Setting up UUCP on 2.11 BSD

1. Setting up 2.11 BSD UUCP

This is a guide for setting up UUCP communication between the host Pi and a guest 2.11 BSD system running on a PiDP-11. It is also generally valid for setting up UUCP communication between any 2.11 BSD guest system running on the simh emulator and any host system, and this first section can probably be used with a real PDP-11 (if the owner is not already familiar with UUCP on 2.11 BSD), but I am primarily writing for the benefit of the PiDP-11 community.

Fortunately, this in general goes a bit easier than setting up UUCP on V7 Unix, because we don't need to recompile the kernel, and we already have a skeleton setup. On the other hand, there are some differences from the setup for V7 Unix, and most of the resources I was able to find covered either older systems (V7 Unix) or modern systems (Taylor UUCP on modern *nix).

First, make sure that /etc/hostname contains the hostname you want the system to use. It should be all-lower-case, or you might run into trouble with certain software later. (I had some issues with the interaction between C-news and INN due to a mixed-case hostname for my Pi). Also change the "`hostname=...`" line in `/etc/netstart`.

Then, look at `/etc/uucp/USERFILE` . You should see something like:

```
,     /usr/spool/uucppublic
```

This will allow any machine, logging in as any user, to access files in the /usr/spool/uucppublic directory. This is a good default. If you want to allow anything other than this, you can add entries in the format:

```
user,machine   /directory1 /directory2 /directory3
```

Which gives the user/machine pair named access to the directories named.

Now edit `/etc/uucp/L.sys` . It will contain something like:

```
test Any TCP uucp test.your.domain login: dummy assword: yahoo!
test Polled TCP uucp test.your.domain
```

Replace this with something of the form:

```
pi-hostname Never TCP telnet pi-hostname
```

The first field tells UUCP that another UUCP system called "`pi-hostname`" exists, and that it might receive calls from that system. The second, "`Never`", tells UUCP that it should never attempt to call that system (all interaction with "`pi-hostname`" will occur as a result of that system calling us). The rest of the line should be specified, but is a dummy value. It tells UUCP how to contact the remote system if we ever make a call to it (which we won't, because this line specifies never to call the remote system).

We don’t make calls to the host because simh, to my knowledge, does not presently contain any method for an emulated system to establish outbound telnet connections over its serial ports. Eventually, we’ll probably have some kind of shim (probably involving tcpser or something like it) that lets us get around this, but for now, we’ll just have the host Pi make all the calls.

Now, use `vipw` to edit `/etc/passwd` to create a user for the Pi to log in as for UUCP:

```
$ vipw
```

And, once you’ve launched vipw add a line something like:

```
pi-uucp:*:101:101::0:0:UUCP login for host pi:/usr/spool/uucppublic:/usr/sbin/uucico
```

Note this should all be typed on one line. It will wrap around in an 80 column terminal. That’s fine, just as long as there’s not a linefeed anywhere.

If `vipw` determines that you’ve created an invalid passwd file entry, it will prompt you to reedit the file.

Now, set a password for the new account:

```
$ passwd pi-uucp
```

UUCP is now set up on 2.11 BSD.

2. Setting up Taylor UUCP on the host Pi

This will be mostly the same as in the *V7 UUCP Installation Guide*, except that the restriction of the host Pi giving itself a 7 character name when communicating with the guest system doesn’t apply. As root, open up /`etc/uucp/sys` in your text editor of choice. Leave the defaults at the top of the file in place.

Add the following block of text:

```
system 211name
time any n
max-retries n
call-login pi-login
call-password pi-passwd
port 211-dz
address localhost
chat "" \d\d\r ogin: \d\L word: \P

alternate 211name-2
port 211-dz-pipe
```

The argument to "system" should be the name you gave to the 2.11 BSD system when setting up UUCP. Call-login should use the same user name you created in /etc/passwd on 2.11 BSD, and call-password should be the same password you set up with passwd in the last step of the 2.11 BSD instructions. For "time", the "any" keyword means that Taylor UUCP can call 2.11 BSD at any time, and the number "n" is the number of minutes you want it to wait between attempts to call 2.11 BSD. It can be omitted if you don't want to have a minimum wait time. For "max retries", n is the number of times UUCP will try calling 2.11 BSD before giving up until the next day.

The chat script is made up of pairs of strings to expect, and the response to send
to those strings. So first, we expect nothing (that is, we immediately send the
"response"), wait two seconds with \d\d, and then send a carriage return, \r.
When "login" on 2.11 BSD sees the carriage return, it will reprint the login prompt.
We then wait for the string "ogin:" (in case we miss the first letter of "login:"),
wait one second, and send our login name with \L. We then wait for "word:", and send
our password with \P.

The alternate name is just a name to identify an alternate means of calling the
2.11 BSD system.

Now edit /etc/uucp/port, and add the following two entries. 211-dz is the method we use to call the 2.11 BSD system on the first attempt, which is just to have uucico itself try to contact the 2.11 BSD system over port 4001, 211-dz-pipe is the alternate method we try if that doesn't work, which is to invoke telnet and use that to try to connect.

```
port 211-dz
type tcp
service 4001

port 211-dz-pipe
type pipe
command /usr/bin/telnet -8 4001
```

Now, edit the simh boot.ini file for your 2.11 BSD system. Halfway down or so, you should see something like this:

```
set dz enabled
set dz lines=8
set dz 7b
;attach -am dz 4000
attach dz 4000
```

I had trouble getting good UUCP communication with the 2.11 BSD system using these settings. Change it to something like this:

```
set dz enabled
set dz lines=8
set dz 8b
attach -a -m dz 4000
attach dz -a -m line=7,4001;notelnet
```

If your 2.11 BSD system is running, bring it down and wait for the `sim>` prompt, then type:

```
sim> do boot.ini
```

This will load the new configuration.

Now, find a convenient file to try to send to the v7 system. Send it to the public directory on the v7 system:

```
$ uucp filename 211name\!~/
```

"`211name`" of course, should be replaced with the name you used for the v7 system. Note that the ! has to be escaped with a backslash, since ! has a meaning on many modern shells.

This queues up the file for sending. Back in the day, our system would generally make scheduled callouts to the other system (or receive scheduled callins from the other system). The uucp program would queue work to be sent during the scheduled call. Since we're just transferring between two systems that we completely control (for now), setting up a scheduled call would be overkill, and also would force us to wait for the call. Instead, we'll just tell UUCP to call immediately.

```
$ sudo uucico -s 211name
```

If all goes well, once uucico finishes executing, we should be able to find our file in `/usr/spool/uucp/public` on the 2.11 BSD system. On the 2.11 BSD system, type:

```
# ls /usr/spool/uucppublic
```

which should display:

```
filename
```

Now send it back. On the 2.11 BSD system, type:

```
# uucp /usr/spool/uucp/public/foo.c piname!~/
```

Note that since the `/bin/sh` on 2.11 BSD doesn't interpret ! in any special way, we don't need to escape it here. However, if you are using csh or tcsh, you will have to escape it.

And then on the pi, type:

```
$ sudo uucico -s 211name
```

Once uucico is done, we should be able to list `/var/spool/uucppublic`, and see our file.

```
$ ls /var/spool/uucp/public
```

If you want to have the Pi poll the 2BSD system automatically on a schedule (for example, if you'll be transferring mail or news), instead of running uucico yourself every time you have something to send, either add the system to /etc/uucp/Poll:

```
schedule 211name hours
```

where "hours" is a list of two-digit hours of the day (between 00 and 23) specifying at what times calls to the BSD system should be made,

or edit /etc/cron.d/uucp to poll the system directly:

```
0     *       *       *       *       uucp    /usr/sbin/uucico -
x3 -f -s 211name
```

which calls the system every hour, on the hour. You could use something like

```
*/15     *       *       *       *       uucp
/usr/sbin/uucico -x3 -f -s 211name
```

to run uucico every 15 minutes.

The Poll file is called by a line in the uucp crontab, every hour. It's not mentioned in the Taylor BSD documentation (that I could find), so I'm not sure of all the syntax for it. Thus, for my system, I have scheduled the call directly in the crontab, as the crontab format is well documented. The documentation can be accessed from the shell by typing:

```
$ man 5 crontab
```

The basic gist of the format is that the first five fields are the minute, hour, day of month, month, and day of week, the sixth field is the user to run as, and the rest of the line is the command to run. A cron line is run any time the current time matches all three of the minute, hour, and month fields, and either one of the day of month or day of week fields. Names can be used for months and days of the week. A * in a field will match any time. Ranges, designated with dash, can be used (e.g, 1-5 in the day of week field will match any day that isn't a weekend), a field can contain a list separated with commas, and /<number> runs at every step of so many minutes/hours/etc. For example */10 in the minutes field will run at every minute divisible by ten, 9-17/2 in the hours field will run at 9 AM, and then every other hour until 5 PM.

That's it! We've set up UUCP.