Setting up C News on 2.11 BSD

1. Preliminaries

This document assumes that you already have UUCP configured and talking with at least one other machine, and that you have `uucico` running on a regular schedule. If either of the above does not apply, see my 2.11 BSD UUCP setup guide (instructions for setting up a schedule to run `uucico` are at the end of that guide).

First, on the Pi, grab the sources for C News from my GitHub repository:

```
$ git clone https://github.com/jwbrase/CNews-PiDP-11
```

The repository contains the original sources, my modifications, and a patch that, when applied to the original sources (on a modern machine, not on 2.11 BSD), produces my modifications.

A note on my modifications:

> My modifications to the sources fix an assumption in the DBZ library that a post-K&R compiler is being used (which causes a switch statement to fail to compile), and assumptions in the statfs handling code that we're running on a 32-bit machine (which also causes a compile failure). They also add a shebang line to a number of shell scripts that will not run on 2.11 BSD if `#!/bin/sh` is not explicitly specified.
>
> C News also sets a number of environment variables, which end up having a large total size, in various shell scripts that it uses, and with memory being limited due to the 16-bit address space), `awk` seems to use a lot of memory when processing numeric arguments, leading to frequent crashes in C News scripts that invoke `awk`. I have edited every script in the package that uses `awk` to run `awk` with a reduced environment, which seems to clear up the issue, if you get any memory faults associated with any C News scripts that leave behind a file called "`awk.core`" in the working directory in question, it probably means that I missed something. You can put a "`set`" statement in the script just before the awk line to get an idea what the environment is, and then change:
>
> ```
> awk '{ print( "Hello world!" ) }'
> ```
>
> to something like:
>
> ```
> (PATH=/bin BIG_ENV_VAR="" exec awk '{ print( "Hello World!") }')
> ```
>
> The longest environment variable in most of these scripts seems to be `PATH`, in most cases truncating this to "`/bin`" works (we can truncate `PATH` to the empty string if we invoke `awk` as `/bin/awk`, six of one, half a dozen of the other).
>
> A minor issue is that the make recipe for the regression tests tests the locking mechanism used by C News, and this test fails for reasons I have not been able to determine. However, the same commands used in the makefile seem to run fine when tested through a shell script or

interactive shell, so I have simply commented out the line in the makefile in my patches. If anyone has any insight on this (why it's happening, how to fix it, if ignoring it is safe), please let me know.

I call my modifications the "PiDP-11 release" of C News. The next thing to do is to tar up the PiDP-11-release directory so we can send it to the 2.11 BSD system:

```
$ cd CNews-PiDP-11/PiDP-11-release
$ tar -cvf C News.tar .
```

Now we need to transfer the file to the 2.11 BSD system. If you have IP connectivity in your 2.11 BSD instance, you can just ftp it over. However, since C News requires a working UUCP system to function, lets use this as an opportunity to make sure that is working. We'll transfer the file by UUCP.

```
$ uucp C News.tar 211name\!\~
$ sudo ucico -s 211name
```

Where "211name" is the UUCP name of your 2.11 BSD system. This will place the file in the public UUCP directory (/usr/spool/uucppublic, if you followed my howto on setting up UUCP) on the 2.11 BSD machine.

Once this is done, go to the 2.11 BSD system and add lines to `/etc/passwd` and `/etc/group` for a user and group called "`news`". It's a good idea to edit `/etc/passwd` with `vipw` instead of normal `vi`, as `vipw` does checks to make sure that the `/etc/passwd` doesn't contain any syntax errors once you're done editing.

`/etc/passwd`:

```
news:*:$UID:$GID::0:0:C news,,,:$NEWSHOME:$SHELL
```

`$UID` and `$GID` should be replaced with user and group ID numbers that do not belong to any existing user/group listed in `/etc/passwd` or `/etc/group`. `$NEWSHOME` will be the home directory for the news user. You will end up in this directory any time you log in as news. I recommend at this point that you read ahead to the instructions on running "`quiz`" to configure C News, and specifically, the discussion of possible answers to the question "`Where should control files go?`", which is used to define an environment variable that the C News scripts use called `$NEWSCTL`. A sensible default for the news home directory is `$NEWSCTL/newsmaster`, so you should know what value you'll be using for `$NEWSCTL` before you continue with these instructions. `$SHELL` should be whatever shell you want to use when performing maintenance as the news user. If you don't know what to put here `/bin/sh` is fine. You can name the news user something else if you want, but "`news`" is a sensible default.

`/etc/group`:

```
news:*:$GID:news
```

`$GID` should be the same `$GID` used in the `passwd` file.

Now run:

```
$ passwd news
```

To set a password for the `news` user.

Now go to the news source directory, and, as root, run:

```
$ chown -R news.news .
```

Certain parts of the build process need to be run as `news`, and will choke if `news` doesn't own the sources.

If you haven't yet created the home directory for the news account, do so now:

```
$ mkdir $NEWSHOME
$ chown news.news $NEWSHOME
```

If `$NEWSHOME` is to be several directories down from the nearest existing directory, you will need to create each directory. For example, if `/usr/etc` does not yet exist, and `$NEWSHOME` is to be `/usr/etc/news/newsmaster`, you'll need to create `/usr/etc`, then `/usr/etc/news`, then `/usr/etc/news/newsmaster`.

2.11 BSD defaults to a `umask` of `026`. This makes it so that files are, by default, not readable to all users. However, C News makes heavy use of shell scripts, which will often be run by users other than the news user. The shell needs to be able to read these scripts, so they will need to be world readable. Therefore, you should set your `umask` to `022` when building and installing C News, which tends to be the default on modern systems anyways. The most straightforward way to do this is to run:

```
$ umask 022
```

every time you log in as `news` or `root` during the build process, but it can get a bit repetitive. If you're using `csh` or `tcsh` for those accounts, you can add the above to `.cshrc` or `.tcshrc` file for the user in question, and it will be run automatically when the shell is launched, or you can add it to `/etc/csh.cshrc` to affect the whole system. Unfortunately, no equivalent file exists for `sh`, the closest equivalent is `.profile`, which is only read when the shell is launched as a login shell, so if you log in directly as a given user, or use "`su - username`" (the "`-`" causes the shell to act as if it had been launched by `login`), the `umask` will be set, but if you just use "`su username`", you'll still be running with the `umask` of the parent shell. Even `/etc/profile`, which more modern variants of `sh` use for systemwide defaults, doesn't seem to be read by 2.11 BSD `sh`, and isn't listed in its documentation. The default `umask` can also be set when compiling the kernel, but that is outside the scope of this document. If the umask remains at `026`, it can cause subtle issues, like news batches arriving on the system, but not making it to the newsspool. It will also cause the "weekly run output" e-mail that root periodically receives to report lots of "cannot open /some/file/or/directory" errors when rebuilding the find database. Try to make sure that your `umask` is set to `022` throughout the build and installation process (unless you're using `(t)csh` for both `root` and the news user, and have the `umask` set in `/etc/csh.cshrc`, there will be at least some situations where you will need to set the

`umask` manually after a login or `su`), because if you don't you'll end up needing to reset the permissions on a whole bunch of files, which is really tedious.

Now, either switch to the `news` user account:

```
$ su - news
```

or telnet in to another terminal and login as `news`.

Now, unpack the news tarball from your public UUCP directory into a new directory under `/usr/src`:

```
$ mkdir /usr/src/cnews
$ cd /usr/src/cnews
$ tar -xvf /usr/spool/uucppublic/C News.tar .
```

Now, run the following command:

```
$ ./quiz
```

This is a configuration script that asks questions about your system and tailors the build process according to your responses.

The default answer to any question is printed between square brackets [like this], and can be selected just by pressing enter. The first two questions are

```
What user id should be used for news files [news]? <enter>
What group id should be used for news files [news]? <enter>
```

If you put "`news`" into `/etc/passwd` and `/etc/group` as shown above, you can just accept the defaults here. Otherwise use whatever username you chose.

The next two questions deal with where to put the article tree (which stores the netnews messages that the machine has received) and the overview files. I'm not entirely sure I understand everything the overview files do, but the general gist is that they act as a cache of information like subject lines so that your newsreader can present you with a list of articles without having to scan through the entire article tree. The paths recommended by `quiz` put these in the same directory, but they can be stored separately, and are stored separately on what seems to be the standard news system for modern machines, INN. `quiz` tells you what the "traditional" location is for these directories, and then recommends what the developers thought were better locations, but you can use whatever you want. The caveat for `quiz`'s "reasonable locations" is that the 2.11 BSD setup that ships with the PiDP-11 software collection has a small disk image for the root filesystem, and a bigger one for `/usr`, and the recommended location of `/var/news` is on the root filesystem, so you risk running out of space if you install there. If you're running on a PiDP-11 (and are using the provided collection of OSes), or otherwise have a small root disk, you'll probably want to use the traditional location of `/usr/spool/news`.

C News uses the environment variables `$NEWSARTS` and `$NEWSOV` to refer to these directories:

```
Where should the article tree go [/var/news]? /usr/spool/news
Where should overview files go [/var/news]? /usr/spool/news
```

The next question deals with where the various configuration files for C news will go (`$NEWSCTL`). The recommended path, `/etc/news` is what's used by INN in the modern day, but once again, if you have a small root filesystem, you may want to use a location under `/usr` (assuming that's on a separate, bigger disk). I used "`/usr/etc/news`", though that's rather non-standard.

```
Where should control files go [/etc/news]? /usr/etc/news
```

Next we're asked where to put the binaries and scripts for C News (`$NEWSBIN`). The recommended value of `/usr/libexec/news` is fine here:

```
Where should programs go [/usr/libexec/news]? <enter>
```

Once you've specified the above directories, create them and chown them to news, or else make install will run into permission issues later.

Having told C News where it should put its stuff, we now have to tell it where to find system programs. 2.11 BSD keeps a few utilities in non-standard locations, so answer "yes" to specify these directories:

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

Then tell C News to add `/usr/ucb` and `/usr/local` to the list.

```
What is the full name of the directory [/urk]? /usr/ucb
Should it go after (as opposed to before) the others [yes]? no
Is there any other directory which should be searched
to find standard programs on your system [no]? yes
What is the full name of the directory [/urk]? /usr/local/bin
Should it go after (as opposed to before) the others [yes]? no
Is there any other directory which should be searched
to find standard programs on your system [no]? <enter>
```

Accept the default umask for C News to use:

```
What umask should C News use [022]? <enter>
```

You'll now be asked where to put the shell configuration file, where the shell scripts that do a lot of the work for C News pick up their configuration. The default of `$NEWSCTL/bin/config` is sensible:

```
Where should the shell configuration file be
located [/usr/etc/news/bin/config]? <enter>
```

C News now wants to know where to e-mail reports of any problems it encounters. If you take the default answers of "`newscrisis`" and "`newsmaster`", you will have to either set up user accounts under these names (and check them for notifications of problems with the news system), or configure `sendmail` to forward mail for these names to you. Since you're a retrocomputing enthusiast doing this in 2020, and not the administrator for a big 1980s Usenet site, you'll probably be a user on your 2.11 BSD system for as long as it's running, so it's perfectly OK to use your own user account for this. Another alternative would be to have such messages mailed to `root`. In *theory*, as 2.11 BSD has IP connectivity and `sendmail`, you could have this go to any e-mail address, but in practice, spam is a bad enough problem in the modern day that any mail server that you don't personally own is almost certain to reject whatever gets sent. It's best to have these reports mailed to a user account on the same system. If you know how to set up `sendmail` aliases, you can go with the defaults of `newscrisis` and `newsmaster`, rather than using an actual user account.

```
Where should C News mail urgent reports [newscrisis]? root
Where should C News mail non-urgent reports [newsmaster]? root
```

Now C News wants to know how much address space your system has (basically, how much memory any one program can use). The PDP-11 was a 16-bit machine, with each program getting 64 kilobytes (-ish, some PDP-11s gave each program 64k for code and 64k for data, and there were some other tricks that could be pulled, if the OS supported it), so some of the component programs in C News will have to squeeze to fit. We need to tell it that so that the build process will keep things small.

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

C News then asks us if our operating system supports a whole bunch of different system calls, 2.11 BSD supports (or not) the ones shown below:

```
Does your system have  fcntl() [yes]?
Does your system have  fgetline() [yes]? no
Does your system have  getopt() [yes]?
Does your system have  gettimeofday() [yes]?
Does your system have  memcpy() [yes]?
Does your system have  mkdir() [yes]?
Does your system have  putenv() [yes]? no
Does your system have  remove() [yes]? no
Does your system have  rename() [yes]?
Does your system have  strchr() [yes]?
Does your system have  strerror() [yes]?
Does your system have  strspn() [yes]?
Does your system have  symlink() [yes]?
```

Now C News wants to know if we'll be processing news over NFS. This wasn't recommended back in the day (C News says it's not recommended right before it asks the question. It defaults to yes,

presumably because compiling support in when it's not needed is better than not having it when it is), and you almost certainly won't be doing it now. Say no unless you know *exactly* what you're doing.

```
Will processing be done over NFS [yes]? no
```

`quiz` now asks us what database package we want to use. The unmodified C News sources won't compile with `dbz`, but I was able to make a small modification to get them to compile, so we'll be using `dbz`:

```
Do you want to use the "dbz" package [yes]? <enter>
```

Next it asks about using custom versions of `stdio` functions for better performance. They do not work on 2.11 BSD. The build process checks to see if they will work before compiling them into anything, and continues without them if they don't, so "yes" won't break anything, but it won't do any good either. Say no.

```
Do you want to use our stdio speedups [no]? <enter>
```

It then asks if your compiler is good enough to generate in-line code for `strchr()`, and that it's OK to guess. Say no.

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

The next three questions involve whether certain system header files are ANSI-C compliant. The first two are, the third is not. As I recall, I used "yes" for the third without disaster, but the example system I received had a "no" answer here. It's thus probably safest to answer "no".

```
Does your system have an ANSI-C-conforming <string.h> [yes]?
<enter>
Does your system have an ANSI-C-conforming <stdlib.h> [yes]?
<enter>
Does your system have an ANSI-C-conforming <stddef.h> [yes]? No
```

2.11 BSD does have `ranlib`, so answer "yes" to the next question.

```
Does your system use ranlib [no]? Yes
```

Now it wants to know what version of make to `use`. 2.11 BSD make does not know about makefile includes, which C News needs. Fortunately, C News has a wrapper for make that can do includes (which will be relevant for later questions), so we'll point it at that script (replace `/usr/src/cnews` with whatever directory you unpacked the sources into):

```
What is the name of the make to be used [make]?
/usr/src/cnews/conf/maker
```

Now it asks us how to do makefile includes. Say "noway"

```
C News relies heavily on being able to put...
<snip>
...The syntax for this varies...
<snip>
Which one is appropriate [bsd]? noway
```

Next, quiz wants to know if we want to use a different compiler than make's default. The 2.11 BSD cc is fine for this job, so we answer no.

```
Do you want to use a compiler other than make's default [no]?
<enter>
```

Next question: What compile options do we want? The default, -O, to generate optimized code, is fine. Some x86 compilers of the era had bugs and wouldn't compile things right if the optimization flag was used, but fortunately this isn't a problem for 2.11 BSD.

```
What options should be given to the compiler [-O]? <enter>
```

We're now asked what linking options we want. The "relaynews" component of C News needs separate I & D spaces, so we need to pass the "-i" argument.

```
What options, if any, should be given for linking []? -i
```

Accept the default for extra libraries needed:

```
What libraries, in addition to the one(s) picked up
automatically by the compiler, should be used when linking C
News []? <enter>
```

Next we need to tell it what mechanism to use to determine the length of the news batch queue. We want "hdb" here.

```
There is a good chance that you will have to customize the
"queuelen" program.  C News knows about several versions:
...
<snip>
...
Which one is most appropriate [hdb]? <enter>
```

A similar question is asked regarding the system call used to find the amount of free disk space. With the original sources, the build chokes on `statfs` and you have to use `bsd`. My modifications include a fix for this issue:

```
C News often wants to ask how much disk space is available.
...
<snip>
...
Which one is most appropriate [statfs]? <enter>
```

The next question is:

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

To be honest, I'm not sure what this means or what all the implications are. Any input from Usenet veterans would be welcome. I believe it's basically asking if you're planning to compress old news and store it for a while before throwing it away entirely. I think the default of "`no`" is safe here: The "`expire`" component of C News is responsible for throwing away news that's older than whatever timeframe the news administrator has set for news to be kept. Usenet at its peak generated enough traffic to fill up any disk very quickly (it still does, but mostly spam and other non-legitimate traffic), so administrators only carried groups of local interest, and often only kept news for a week, sometimes only long enough so that news that came in over a long weekend didn't expire before anyone could read it. Usenet these days is mostly spam, so you probably don't want to connect to it, and the number of people on the entire internet with a retrocomputing interest in UUCP and netnews is probably less than the number of users that a large site like UC Berkeley would have had back in the day, so the traffic on any non-Usenet UUCP network you might participate in in 2019 is not likely to fill up your 2.11 BSD disk image, let alone the disk on the physical hardware you're emulating it on, any time soon. So just press `<enter>` here.

Next question is:

```
Are you particularly short of disk space [no]?
```

C News's defaults will cause it to *think* you're short of disk space as soon as you install it, and if you were connecting to real Usenet, it would be right, but for the traffic you're likely to get, you can just say no here. Hit `<enter>`.

Now it wants to know if we're running our news system distributed across a network. We are not.

```
Are you running C News on a group of machines hooked together
with NFS, run essentially as a single system with a single
administration, with articles filed on one "server" machine
[no]? <enter>
```

Now it asks us where to put the programs that will receive news from other systems, the default, at `$NEWSBIN/input`, is fine:

```
Where should "rnews" and "cunbatch" be put
[/usr/libexec/news/input]? <enter>
```

Then it asks us where "`inews`", which injects locally posted news into the news system, and "`readnews`", "`checknews`", and "`postnews`" (which together implement a very basic newsreader program), should go. The default of `/usr/bin` is fine here.

```
What directory should these commands go in [/usr/bin]? <enter>
```

You're now done with `quiz`, now it's time to actually build C News.

2. Building

Next, edit the top-level makefile and make sure the `MAKE=` line contains the full path for conf/maker that you entered at the "What is the name of the make to be used?" step of `quiz`:

```
MAKE=/usr/src/cnews/conf/maker
```

Now, edit `include/config.make`, and make the same change. It will probably have the path as `../conf/maker` , which seems like it should work, but the build always dies on me if I don't use an absolute path, so you'll need the full path. Also change the `CFLAGS` and `LINTFLAGS` lines to use a full path to the `include` directory in the C News source tree rather than a relative path. All the other relative paths is this file seem not to cause any trouble.

Eg:

```
CFLAGS=$(MORECFLAGS) -O -DSMALLMEM -I/usr/src/cnews/include
```

`include/config.make` will be rewritten every time you run `quiz` (if you don't Ctrl-C out of it before answering the last question, which prevents it from making any changes), so if you've run `quiz` since the last time you made the above changes, you will need to re-edit or the build will fail.

At this point we're ready to compile. Type:

```
$ make
```

And go get a cup of coffee / grab a bite to eat. The build will take a while.

Once that's done, type:

```
$ make r
```

To run the regression tests.

If you are not using my patches, see my commentary on my patches at the beginning of this document for a list of problems I encountered and patched around that you may also run into trouble with.

If the regression tests pass (the documentation says that failures that don't stop make aren't serious), then run:

```
$ make install
```

If that succeeds, set permissions for `$NEWSBIN/input/newsspool`

```
$ chmod ug+s $NEWSBIN/input/newsspool
```

Then:

```
$ make setup
```

Then, login as `root` (or whoever owns the directory you used for the last question in `quiz`), go to the C News source directory, and run:

```
# make ui
```

And, if you want C News's `readnews`, `checknews`, and `postnews`:

```
# make readpostcheck
```

3. Configuration

Exit your root shell and go back to working as the news administrator. At this point, `README.install` recommends that you edit at least the following files in `$NEWSCTL` with parameters suitable to your system: `batchparms, controlperm, explist, mailname, mailpaths, organization, postdefltgroup, readnews.ctl, sys, whoami`

Starting with `batchparms,` I just commented out all the lines except the `/default/` line. This was used to configure how to send news batches to other systems, depending on things like the speed and cost of the link between the two systems (does the other side have a fast modem? Is it an international call?), and the characteristics of the remote system (if the systems on both ends of the link have fast CPUs, and the link is slow or expensive, aggressive compression algorithms can be used). For now the defaults are probably good.

`controlperm` says who is allowed to send various netnews control messages to this machine (creating groups, deleting groups, etc). Comment out or delete anything regarding the old usenet hierarchies: Modern Usenet is mostly full of spam, and the volume would be too much for a PDP-11 (even an emulated PDP-11 running at faster-than-historical speeds on a modern processor). If enough people are interested, maybe the PiDP-11 community will eventually have its own netnews network running, and we'll have administrators in charge of creating or deleting groups network-wide: this is the file that tells C News which administrators' control messages you trust: if only the "all" line at the

end of the file is uncommented, only groups that you personally and manually set up on your system will be visible. If you have something like this in `controlperms`:

```
usepi                           fred@foobar         n       y
usepi                           fred@foobar         r       n
usepi                           any                 nr      nq
```

Then if the user "`fred`" at the host "`foobar`" posts a "**n**ew group" message to the network that creates a group under the "`usepi`" hierarchy, your system will automatically create that group locally without action from you, and report to you that it has done so (y is for **y**es), but if he posts a "**r**emove group" message for the same hierarchy, it will reject that control message (n is for **n**o), and report it to you. If anyone else tries creating or removing a group in that hierarchy it will reject it **q**uietly (without spamming you with a report). If you had "`nv`" instead of "`nq`", it would **v**erbosely include the full control message in the report. `v` and `q` can also be combined with `y`. Per the default `controlperms` file that gets installed with C News, the best practice seems to have been to accept control messages from trusted remote administrators to create groups (if the admin's address gets spoofed, the group can always be deleted), but to only report group removals to the local admin (so that the local admin can verify that the control message indeed did come from the remote administrator and remove the group manually), and to quietly reject all new group / remove group messages that did not come from a trusted remote admin (so that the local admin isn't bothered by hooligans trying to create havoc by adding or removing groups).

The next file is `explist`. This tells C News when to start auto-deleting netnews messages. Unless you are participating in a very high-volume UUCP network, it is probably safe to comment everything out except the last line beginning with "`all`", and to change the "`7`" in that line to "`never`", as you probably won't run out of disk space too quickly to intervene manually. But for systems that carry traffic on the real-life USENET, expiring articles to preserve disk space was, and is, a major concern.

`mailname` is the name of the system that should appear in "`From`" headers on netnews messages. This is so that people can reply to the author of a netnews message by mail. For now you can just leave it as the hostname of your system, if a PiDP-11 netnews network gets going, it will probably include a bang path from a major site or an internet domain name.

`mailpaths` contains aliases with routes that posts to moderated newsgroups should be sent to. This can probably be left alone for now.

`organization` contains the name of your organization. Since you're a hobbyist in 2020, not a university/company sysadmin in 1989, this can be anything you want, maybe a fictional network provider founded by you (BraseNet), maybe some whimsical secret society (Illuminati: Eureka, KS branch), or an organization from a favorite show/movie/book (Starfleet, Foundation, Galactic Republic).

`postdefltgroup` is the group that posts will go to if the user does not specify a newsgroup. Something like `yourhostname.general` is likely good here.

`readnews.ctl` specifies what groups users are subscribed to by default, and what groups they must subscribe to. If you have no other users but yourself on the system, "`mustsub`" can probably be left

out. If you're letting others log in, it might be good to have a local announcements group that everybody must subscribe to. "`defsub`" can be whatever you want.

`sys` describes what newsgroups your system is willing to receive, and what it feeds to other systems. It will be highly site specific, according to who you have netnews feeds from/to. The format is described by `man/newssys.5` (you can read this with `nroff -ms man/newssys.5 | less`). When we test our ability to send and receive news, later on in this guide, we make appropriate additions to the file to send and receive news in our test newsgroup.

`whoami` defines the news name of the system. This, again, will be the hostname of your system.

At this point, run:

```
$ make cmp
```

If this finishes with "No worrisome differences", the installation `README` then says that we should set up appropriate aliases for news reporting. If you told `quiz` to send reports to your user account (or to root), no action is needed here. If you were planning on using a `sendmail` alias, set that up if you haven't already.

Now, add `conf/crontab` to the `crontab` for your news administrator user account. This sets up a schedule for news system tasks to be run, like processing news that has arrived from neighboring systems, sending news to neighbors (either locally posted, or news from other sites that a given neighbor hasn't seen yet), and various other housekeeping:

```
$ crontab conf/crontab
```

The default `crontab` file is probably fine for now, but, as you build confidence and experience with C News, if you want to change how things are scheduled and aren't too familiar with Unix:

```
$ man 5 crontab
```

describes the file format that `cron`, the traditional name for the system scheduling service on Unix systems, uses. You can write up a schedule ("cron table" → "crontab") according to the format described, and then install it with the `crontab` program. For more information on the `crontab` program (as opposed to the `crontab` file format), see:

```
$ man 1 crontab
```

Next, as root, edit `/etc/rc.local` to add `$NEWSBIN/maint/newsboot` at the end. `/etc/rc` and `/etc/rc.local` are shell scripts that are run at every boot to start system services, run maintenance tasks, etc. By convention, `/etc/rc` contains stuff that is common to every 2.11BSD system, and calls `rc.local` when it finishes, and `rc.local` contains stuff added by the local administrator. `newsboot` does certain boot-time maintenance on the news system. Note that you will have to use whatever value you defined for `$NEWSBIN`, not the literal string "`$NEWSBIN`".

Now, edit /etc/uucp/L.cmds. This tells the uucp system what commands uux is allowed to run on behalf of remote machines. We need to add rnews. When another machine logs in to transfer news, it runs rnews via uux. rnews then takes the news from the remote system as input and deposits it in $NEWSARTS/in.coming. The crontab we just set up then has a schedule item for a job that moves the individual news articles into the appropriate newsgroup directories. We also need to add $NEWSBIN/input to the PATH= directive in L.cmds so that uux can find rnews. The lines you add will look something like this:

```
PATH=/bin:/usr/bin:/usr/ucb:/usr/new:/usr/libexec/news/input
rnews,error
```

(The "error" after rnews specifies that an e-mail should be sent back to the news user on the sending system if rnews exits with an error).

The next step is to install the man pages and documentation. The installation README says this is too system-specific for the makefiles to attempt it, and I haven't yet figured this out on 2.11 BSD. Most of the stuff in the man and doc subdirectories in the C News source directory can be read with:

```
$ nroff -ms document | less
```

Where "document" is the name of the document you want to read. Some documents, however, appear to end up with words missing when processed with the nroff -ms macros. If anyone can figure out how this all needs to be done, please let me know.

4. Testing

The next thing to do is create a local newsgroup to make sure that the relevant parts of C News are working. As the news administrator:

```
$ cnewsdo addgroup insert.group.name.here
```

The next order of business is to set up $NEWSCTL/sys to tell C News that we want to receive this group from other systems, and that we want to feed it to the host Pi. Find the line that starts with "ME:" , and edit it to read:

```
ME:insert.group.name.here
```

Now, create a new line that reads:

```
pi-hostname:insert.group.name.here/all:f:
```

Where "pi-hostname" is the UUCP name of your Raspberry Pi (this will be the same as the hostname for the Pi if you followed my UUCP setup guide). All other lines should be empty or commented out. Save and close the file.

Next, as the news user, create a directory for the host Pi in `$NEWSARTS/out.going`:

```
$ mkdir $NEWSARTS/out.going/pi-hostname
```

This needs to be done any time a new neighbor system is added to `$NEWSCTL/sys`. The directory is not created automatically, and if it is not created, the news system will fail to transfer news to that neighbor (it will receive news from that system without issues, though).

Then, as any user:

```
$ postnews insert.group.name.here
```

It's optional to provide a newsgroup name to `postnews` on the command line. If you don't, it will ask you first thing when the program starts up. It will then ask for a subject, and, once that is provided, it will drop you into the editor defined by the `VISUAL` and `EDITOR` environment variables in your shell. If neither of these variables are defined, `postnews` will drop you into `ed`, the original Unix text editor. If you can handle a line editor, you can just write your message with `ed`, the man page is available to familiarize yourself with the exact command set used. If you're of my generation or later, and don't know what a line editor is, this probably isn't the best time to learn how to use one, so you will probably want to set these variables in your current session with:

```
$ VARIABLE=value
$ export VARIABLE
```

You will then want to edit your shell profile (.profile in your home directory if you're using the default shell, `/bin/sh`) to define these variables at login for your user account in the future. Just set the `VISUAL` and/or `EDITOR` variable to whatever text editor you have been using so far.

Postnews will start the editor with a file that looks something like this:

```
Newsgroups: <The group name you provided will be here>
Subject: <The subject you provided will be here>

REPLACE THIS LINE WITH YOUR TEXT (leave the preceding blank line
alone)
```

As the text in the file instructs, replace the instruction line with the text of your message. Don't delete or type in the blank line between the header and the message text, or C News will emit angry messages about invalid header fields. Often when this happens the message seems to post anyway. When done, save your message, and `postnews` should place it into `$NEWSARTS/in.coming`. The next time the news `crontab` runs a "`newsrun`" job, the message will be placed into the directory under `$NEWSARTS` corresponding to the group. If the group is called "`this.is.a.test`", the first message posted to that group will be found at `$NEWSARTS/this/is/a/test/1`. The message should also appear in the `history` file, the `log` file, and not in the `errlog` file (all three will be under `$NEWSCTL`).

If you exit the editor without saving, postnews will display the message:

```
This posting does not appear to have been edited properly.
Abandon it [y] ?
```

Answering "y" here will back out of posting the message, answering "n" will dump you back into the editor. I had initially assumed that mangled messages (stuff like no blank line after the header) would also generate this message, but they don't appear to, at least for the low-severity deliberate mangling I've tried.

Now try running:

```
$ readnews -p -n insert.group.name.here
```

The message should now be printed out.

5. Setting up INN on the host Pi

At this point the installation `README` tells us to get a feed from somebody and see if batches are arriving and being processed. Part of my hope in writing this guide is to establish a UUCP network among PiDP-11 owners, so hopefully eventually this step will be just a matter of coordinating with somebody who is already part of the network. But such a network does not yet exist, and it will probably also require some work to develop a procedure to connect two PiDP-11instances (as far as I know, `simh` only accepts inbound telnet connections, as opposed to being able to establish an outgoing connection, and, furthermore, since traffic on our hypothetical hobbyist UUCP network would be going over the internet, it would be best to encrypt it in transit). So for now, we'll have to connect C News on 2.11 BSD with a modern news server on the host Pi to make sure C News is working, and inter-site news traffic will have to be between Pi's. Hopefully eventually we'll be able to cut modern systems out of the loop, though.

First of all, when UUCP contact is made with the 2 BSD system, it will need to be able to invoke `rnews` on the Pi to be able to transfer news. The news server we will install is configured by default to allow `rmail` and `rnews` (and only `rmail` and `rnews`) to be run by `uux`, but the news package we will install keeps `rnews` in a place that is not on the list of directories Taylor UUCP searches by default. So we will need to edit `/etc/uucp/sys` , and add the following line somewhere at the top (before the definitions of any specific systems):

```
command-path /usr/sbin /usr/lib/news/bin
```

Save the file, then, install our news server, inn2:

```
$ sudo apt-get install inn2
```

Now, as root, edit `/etc/news/inn.conf`. Look for the "`pathhost`" line, which should be near the top of the file. Put the hostname of your Pi here. The INN documentation recommends using a fully-qualified domain name, but since we won't be connecting to actual Usenet, I think just the bare

hostname should do. You might add something to the "domain" line if your LAN has a domain. "organization" can just be whatever you used for $NEWSCTL/organization in the previous section. "complaints" doesn't need to be set for now, but if we do get a hobbyist UUCP network going, it may be good to set it to some local address that will reach you. "server" should be "localhost". Any time you edit inn.conf, you need to run:

```
$ sudo service inn2 restart
```

Now edit /etc/news/newsfeeds . It has a format that looks a lot like the $NEWSCFG/sys file on the 2BSD end, but it doesn't work quite the same way. In particular, there's a big comment before the "ME" line telling you how the format for this line is different from the rest of the file (and thus very different from $NEWSCTL/sys in C News). You will probably want to read newsfeeds(5) at some point so that you understand how to edit this file to change what systems you accept news from and which groups you accept from them. For now, however, we will write two entries to allow us to send and receive our test newsgroup.

First, edit the "ME" line to read as follows:

```
ME:!*/!local::
```

This will reject anything from other sites having a distribution list of "local", and accept anything else. Next, look for the comment beginning with "A UUCP feed". Under this place the following entry:

```
2bsdhostname\
          :*,!junk,!control,!control.*\
          :Tf,Wnb:
```

The backslashes allow us to write this entry across several lines for readability. 2bsdhostname should be the hostname of the guest 2.11 BSD system. Save and close the file.

Now edit /etc/news/incoming.conf . The existing entry for "peer ME" should do for the host Pi, now add an entry for the 2.11 BSD system:

```
peer 2bsdhostname{
     hostname  "2bsdhostname"
}
```

The next file to edit is /etc/news/expire.ctl . This file tells INN when to delete articles. As on the 2.11 BSD end, we probably don't want to expire anything for the time being. Find the line that says:

```
*:A:1:15:never
```

And change the "15" to "never".

Now edit /etc/news/moderators. This should just contain a single entry (for now):

```
*:yourusername@pihostname
```

The last file to edit is `/etc/news/send-uucp.cf` . This controls the INN component that sends out UUCP batches. Add a line like the following:

```
2bsdhostname        none      1048576
```

This gives the hostname that the batch is to be sent to, the compression method to be used, and the maximum size of a batch (1MB). I tried using the "`compress`" compression method, and found that UUCP on the 2BSD end was choking on the batches, so I'm using "`none`" for now.

The last file to edit is `/etc/cron.d/inn2` . This schedules various periodic jobs for INN. Uncomment the line that says:

```
22 *    * * *   news    send-uucp.pl
```

This will schedule any outgoing news that has collected to be handed off to UUCP at 22 past the hour, every hour. UUCP will then deliver then news at its next scheduled call to the remote system. If `/etc/cron.d/uucp` calls the 2BSD system more than once per hour, you may want to change this line to send batches more frequently.

Now, run:

```
$ sudo ctlinnd newgroup insert.group.name.here
$ sudo ctlinnd begin 2bsdhostname
```

using the name of the test group you set up on 2.11 BSD earlier for the newgroup command.

5. Testing news transfers

Now that INN is up and running, we can see if news is properly being transferred between the two systems. First, we will need to set up a newsreader on the Pi. I've had the best success with `tin`.

```
$ sudo apt-get install tin
```

`tin`, by default, expects the article spool and overview directory tree to both be rooted at the root of the entire news spool tree, `/var/spool/news` . INN keeps the article spool and overview tree in two separate directories under `/var/spool/news` . Therefore, we will need to make some changes to tin's defaults. Open `/etc/tin/tin.defaults`, and look for the section "reading from a local spool". Under this section there will be two variables, "`spooldir`" and "`overviewdir`". Define these so they point at INN's article and overview directories:

```
spooldir=/var/spool/news/articles
overviewdir=/var/spool/news/overview
```

Now, go to the 2 BSD system and post another test article to your test newsgroup with postnews. If all goes well, it will appear in `$NEWSARTS/in.coming`, then, as various cron jobs do their thing, be

transferred to the directory for the appropriate newsgroup, be transferred into `$NEWSARTS/out.going`, and then a `uux` job to run `rnews` on the Pi will be scheduled. The next time the Pi calls, the news will be transferred, and it should appear in `/var/spool/news/incoming` on the Pi, and eventually INN will move it into the appropriate newsgroup. Given that, by default, many of the cron jobs involved on either system run once per hour, it will take about that long for the process to complete. So wait a while, then, on the Pi, run:

```
$ tin
```

Initially, you won't be presented with anything to read, because you aren't subscribed to anything. There are two things you can do: either hit "`y`" to show all groups available instead of just subscribed ones, then use the arrow keys to select your test group and hit "`s`" (lower case) to subscribe to the selected group, or else hit "`S`" to provide a group name to subscribe to (this can actually subscribe to multiple groups, if you use wildcards in the group name you specify. See `wildmat (3)`, for information on the wildcard matching syntax). Once you're subscribed to the group, select it and press `<enter>` to view the articles in the group. Articles will be grouped as threads by subject (with only threads with unread messages shown by default). Selecting a thread will read the first article in the thread, you can view subsequent articles by hitting "`n`". A thread for the test article you posted should appear. Once you've pulled the article up, you can hit "`f`" to "**f**ollow-up" the article (nowadays we'd say "reply to the article", but "follow-up" is the mnemonic for the key to press). Write up the reply, save and exit in the text editor when done, and then hit "`p`" to post. If the hostname that appears in the "`from`" address is not a dotted domain name, `tin` will complain. If this happens, you can hit "e" to re-edit the article, and add "`.uucp`" to the hostname. If all goes well, the article should make its way back to the 2.11 BSD system in an hour or so. As long as it does we've got everything set up!

The one issue remaining is that the "`readnews`" program that comes with C News leaves a lot to be desired, compared to practically any other newsreader available. Fortunately, 2.11 BSD includes a newsreader program called "`rn`" that combines the functionality of `readnews / postnews / checknews` . The program is only included as source, however, and as it was written for machines with much more memory, getting it to compile and run on a PDP-11 takes some setup (though it is fairly straightforward). I will document this in either a future version of this document or a new howto.