which should typically only need to be done once during first-time installation. The attributes listed below are the only level 0 attributes actually used by the USENET distribution code; a few others are generated and have been retained with a view to merging this process with rn configuration.

One of the following attributes will be defined at the beginning of config.h:

V7 -- Version 7
SIII -- System III
BSD4_1 -- Berkeley 4.1
BSD4_1C -- Berkeley 4.1C
BSD4_2 -- Berkeley 4.2
BSD4_3 -- Berkeley 4.3
SYSV1 -- System V Release 1
SYSV2 -- System V Release 2
SYSV3 -- System V Release 3

Thus specifies which variant of UNIX you're running. This information is mostly just used to set one of two symbols V7 or USG to compile the proper code for tty control and a few other wrinkles. See system.h for details.

12.1.1. SMALL_ADDRESS_SPACE

Define this if your machine has 16 bit (or less) pointers. If you are on a pdp11 or one of Intel's 16-bit disasters this is automatically defined.

12.1.2. strchr, vfork and friends

The next group of attributes includes several defines that mask away such features as the void type and Berkeley vfork() on systems that do not have them. These should be readily transparent to anyone with enough knowledge of C and UNIX to hack around at this level.

12.1.3. EUNICE, TERMIO, NORMSIG

These attributes are inherited from the old rn Configure and not currently used in news. They are left in for possible future use.

12.1.4. CURSES If this is undefined, we link in the curses emulation provided with the distribution. If it's off, we use the system's native curses implementation.

12.1.5. STATLINE Tells whether displays on the system should be assumed to have a status area displaying date/time and a mail icon. Not presently used.

12.1.6. FCNTL, LOCKF, LOCKING, MKDIR, RMDIR, GETCWD, GETPWENT, USTAT, DRAND48

These are defined/undefined according as the corresponding system or library calls are found in their accustomed places.

12.1.7. KEYPAD, LIBNDIR, USENDIR, VMS

These signal the presence of (respectively) SVr2 keypad features in curses(3), two different flavors of directory routines, and an underlying VMS file system. They are only used in small, localized sections of code.

12.1.8. GCOS

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 GCOS field of /etc/passwd. If your system has such a database, GCOS should be defined. If not defined, articles posted will only receive full names from local user information specified in NAME or ~/.name by the user. If you have a nonstandard GCOS format (not finger or RJE) it will be necessary to make local changes

to fullname.c as appropriate on your system.

The next six attributes trigger alternative ways for the software to get the system name (uucp name) at startup. Only one of these should be defined.

12.1.9. UNAME

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

12.1.10. GHNAME

Define this if the 4.2BSD gethostname 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.

12.1.11. UUNAME

Define this if your sitename lives in /etc/uucpname or /local/uucpname.

12.1.12. WHOAMI

Define this if your sitename is the value of a #define sysname in the file /usr/include/whoami.h. Note that the lookup is done on the text of the file itself at startup time, so binaries that use it are portable.

12.1.13. HOSTCMD

If this is defined, its value is taken to be the text of a command that yields the site name on stdout; it will be popen(3)ed and read at startup.

12.1.14. SITENAME

If defined, the value of this attribute is the sitename to be compiled into the software.

12.1.15. PAGE

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

12.2. Level 1 Configuration Attributes

The following attributes are defined by level 1 configuration. Names followed by (*) may be overridden by the attributes file if RUNTIME is on.

12.2.1. NONLOCAL

If this attribute is on, the readers and postnews are compiled to use a network service library for their history, active file and article file accesses. The particular network service library linked in may vary according to network medium and service type. As of January 1988 only one such service library is defined, for use with NNTP.

12.2.2. NNTPSERVER*

This attribute (used only if NONLOCAL is on) gives the name of the NNTP host to connect to.

12.2.3. 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 NEWSADMIN do not need to represent the same user.

12.2.4. NEWSGRP

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

12.2.5. LIBDIR*

This directory will contain various news executables. It is normally /usr/lib/news. ADMDIR*

This directory will contain (sharable) news database files. It is normally /usr/lib/news.

12.2.6. TEXTDIR*

This directory contains subdirectories in which news articles will be stored. It is normally /usr/spool/news. If you are familiar with the internals of older versions of news, note that this location used to be called SPOOLDIR.

12.2.7. SPOOLDIR*

This is the directory where spooled incoming news is kept (assuming SPOOLNEWS is on) before rnews -U gets at it. It is normally /usr/spool/news/.rnews. If you are familiar with the internals of older versions of news, note that the root of the article tree, formerly called SPOOLDIR, is now TEXTDIR.

12.2.8. BATCHDIR*

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

12.2.9. BINDIR

This is the directory in which *readnews*, *postnews*, *vnews*, *mailnews*, and *checknews* are to be installed. This is normally /usr/local/bin.

12.2.10. UUBINDIR

This is the command directory where rnews will live, normally /usr/bin. If you decide to set UUBINDIR to a local binary directory, you should consider that the **rnews** and (if you are fed by 2.10.X sites) **cunbatch** commands must be in a directory that can be found by **uuxqt**, which normally only searches /bin and /usr/bin (unless you modify uuxqt).

12.2.11. FROMNAME*

This gives the sitename that the news software will insert in the From lines of mail and news originated at this site. It should be a fully-qualified Internet domain name such as ‘snark.uu.net’ or ‘ucbvax.berkeley.edu’.

12.2.12. PATHNAME*

This should be the name you want to appear in the Path lines of outgoing news.

12.2.13. TRUENAME*

This should be the name you wish the news system to give as originator when originating notification mail to the administrator from the site.

12.2.14. ORGANIZATION*

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 article. 40 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’’, or ‘‘U.C. Berkeley’’ or ‘‘MIT’’ or ‘‘Computer Corp. America, Cambridge, Mass’’.

12.2.15. ORGDISTRIB

If this attribute is defined it should be the name of the distribution you consider local for administrative purposes. Group creation and deletion messages with distribution lines that don't exactly match ORGDIST will then be intercepted and mailed to the news administrator. Also, inews with -C will create the group only within the ORGDIST distribution (the longer newgrp form of inews invocation can still be used to send creation messages to wider distributions).

12.2.16. MAILSERV*

This should be a command string invoking invoking your mail back end with whatever switches are necessary to cause it to accept a message and accept a list of mail destinations as command line arguments.

12.2.17. MAILFMT*

This attribute defines the mail format used by your mailer; it controls the way news is saved to mailbox files. Accepted values are "V7" and "MMDF". The default is V7.

12.2.18. MAILCHAR*

This defines the character expected to begin mailbox files ('F' by default).

12.2.19. BACKBONE*

This symbol should have as its value the address of a backbone site, with the string "%s" where a user address would go. This will be used as a sprintf(3) fprmat to generate submission addresses for submissions for moderated groups for which there is no local entry in the mailpaths file.

12.2.20. SMARTHOST*

If your system has a mailer that understands ARPA Internet syntax addresses (user@site.domain) and can forward to them, set this to "%s". If you have a neighbor that can do domainist routing, set it to the address of that neighbor with a %s where the domainist address should go. Setting this attribute to the empty string disables domainist features.

If SMARTHOST is nonempty, readers will use the From or Reply-To headers (which typically contain Internet-style addresses) to generate the mail paths for replying to postings. If you leave it disabled and replies will use the Path header (this gives you less reliable return paths).

You can turn this on if you have the UUCP Project's *smail*(1) autorouting mailer back end installed.

12.2.21. 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.

12.2.22. FASCIST*

If this attribute is defined, each run of postnews or rnews checks ADM/authorized to see if the user is allowed to post to the given newsgroup. If the username is not in the file ADM/authorized then the default in the symbol FASCIST is used. For details on the format of the file "authorized", see

12.2.23. COMMUNIST*

If this symbol is defined, each run of a reader checks ADM/authorized to see if the user is allowed to read the given newsgroup. If the username is not in the file ADM/authorized then the default in the symbol COMMUNIST is used. For details on the format of the file "authorized", see

12.2.24. 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** will be run. Normally, this program should be placed in LIBDIR.

12.2.25. LEASTUID*

This is the lowest numeric user ID that will be considered a ‘‘real user’’, i.e. one that expire will take into account when deciding whether or not articles in a volatile group. This feature is included so that accidental creation of a .newsrc file by rarely-accessed system accounts like root or bin won’t effectively disable the volatile-groups feature.

12.2.26. 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 (default 4 weeks).

12.2.27. DFTXMIT

These are the flags uux will be called with. You always want to use the − flag, which takes message input from standard input. Experience has shown that the −r option to suppress immediate startup is a good idea. You also want to suppress the back-mailing of transmission-status reports that uux normally does.

Version 7 and Berkeley 4.1: use −, −r. If you have or can get source, patch your uux to accept the −z and −n options (which should suppress mail notifications on success and failure of uux respectively) and use −z.

System III, System V Release 1, Berkeley 4.2 and 4.3: use −, −r, −z.

System V Releases 2 and 3: use −, −r and −n only; on these systems, −n suppresses both kinds of notification, and −z does the wrong thing.

12.2.28. ZRETURN

If you have any neighbors that can’t enable the −z option, define ZRETURN. This will force all status returns from rnews to be zero, achieving the same affect as if they spawned their uux jobs with −z. This is good to try if your feeds complain of nuisance mail coming back to them after they send news.

12.2.29. UXMIT

This is the default command used if the U flag is present in the flags portion of a feeds file line. In this case, the 2nd %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. Note that System III and System V Release 2 have this option already installed.

12.2.30. MULTICAST

If your transport mechanism supports multi-casting of messages, define this. See the description of the ‘M’ and ‘O’ options in the Installation Guide for details, and UUCAST for one supported use.

12.2.31. UUCAST

If this is defined, you can try to use multicast addressing over UUCP. Warning: the code that does this diddles UUCP workfiles and is very implementation-dependent. Recommended for uucp wizards only. Use at your own risk. MAILFRONT

The name of the default interactive mailer front end on your system, to be used for composing replies to articles. On archaic versions of UNIX this may be the same as MAILPROG. More recent ones will plug BSD Mail or mailx in here. Note that this sets a default which can be overridden by the MAILER environment variable.

12.2.32. TMAIL

If defined, this should name a version of the Berkeley *Mail* program that has the −T option. The **-M** option to *readnews* that this attribute used to control is now defunct; TMAIL now names the default mailer used by the new **mailnews**(1) interface if it's defined (if not, the local default mailer will be used).

12.2.33. 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,news.announce.important**.

12.2.34. 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**.)

12.2.35. N_UMASK

Mask for *umask(2)* system call. Set 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.

12.2.36. 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.

12.2.37. NONEWGROUPS

NONEWGROUPS is undefined by default. If it 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.

12.2.38. SPOOLMIN*

This is the threshold number of free blocks that must be available on your spool device before the *sendbatch*(8) utility will run. Try to set this just above the average size of your daily throughput to prevent blowouts.

12.2.39. NICENESS

If NICENESS is defined, rnews does a nice(NICENESS) before processing news.

12.2.40. SPOOLNEWS

If this symbol is defined, inews and rnews append all news received from other sites to a batchfile under SPOOL, which is processed and broadcast the next time expire or rnews -U runs. You can use this to defer your news processing till some crontab-scheduled time in the wee hours.

12.2.41. SPOOLPOST

If this symbol is defined, news originated at your site is also spooled.

12.2.42. RUNTIME

If this symbol is defined, runtime configuration features are enabled, and the newsattr() function looks for various netnews attributes to be set in ˜NEWSUSER/attributes.

12.2.43. DEBUG

This switch compiles in code for the -D options of checknews, the readers, rnews, sendbatch, expire, uurec, postnews and inews. These options are very useful for troubleshooting.

12.2.44. TMNCONVERT

Causes the code to read and write 2.10.3-format history files, but won't handle databases made with the old dbm(3) code.

12.2.45. FEEDBITS

If this is defined rnews will do all its subscription computations at startup time when it sees a batch. This will involve m * n ngmatch calls, where m is the number of groups in the active file and n is the number of entries in the feeds file. Doing this may avoid a much larger number of ngmatch() calls during normal processing (in particular, this wins if there are more than m newsgroups referred to in the average batch). See also CACHEBITS.

12.2.46. CACHEBITS

If FEEDBITS is on this switch introduces a further refinement; it causes subscription data to be cached in LIB/feedbits. On normal startup the bitmap initialization code will go read this file instead of doing ngmatch() calls. If this file is nonexistent or the feed file has been modified since it was last generated, a new one will be written.

12.2.47. HASHGROUPS

Trades memory for group-lookup speed. See the commends in D.news/rdactive.c for details.

12.2.48. NEWCTRL

NOTE: This Feature Not Yet Supported. Hack At Your Own Risk. You Have Been Warned!

This experimental switch enables compilation and installation of a control handler binary separate from rnews but called by it in the appropriate cases. The intent is to trade off slower execution in the uncommon case of control messages for a significantly smaller and hence faster-loading and swapping rnews.

12.2.49. COMPRESS

This is the command used to compress outgoing news if the C option is specified in a destination system's feed entry. You should not normally need to change this.

12.2.50. DECOMPRESS

This is the command used to decompress when rnews detects that incoming news is compressed or if the -d option to rnews is used. You should not normally need to change this.

12.2.51. SIGLINES*

The value of SIGLINES (which defaults to 4) is the upper limit on the number of signature lines that postnews will copy from a user's .signature file.

12.2.52. QUOTELIM*

If this symbol is defined it causes inews to reject articles that contain more than its value as a percentage of inclusion lines. What inews thinks is an inclusion line is subject to hackery, see the header comments in inews.c for details.

12.2.53. LONGTEXT*

A minimum linelength that qualifies an article as long, i.e. subject to quote checking (25 is a typical value). Only long articles have quote checking done on them.

12.2.54. FEEDBACK

This symbol enables the USENET reader feedback features. If this symbol is on, readership counting and the Meta-P ("praise") and Meta-C ("condemn") reader commands are enabled. These update a database at LIB/feedback with information that can be abstracted and mailed to a central collection point periodically.

12.2.55. SORTACTIVE

Define this if you want the news groups presented in the order of each persons .newsrc instead of the active file.

12.2.56. 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. Configure looks in reasonable places for things that might be editors when computing the default for this; it knows about emacs, vi and ed.

12.2.57. CHECKMAIL

If defined, reader interfaces will show you when mail comes in during a news-reading session.

12.2.58. SHOWTIME

If defined, reader interfaces will display the current time during a news-reading session.

12.2.59. DIGPAGE

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

12.3. Space-conservation symbols

These are normally define, but may be turned off in confignews.h to save text space in the executables

12.3.1. ENCODE

This attribute must be defined in order to enable the code that implements the E transmission option, otherwise using it will cause an error to be logged.

12.3.2. DECODE

This attribute must be enabled in order to compile the code that does c7unbatch reception. If it’s not, calling rnews via the c7unbatch link will log an error.

12.3.3. ZAPNOTES

Define this if you want old style notesfile ids in the body of the article to be converted into NF-ID: fields in the header by inews.

12.3.4. BNCVT

Enables processing of old-style ‘bnproc’ news batches.

12.4. Parameters in news.h

Some symbols that configure various features in and out are discussed in the ‘‘Installation Guide’’ section on porting the code.

12.4.1. BIGGROUPS

If you care, this can be defined to force article numbers to be type long; by default they are int.

12.4.2. CMPMAGIC

This string gives the magic bytes expected to head compressed files. It is used by rnews to detect them.

13. Troubleshooting

If your users are getting "session trail corrupted" messages at the end of reader sessions, it probably means your malloc(3) is losing. Shorter read sessions (which do fewer mallocs) will avoid the problem. The syndrome is harmless, except that the user's feedback gets lost.

If you get strange behavior in readers following - commands, try disabling the RUNCOUNT space optimization in session.c -- there are no bugs known there at release time, but the extra logic and data-structure complexity involved there troubles the author.

If you see a TROUBLE message in your logfiles that indicates that rnews or expire has died, look in TEXT/.tmp. Typically, it will contain a cltext* file and a newsart* file. The former will be the clear text of the input batch it was processing, and the latter a clear copy of the article that it died on. The core file from the run will also land here.

14. Possible futures

The newsgroup aliasing in post.c should use the dballoc routines.

Someday (on System V versions) all the database-handling logic should move into a server process accessed though an IPC transaction protocol by all readers, posters, and prowlers.

The astute will notice that some of the rn configuration symbols have been imported into config.h without being used by the existing USENET source. This was done for future-historical reasons; someday, rn might be merged into the netnews distribution.

Erik Fair suggests "Given that there are so many newsgroups now, why not take a heirarchical approach to presenting people the newsgroups? (i.e. show [neophytes] first the seven top levels + whatever else is around (local stuff, alt, bionet, etc.) and ask which tree they'd like to go down?)."

The whole thing should be moved into C++. ;-)