The MacTerminal MINI−HOWTO

Table of Contents

The MacTerminal MINI−HOWTO

Robert Kiesling

v1.4, 9 November 1997

This mini HOWTO describes the 1,002nd use for a dead Macintosh (grin): how to configure the Mac for use as a Linux terminal. Configurations using getty and the terminal program kermit are described, as well as using kermit peer−to−peer networking between between Linux and a Macintosh. This document may be reproduced freely, in whole or in part, provided that any usage conforms to the general copyright notice of the HOWTO series of the Linux Documentation Project. See the file COPYRIGHT for details. Send all complaints, suggestions, errata, and any miscellany to kiesling@terracom.net, so I can keep this document as complete and up to date as possible.

---

1. Introduction.

This mini−HOWTO should give you some Insanely Great ideas for how to make your Macintosh work with Linux. Unfortunately, I have been very busy, and so I haven't been able to include even half of what I wanted to include, like using MacTCP and Open Transport to connect to your Linux box via a PPP line. That will need to wait for future versions.

This mini−HOWTO doesn't cover networking with LocalTalk and AppleTalk, either. I might explore these avenues if there's enough interest in, say, printing to a LaserWriter printer from Linux. Otherwise, it seems to me that such applications, being more trouble than they're worth (not to mention pricey), are beyond the scope of this document.

I don't plan to cover MkLinux in this document, either. It's more than adequately documented elsewhere.

So if you have ideas for this document, drop me a line at the e−mail above. Both systems embody a lot of the beginner's mindset as well as technical prowess, and in my opinion they don't talk to each other nearly enough.

---

2. Setting up a serial link.

To set up a serial link between a Mac and a Linux machine, you will need, on the Linux side, either a DB9 Female−to−DB25 Male serial cable or a DB25 Female−to−DB25 Male serial cable, depending on your serial port. On the Macintosh side, you will need a DIN9−to−DB25 Male high−speed modem cable.

Make sure that the cable is labeled a "high speed" cable, because some older Macintosh cables are configured with their handshaking lines tied high, which makes them useless for high−speed serial connections.

You will also need a null modem adapter, available at Comp USA, Radio Shack, and similar outlets, and a DB25 Female−to−DB25 Female serial gender changer to connect the two serial cables.

I have heard that Mac printer cables are really null modem cables in disguise, but I can't confirm this. Some of them are DIN9−to−DIN9 anyway, and wiring one into a serial link would be more trouble than it's worth.

If this sounds like Greek to you, read the Serial−HOWTO for details of RS−232 cable configurations and data transmission protocols.

Before connecting the Mac and the Linux machines, you should determine that you have a working serial port on both machines, either by connecting a modem and dialing out to another computer with `minicom` (Linux), ZTerm (Mac), `kermit` (either), or the communications program of your choice.

The latest version of minicom is available from sunsite.unc.edu/pub/Linux/apps/serialcomm/dialout and mirror sites.

ZTerm is a complete, easy to use comm program. Unfortunately, it's shareware. A current version is available from mac.archive.umich.edu and outlets like it.

The `kermit` program has been ported to every computer and operating system in existence. The archives are located at ftp.columbia.edu/kermit.

You should strongly consider using `kermit` on both machines at this stage at least, because 1) it's free (although it's not covered by the Free Software Foundation's General Public License); and 2) it's a lot less confusing to have `kermit` on both machines than two completely different communications programs.

If you have another way to determine that the serial ports of the two machines are operational, feel free to use that. The point is to ensure that both machines have working serial ports.

Making the actual serial connection should be easy, given the directions above. In case it isn't, the connection looks like this:

```
Linux PC    DB9- or DB25-  Null     Gender   DIN9-to-        MacBox
---------   to-DB25 male   Modem    Changer  DB25 Mac       --------
|       |   serial cable. |     |  |     |   Serial Cable  |      |
|       |-----------------|     |--|     |-----------------|      |
```

```
 |       |                  |     |  |     |                 |      |
 ---------                   Adapter                          --------
```

3. Client−server connection with `kermit`.

This is the most transient of all the configurations described here. It requires the least amount of system configuration, although in operation, it is the more difficult to use of the systems described here.

In brief, you start `kermit` on both the Linux machine and the Mac, and place one of them in server mode. It doesn't matter which machine is the client ant which is the server, because this is a peer−to−peer connection. However, the Linux `kermit` can take advantage of Linux's superior scripting abilities, so it seems logical (to me at least) to designate the Linux−side `kermit` as the server, because this is the more readily automated task.

You should ensure that `kermit` is installed correctly on both the Mac and the Linux PC. Follow the instructions in the respective kermit distributions. On the Linux machine type `kermit` at the shell prompt to start it. You may need root permissions in order to set the port and baud rate.

`kermit`, the recent POSIX versions for Unices, supports baud rates up to 115 Kbps. The more recent Macintosh versions support serial port speeds up to 57.6 Kbps. This should be more than sufficient for any dumb tty−type application, but if you need a higher−speed connection, you're s.o.l, as far as `kermit` and serial lines are concerned. However, `kermit` provides facilities for communication over a TCP/IP link, but I haven't been able to test it. See the alternative in the following sections. Just remember, especially on the Mac side, to use a different port for `kermit` serial connections than your TCP/IP connections, because Mac `kermit` will rudely hose a serial port that is already in use.

With that in mind, your `.kermrc` file would contain something like this:

```
echo Executing site initialization file /usr/local/bin/ckermit.local.ini....
set prompt Chanel3 >
set line /dev/ttyS0
set baud 38400
set send packet−length 2000
set receive packet−length 2000
set block 3
set file type binary
```

Then, in your `~/.kermrc` file, you would have a line like

```
take /usr/local/bin/ckermit.local.ini
```

On the Macintosh side, set the same communication parameters for bps, stop bits, parity, and word length. Some older versions of Mac Kermit do not support 2k packets, so you might need to set a smaller packet size. Howerver, `kermit` sets the communication packet length based on the receive packet−length setting, so you need to set a shorter packet size on the Linux end, too.

To actually communicate over the link, you need to enter server mode on either the Mac or Linux side. It doesn't matter which. See the `kermit` docs for details of server mode.

3.1 Macintosh resources.

This is one of the very few `kermit` applications where setting a `text` file type for transfers is useful. This is because Macintosh files have two parts: the **data fork** and the **resource fork**. The data fork corresponds to what we in the Linux world think of as a file: it's the actual data. The resource fork contains bitmaps for the icons, keymaps, font specifications, and the like. If you transfer a file from Linux to the Mac, the file won't be recognized as a text file by the Mac, if you use binary mode.

When transferring binary files between the two systems, you should use the Macintosh `.hqx` BinHex format, which is a 7−bit encoding of an 8−bit data file. Mac utilities like BinHexer or StuffIt will covert the file to its binary form.

If you have a text file which inadvertently ends up as a data−only file on the Mac, it's likely that it won't even appear in an Open dialog list box. What you need to do is open the file with ResEdit, which is available from `mac.archive.umich.edu`. ResEdit will tell you that the file you're opening has no resource fork and then asks if you would like to add one. You should answer "Yes" to this question. You can then edit the file's Type and Creator by selecting the Open Special option of the File menu. All Macintosh text files are type `TEXT`, so replace the question marks in the Text box with that. The Creator code depends on your text editor or word processor. Each one is unique, incidentally, and is how the Mac identifies different apps. The Creator code for GNU Emacs on the Mac is `EMAC`, for example. If in doubt what the creator code of your text editor or word processor is, use `ttxt`, which is the creator code for TeachText (which is the Mac equivalent of `EDLIN.EXE`.) Then your real word processor or text editor can translate the file from TeachText to its native type.

There are many other neato things which TeachText can do, so it's worthwhile to keep it permanently on your Mac. The book *Voodoo Mac,* by Kay Yarborough Nelson, is a good source of tried−and−true Macintosh tricks that use ResEdit, TeachText, the Finder, and other overlooked programs.

---

4. Logging in via `kermit`.

Configuring Linux to use the Mac as a `login:` terminal is even easier. `kermit` is ideal for this purpose, because it is one of the few free communication programs which provides credible VT100/120/220 emulation.

Essentially, what you want to do is start `kermit` on the Macintosh side as in the previous section, but rather than issue server commands, you enter `connect` mode. This is the normal terminal emulation mode that most people use, anyway.

On the Linux side, the serial line must be configured with a `getty` on it to start a `login:` shell. To do this, you need to tell `init` that the serial line has a terminal on it. In your `/etc/inittab` file you will need a line something like this:

```
T1:23:respawn:/sbin/getty −L ttyS0 9600 vt100
```

Be sure to substitute the appropriate serial device for `/dev/ttyS0` and the correct baud rate for `9600` in the command line above.

This command tells `getty` to start `login` (the `−L` switch) on the terminal display, and, when the login times out, to re−start (`respawn`) the login program until someone logs in. If no device is connected to the serial line, or if the connection is defective, you may see a message on the system console like: `/dev/ttyS0 respawning too fast: disabling for 5 minutes`. If this happens, you can return things to normal by (as root) `kill`ing the `getty` process, or using the `init q` command. Both of them have the effect of re−spawning the `getty` processe(s). If everything is in order, you should see the Linux banner and login prompt on the Mac's `kermit` window. That's all there is to it.

Also, if you use something besides vanilla `getty`, like `getty_ps`, the command above will look somewhat different. The important thing to remember is that everything to the right of `/sbin/getty` is an argument for `getty` itself; not `init`. You should look at the manual pages for `getty`, `init`,and `inittab` if you have questions concerning the setup of `init` and `getty`.

The Serial HOWTO provides helpful details on how to configure `/etc/inittab` for `getty_ps`, if that's what your system uses.

To transfer files back and forth between the Macintosh and the Linux machine, you can (via the Mac's Kermit) issue the `kermit -x` command to start the Linux `kermit` in server mode. You can then use the normal file transfer commands to send files across the serial line. It's useful to set a prompt in your `~/.kermrc` with a line like

```
set prompt Linux-kermit >
```

Otherwise, remembering which machine you're on can quickly become confusing.

4.1 Other Mac terminal programs.

This method should work equally well for any other Mac terminal program. If you have ZTerm, you can use `rz` and `sz` on the Linux machine to transfer files via the ZModem protocol. If Microphone Lite came bundled with your fax modem, that works equally well, albeit without `kermit's` superior scripting and configuration facilities.

---

5. Conclusion.

If you have questions about any of this material, or suggestions for future directions of Mac−Linux serial−line connectivity, don't hesitate to drop me a line at `kiesling@terracom.net`.
