The Interface Between C News And The Outside World

Henry Spencer

Dept. of Zoology
University of Toronto

Intro and Generalities

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

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

Directories And Userids

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

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

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

Unix Utilities

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

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

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

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

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

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

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

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

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

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

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

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

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

Libraries

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

Networking

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

Highly System-Specific Things

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

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

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

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

Input

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

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

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

To actually process incoming news, *input/newsrun* gets invoked to decompress the spooled batches and feed them to *relay/relaynews* (see below). There is an option for *newsspool* to invoke *newsrun* when a batch is spooled, but a (usually) preferable method is to have *cron* invoke *newsrun* once an hour. *Newsrun* does its own locking to prevent multiple occurrences running simultaneously. There is a related program,

input/newsrunning, that can be used to set or clear a flag that stops *newsrun*; this may be a useful tactic if *newsrun* should not run at certain times. Both *newsrun* and *newsrunning* must be run as *news*.

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

News Readers

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

Output

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

Expiry And Related

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

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

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

Reboots and Administration

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

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

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