The Linux Serial HOWTO

by Greg Hankins, greg.hankins@cc.gatech.edu

v1.8.1, 9 October 1995

This document describes how to set up serial communications devices on a Linux box.

Contents

1 Introduction

This is the Linux Serial HOWTO. All about how to set up modems and terminals under Linux, some serial tips, and troubleshooting.

1.1 Copyright

In short, we wish to promote dissemination of this information through as many channels as possible. However, we do wish to retain copyright on the HOWTO documents, and would *like* to be notified of any plans to redistribute the HOWTOs.

If you have questions, please contact Greg Hankins, the Linux HOWTO coordinator, at `gregh@sunsite.unc.edu` via email, or at +1 404 853 9989.

1.2 Other sources of information

- man pages for: `agetty(8)`, `getty(1m)`, `gettydefs(5)`, `init(1)`, `login(1)`, `mgetty(8)`, `setserial(8)`
- Your modem manual
- UUCP HOWTO: for information on setting up UUCP
- Printing HOWTO: for setting up a serial printer
- NET-2 HOWTO: all about networking, including SLIP, CSLIP, PLIP and PPP
- BUPS HOWTO: setting up UPS boxen connected to your serial port
- Term HOWTO: everything you wanted to know about the `term` program
- USENET newsgroups:

```
comp.os.linux.advocacy            Benefits of Linux compared to other operating systems.
comp.os.linux.announce            Announcements important to the Linux community.
comp.os.linux.answers             FAQs, How-To's, READMEs, etc. about Linux.
comp.os.linux.development.apps    Writing Linux applications, porting to Linux.
comp.os.linux.development.system  Linux kernels, device drivers, modules.
comp.os.linux.hardware            Hardware compatibility with the Linux operating system.
comp.os.linux.misc                Linux-specific topics not covered by other groups.
comp.os.linux.networking          Networking and communications under Linux.
comp.os.linux.setup               Linux installation and system administration.
comp.os.linux.x                   Linux X Window System servers, clients, libs and fonts.
```

- the Linux serial mailing list. To join, send email to `majordomo@vger.rutgers.edu`, with "`subscribe linux-serial`" in the message body. If you send "`help`" in the message body, you get a help message.

1.3 New versions of this document

New versions of the Serial-HOWTO will be placed on *sunsite.unc.edu* `<ftp://sunsite.unc.edu:/pub/Linux/docs/HOWTO/Serial-HOWTO>` and mirror sites. There are other formats, such as a PostScript and `dvi` version in the `other-formats` directory. The *Serial HOWTO* `<http://sunsite.unc.edu/mdw/HOWTO/Serial-HOWTO.html>` is also available for WWW clients such as `mosaic`. It will also be posted regularly to `comp.os.linux.answers`.

1.4 Feedback

Please send me any questions, comments, suggestions, or additional material. I'm always eager to hear about what you think about the HOWTO. I'm also always on the lookout for improvements! Tell me exactly what you don't understand, or what could be clearer. You can reach me at `greg.hankins@cc.gatech.edu` via email. I can also be reached at:

```
Greg Hankins
College of Computing
801 Atlantic Drive
Atlanta, GA 30332-0280
```

via snail mail, and at *my home page* `<http://www.cc.gatech.edu/staff/h/Greg.Hankins/>` via the WWW.

Please include the version number of the Serial HOWTO when writing, this is version 1.8.1.

1.5 Disclaimer

Your mileage may vary. The answers given may not work for all systems and all setup combinations.

2 Supported serial hardware

Linux is known to work with the following serial hardware.

- standard PC serial boards (COM1 - COM4), to which external serial devices (modems, serial mice, etc...) can be connected
- standard PC internal modems (COM1 - COM4)
- Quickpath Systems Port-Folio 550e (allows IRQs of 3, 4, 5, 9, 10, 11, 12, and 15)

2.1 Multiport serial boards (with 16450/16550A UARTs)

- AST FourPort and clones (4 port)
- Accent Async-4 (4 port)
- Arnet Multiport-8 (8 port)
- Bell Technologies HUB6 (6 port)
- Boca BB-1004 (4 port), BB-1008 (8 port), BB-2016 (16 port)
- Boca IOAT66 (6 port)
- Boca 2by4 (4S/2P)

- Computone ValuePort V4-ISA (AST FourPort compatible)
- PC-COMM (4 port)
- STB-4COM (4 port)
- Twincom ACI/550
- Usenet Serial Board II (4 port)

In general, Linux will support any serial board which uses a 8250, 16450, 16550, 16550A (or compatible) UART, or an internal modem which emulates one of the above UARTs.

Special note on the BB-1004 and BB-1008, they do not support DCD and RI lines, and thus are not usable for dialin modems. They will work fine for all other purposes.

2.2 Intelligent multiport serial boards

- Comtrol RocketPort (36Mhz ASIC - 4, 8, 16 or 32 port) (contact `info@comtrol.com` or *Comtrol's Home Page* `<http://www.comtrol.com>`. Driver location: `tsx-11.mit.edu/pub/linux/packages/comtrol`)
- Computone IntelliPort II (16Mhz 80186 - 4, 8, or 16 port) Computone IntelliPort II EXpandable (20Mhz 80186 - modular 16 - 64 port) (pre-ALPHA driver, contact Michael H. Warfield, `mhw@wittsend.atl.ga.us`)
- Cyclades Cyclom 8Y (8 port), and 16Y (16 port) (Cirrus Logic CD-1400 RISC UARTs) (contact `cyclades@netcom.com`)
- DigiBoard PC/Xe (12.5MHz 80186 processor - 2, 4, 8, or 16 port), and PC/Xi (12.5MHz 80186 processor - 8, or 16 port) (contact Troy De Jongh, `troyd@digibd.com`. Driver location: `ftp.digibd.com:/digiline/drivers/linux`)
- Digiboard COM/Xi (10MHz 80188 processor - 4 or 8 port)
  (pre-ALPHA driver contact Simon Park, `si@wimpol.demon.co.uk`)
- Hayes ESP8 (8 port)
  (pre-ALPHA driver, contact Dennis Boylan, `dennis@lan.com`)
- Omega COMM-8 (8 port)
  (contact Vance Petree, `vpetree@infi.net`)
- Specialix SIO - (modular, 4 - 32 port)
  (ALPHA driver, contact Simon Allen, `simonallen@cix.compulink.co.uk`)
- Stallion EasyIO-4 (4 port), EasyIO-8 (8 port), and Stallion EasyConnection (modular, 8 - 32 port) (Cirrus Logic CD-1400 RISC UARTs), Stallion (8MHz 80186 processor - 8 or 16 port), Brumby (10 MHz 80186 processor - 4, 8 or 16 port), ONboard (16MHz 80186 processor - 4, 8, 12, 16 or 32 port), EasyConnection 8/64 (25MHz 80186 processor - modular, 8 - 64 port) (contact Greg Ungerer, `gerg@stallion.oz.au` or *Stallion's Home Page* `<http://www.stallion.com>`. Driver Location: `ftp.stallion.com:/drivers/ata5/Linux`)

Drivers for the Cyclades, DigiBoard, and Specialix boards can be found on `sunsite.unc.edu:/pub/Linux/kernel/patches/serial` and mirror sites.

3 What are the names of the serial ports?

There are the 4 serial devices corresponding to COM1 - COM4:

```
/dev/cua0, /dev/ttyS0 (COM1) address 0x3f8 IRQ 4
/dev/cua1, /dev/ttyS1 (COM2) address 0x2f8 IRQ 3
/dev/cua2, /dev/ttyS2 (COM3) address 0x3e8 IRQ 4
/dev/cua3, /dev/ttyS3 (COM4) address 0x2e8 IRQ 3
```

The `/dev/ttyS`N devices are for incoming connections and `/dev/cua`N devices for outgoing connections. N is the serial port number. In this document, I refer to COM1 as `ttyS0`, COM2 as `ttyS1`, COM3 as `ttyS2`, and COM4 as `ttyS3`. If I am referring to a specific device in `/dev`, I will always prepend `/dev` to avoid confusing you.

On some installations, two extra devices will be created, `/dev/modem` for your modem and `/dev/mouse` for your mouse. Both of these are symbolic links to the appropriate `/dev/cua`N device which you specified during the installation (unless you have a bus mouse, then `/dev/mouse` will point to the bus mouse device).

There has been some discussion on the merits of `/dev/mouse` and `/dev/modem`. I *strongly* discourage the use of these links. In particular, if you are planning on using your modem for dialin you will run into problems because the lock files will not work correctly if you use `/dev/modem`. Use them if you like, but *be sure they point to the right device*.

3.1 Major and minor device numbers of serial devices in /dev

```
/dev/ttyS0 major 4, minor 64    /dev/cua0 major 5, minor 64
/dev/ttyS1 major 4, minor 65    /dev/cua1 major 5, minor 65
/dev/ttyS2 major 4, minor 66    /dev/cua2 major 5, minor 66
/dev/ttyS3 major 4, minor 67    /dev/cua3 major 5, minor 67
```

Note that all distributions should come with these devices already made correctly.

3.1.1 Creating devices in /dev

If you don't have a device, you will have to create it with the `mknod` command.

Example, suppose you needed to create devices for `ttyS0`:

```
linux# mknod -m 666 /dev/cua0 c 5 64
linux# mknod -m 666 /dev/ttyS0 c 4 64
```

You can also get the `MAKEDEV` script, available on the usual FTP sites. This simplifies the making of devices. For example, if you needed to make the devices for `ttyS0` you would type:

```
linux# cd /dev
linux# MAKEDEV ttyS0
```

This handles the devices creation for the incoming and outgoing devices.

3.1.2 Notes for multiport boards

The devices your multiport board uses depends on what kind of board you have. These are listed in detail in the `rc.serial` which comes with the `setserial` program. You will probably need to create these devices. Either use the `mknod` command, or get the `MAKEDEV` script. Devices for multiport boards are made by adding "64 + the port number". So, if you wanted to create devices for `ttyS17`, you would type:

```
linux# mknod -m 666 /dev/cua17 c 5 81
linux# mknod -m 666 /dev/ttyS17 c 4 81
```

Note that "64 + 17 = 81". Using the `MAKEDEV` script, you would type:

```
linux# cd /dev
linux# MAKEDEV ttyS17
```

4 What is getty?

`getty` is a program that handles the login process when you log onto a Unix box. There are 3 versions that are commonly used with Linux: `agetty`, `getty_ps` and `mgetty`.

4.1 About getty_ps

This version of `getty` was written by Paul Sutcliffe Jr., `paul@devon.lns.pa.us`. Kris Gleason, `gleasokr@boulder.colorado.edu` currently maintains it. 2.0.7e is the latest version, and supersedes any older versions. Most distributions come with the `getty_ps` package installed as `/sbin/getty`.

The `getty_ps` package contains two getties. `getty` is used for console, and terminal devices - and `uugetty` which is used for modems. I use this version of `getty`, so I will focus on the `getty_ps` package in this HOWTO.

4.2 About mgetty

`mgetty` is a version of `getty` written by Gert Doering `gert@greenie.muc.de`. In addition to allowing logins, `mgetty` also provides class 2 FAX support through `sendfax`, which accompanies `mgetty`. `mgetty+sendfax 0.22` is the latest version of this package. The `mgetty` documentation is quite good, and does not need supplementing. Please refer to it for installation instructions.

4.3 About agetty

`agetty` is the third variation of `getty`. It was original written by W.Z. Venema, `wietse@wzv.win.tue.nl`. It's a simple implementation of `getty`.

5 What is setserial?

`setserial` is a program which allows you to look at and change various attributes of a serial device, including its port address, its interrupt, and other serial port options. It was initially written Rick Sladkey, and was heavily modified by Ted T'so `tytso@mit.edu`, who also maintains it. The newest version is 2.10, and can be found on the Linux FTP sites. You can find out what version you have by running `setserial` with no arguments.

When your Linux system boots, only `ttyS{0-3}` are configured, using the default IRQs of 4 and 3. So, if you have any other serial ports provided by other boards or if `ttyS{0-3}` have a non-standard IRQ, you *must* use this program in order to configure those serial ports. For the full listing of options, consult the man page.

6 How do I dial out with my modem?

6.1 Hardware requirements

First, make sure you have the right cable. Your modem requires a straight through cable, with no pins crossed. Any computer store should have these. Make sure you get the correct gender. If you are using the DB25 serial port, it will always be the male DB25. Do not confuse it with the parallel port, which is the female DB25. Hook up your modem to one of your serial ports. Consult your modem manual on how to do this if you need help.

6.1.1 Notes on internal modems

For an internal modem, you will not need a cable. An internal modem does not need a serial port, it has one built in. All you need to do is configure it to use an interrupt that is not being used, and configure the port I/O address. Consult your modem manual if you get stuck. Also, see section 9 if you need help on choosing interrupts or addresses.

Due to a bit of stupidity on IBM's part, you may encounter problems if you want your internal modem to be on `ttyS3`. If Linux does not detect your internal modem on `ttyS3`, you can use `setserial` and the modem will work fine. Internal modems on `ttyS{0-2}` should not have any problems being detected.

6.2 Talking to your modem

Use `kermit` or some other simple comm program to test the setup, before you go jumping into complex comm programs. (For legal reasons, `kermit` is not distributed with commercial distributions. You can find the latest version of **kermit** on **sunsite.unc.edu:/pub/Linux/apps/comm** and mirror sites.) For example, say your modem was on **ttyS3**, and it could handle 38400 bps. You would do the following:

```
linux# kermit
C-Kermit 5A(188), 23 Nov 92, POSIX
Type ? or HELP for help
C-Kermit>set line /dev/cua3
C-Kermit>set speed 38400
/dev/cua3, 38400 bps
C-Kermit>c
Connecting to /dev/cua3, speed 38400.
The escape character is Ctrl-\ (ASCII 28, FS)
Type the escape character followed by C to get back,
or followed by ? to see other options.
AT
OK
<ctrl>-\-C
(Back at linux)
C-Kermit>quit
linux#
```

If your modem responds to `AT` commands, you can assume your modem is working correctly on the Linux side. Try calling another modem. If you don't like `kermit`, try one of the more advanced comm programs. Check out section 11 about comm programs if you need some pointers.

When you dial out with your modem, set the speed to the highest bps rate that your modem supports. Since there is no speed named 57600 or 115200 bps, you must use the `setserial` program to set your serial port to a higher speed. See section 10 for how to do this. Then, set the speed to 38400 bps in your comm program.

6.3 Dial out modem configuration

For dial out use only, you can configure your modem however you want. If you intend to use your modem for dialin, you *must* configure your modem at the same speed that you intend to run `getty` at. So, if you want to run `getty` at 38400 bps, set your speed to 38400 bps when you configure your modem. This is done to prevent speed mismatches between your computer and modem.

I like to see result codes, so I set `Q0` - result codes are reported. To set this on my modem, I would have to precede the register name with an `AT` command. Using `kermit` or some comm program, connect to your modem and type the following: `ATQ0`. If your modem says `OK` back to you, then the register is set. Do this for each register you want to set.

I also like to see what I'm typing, so I set `E1` - command echo on. If your modem has data compression capabilities, you probably want to enable them. Consult your modem manual for more help, and a full listing of options. If your modem supports a stored profile, be sure to write the configuration to the modem (often done with `AT&W`, but varies between modem manufacturers) if not you will have to set the registers every time you turn on, or reset your modem.

6.4 Hardware flow control

If your modem supports hardware flow control (RTS/CTS), I highly recommend you use it. This is particularly important for modems that support data compression. First, you have to enable RTS/CTS flow control on the serial port itself. This is best done on startup, like in `/etc/rc.d/rc.local` or `/etc/rc.d/rc.serial`. Make sure that these files are being run from the main `rc.M` file! You need to do the following for each serial port you want to enable hardware flow control on:

```
stty crtscts < /dev/cuaN
```

You must also enable RTS/CTS flow control on your modem. Consult your modem manual on how to do this, as it varies between modem manufacturers. Be sure to save your modem configuration if your modem supports stored profiles.

7 How do I dial in and out with my modem using getty_ps?

Get your modem to dial out correctly. If you haven't read section 6 go *read it now*! It contains *very* important setup information.

7.1 Dial in and out modem configuration

For dialin and dialout use, you *have* to set up your modem a certain way (again, using `AT` commands on your modem):

```
E1      command echo ON
Q0      result codes are reported
V1      verbose ON
S0=0    never answer (uugetty handles this with the WAITFOR option)
```

If you don't set these correctly, your `INIT` string in your config file may fail, hosing the whole process. But, more on config files below...

```
&C1    DCD is on after connect only
&S0    DSR is always on
       DTR on/off resets modem (depends on manufacturer - RTFM)
```

These affect what your modem does when calls start and end.

If your modem does not support a stored profile, you can set these through the `INIT` string in your config file. See below. Some modems come with DIP switches that affect register settings. Be sure these are set correctly, too.

I have started a collection of modem setups for different types of modems. So far, I only have a few of them, if you would like to send me your working configuration, please do so! If you would like me to send you one of the configurations, just mail me and ask. I'm not listing them here due to space concerns. I have setups for Supra, Telebit T1600, USR Courier and Sportster, and Zoom 14.4/28.8 modems.

7.2 Installing getty_ps

By default, getty_ps will be configured to be Linux FSSTND (**F**ile**S**ystem **ST**a**ND**ard) compliant, which means that the binaries will be in /sbin, and the config files will be named /etc/conf.{uu}getty.ttySN. This is not apparent from the documentation! It will also expect lock files to go in /var/lock. Make sure you have the /var/lock directory.

If you don't want FSSTND compliance, binaries will go in /etc, config files will go in /etc/default/{uu}getty.ttySN, and lock file will go in /usr/spool/uucp. I recommend doing things this way if you are using UUCP, because Taylor UUCP will have problems if you move the lock files to where it isn't looking for them.

getty_ps also uses syslogd to log messages. See the man pages for syslogd(1) and syslog.conf(5) for setting up syslogd, if you don't have it running already. Messages are logged with priority LOG_AUTH, errors use LOG_ERR, and debugging uses LOG_DEBUG. If you don't want to use syslogd you can edit tune.h in the getty_ps source files to use a log file for messages instead, namely /var/adm/getty.log by default.

When you have decided if you want FSSTND, and syslog, edit tune.h and the Makefile in the getty_ps source directory to reflect you decisions. Now, install according to the instructions.

From this point on, all references to getty will refer to getty_ps. References to uugetty will refer to the uugetty that comes with the getty_ps package.

7.3 Setting up uugetty

For dialing into, and out from your modem, we want to use uugetty. uugetty does important lock file checking. Update /etc/gettydefs to include entries for modems (note that the entries point to each other, these are not for fixed speed):

```
# Modem entries
38400# B38400 CS8 # B38400 SANE -ISTRIP HUPCL #@S @L @B login: #19200
19200# B19200 CS8 # B19200 SANE -ISTRIP HUPCL #@S @L @B login: #9600
9600# B9600 CS8 # B9600 SANE -ISTRIP HUPCL #@S @L @B login: #2400
2400# B2400 CS8 # B2400 SANE -ISTRIP HUPCL #@S @L @B login: #1200
1200# B1200 CS8 # B1200 SANE -ISTRIP HUPCL #@S @L @B login: #300
300# B300 CS8 # B300 SANE -ISTRIP HUPCL #@S @L @B login: #38400
```

If you have a 9600 bps or faster modem, with data compression, you can lock your serial port speed and let the modem handle the transitions to other bps rates. Then, instead of the step down series of lines listed above, /etc/gettydefs only needs to contain one line for the modem:

```
# 38400 fixed speed
F38400# B38400 CS8 # B38400 SANE -ISTRIP HUPCL #@S @L @B login: #F38400
# 19200 fixed speed
F19200# B19200 CS8 # B19200 SANE -ISTRIP HUPCL #@S @L @B login: #F19200
# 9600 fixed speed
F9600# B9600 CS8 # B9600 SANE -ISTRIP HUPCL #@S @L @B login: #F9600
```

If you have your modem set up to do RTS/CTS hardware flow control, you can add CRTSCTS to the entries:

```
# 38400 fixed speed with hardware flow control
F38400# B38400 CS8 CRTSCTS # B38400 SANE -ISTRIP HUPCL CRTSCTS #@S @L @B login: #F38400
# 19200 fixed speed with hardware flow control
F19200# B19200 CS8 CRTSCTS # B19200 SANE -ISTRIP HUPCL CRTSCTS #@S @L @B login: #F19200
# 9600 fixed speed with hardware flow control
F9600# B9600 CS8 CRTSCTS # B9600 SANE -ISTRIP HUPCL CRTSCTS #@S @L @B login: #F9600
```

If you want, you can make `uugetty` print interesting things in the login banner. In my examples, I have the system name, the serial line, and the current bps rate. You can add other things:

```
@B    The current (evaluated at the time the @B is seen) bps rate.
@D    The current date, in MM/DD/YY.
@L    The serial line to which getty is attached.
@S    The system name.
@T    The current time, in HH:MM:SS (24-hour).
@U    The number of currently signed-on users.  This is  a
      count of the number of entries in the /etc/utmp file
      that have a non-null ut_name field.
@V    The value of VERSION, as given in the defaults file.
To display a single '@' character, use either '\@' or '@@'.
```

Next, make sure that you have an outgoing and incoming device for the serial port your modem is on. If you have your modem on `ttyS3` you will need the `/dev/cua3`, and `/dev/ttyS3` devices. If you don't have the correct devices, see section 3.1.1 on how to create devices, and create the devices.

7.4 Customizing uugetty

There are lots of parameters you can tweak for each port you have. These are implemented in separate config files for each port. The file `/etc/conf.uugetty` will be used by *all* instances of `uugetty`, and `/etc/conf.uugetty.ttyS`*N* will only be used by that one port. Sample default config files can be found with the `getty_ps` source files, which come with most Linux distributions. Due to space concerns, they are not listed here. Note that if you are using older versions of `getty` (older than 2.0.7e), or aren't using FSSTND, then the default file will be `/etc/default/uugetty.ttyS`*N*. My `/etc/conf.uugetty.ttyS3` looks like this:

```
# sample uugetty configuration file for a Hayes compatible modem to allow
# incoming modem connections
#
# alternate lock file to check... if this lock file exists, then uugetty is
# restarted so that the modem is re-initialized
ALTLOCK=cua3
ALTLINE=cua3
# line to initialize
INITLINE=cua3
```

```
# timeout to disconnect if idle...
TIMEOUT=60
# modem initialization string...
# format: <expect> <send> ... (chat sequence)
INIT="" AT\r OK\r\n
WAITFOR=RING
CONNECT="" ATA\r CONNECT\s\A
# this line sets the time to delay before sending the login banner
DELAY=1
#DEBUG=010
```

Add the following line to your `/etc/inittab`, so that `uugetty` is run on your serial port (substituting in the correct information for your environment - port, speed, and default terminal type):

```
S3:456:respawn:/sbin/uugetty ttyS3 F38400 vt100
```

Restart `init`:

```
linux# init q
```

For the speed parameter in your `inittab`, you want to use the highest bps rate that your modem supports. Since there is no speed named 57600 or 115200, you must use the `setserial` program to set your serial port to a higher speed. See section 10 for doing this. Then, use 38400 bps in your `inittab`.

Now Linux will be watching your serial port for connections. Dial in from another site and login to you Linux system. Rejoice.

`uugetty` has a lot more options, see the man page for `getty(1m)` for a full description. Among other things there is a scheduling feature, and a ringback feature. RTFM :-).

7.5 US Robotics Notes

To get my USR Courier modem to reset correctly when DTR drops, I had to set `&D2` and `S13=1`.

7.6 Supra Notes

Supra modems treat DCD differently than other modems. If you are using a Supra, you must set `&C0` and *not* `&C1`. You must also set `&D2` to handle DTR correctly.

8 How do I set up a terminal connected to my PC?

8.1 Hardware requirements

Make sure you have the right kind of cable. A null modem cable bought at a computer store will do it. But it must be a *null modem* cable! Make sure you are using your serial port, the male DB25 or the DB9, and not your parallel port.

For a DB25 connector, you need a minimum of:

```
TxD   Transmit Data         2 - 3       RxD   Receive Data
RxD   Receive Data          3 - 2       TxD   Transmit Data
SG    Signal Ground         7 - 7       SG    Signal Ground
```

If you want to have hardware handshaking signals, you must have a full null modem cable:

```
TxD   Transmit Data         2 - 3       RxD   Receive Data
RxD   Receive Data          3 - 2       TxD   Transmit Data
RTS   Request To Send       4 - 5       CTS   Clear To Send
CTS   Clear To Send         5 - 4       RTS   Request To Send
DSR   Data Set Ready        6 - 20      DTR   Data Terminal Ready
SG    Signal Ground         7 - 7       SG    Signal Ground
DCD   Carrier Detect        8 - 20      DTR   Data Terminal Ready
DTR   Data Terminal Ready  20 - 6       DSR   Data Set Ready
DTR   Data Terminal Ready  20 - 8       DCD   Carrier Detect
```

If you have a DB9 connector on your serial port, try the following:

```
                          DB9   DB25
RxD   Receive Data          2 - 2       TxD   Transmit Data
TxD   Transmit Data         3 - 3       RxD   Receive Data
SG    Signal Ground         5 - 7       SG    Signal Ground
```

Alternatively, a full DB9-DB25 null modem cable:

```
                          DB9   DB25
DCD   Carrier Detect        1 - 20      DTR   Data Terminal Ready
RxD   Receive Data          2 - 2       TxD   Transmit Data
TxD   Transmit Data         3 - 3       RxD   Receive Data
DTR   Data Terminal Ready   4 - 6       DSR   Data Set Ready
DTR   Data Terminal Ready   4 - 8       DCD   Carrier Detect
SG    Signal Ground         5 - 7       SG    Signal Ground
DSR   Data Set Ready        6 - 20      DTR   Data Terminal Ready
RTS   Request To Send       7 - 5       CTS   Clear To Send
CTS   Clear To Send         8 - 4       RTS   Request To Send
(RI Ring Indicator          9 not needed)
```

If you are not using a full null modem cable, you might have to do the following trick: on your computer side of the connector, connect RTS and CTS together, and also connect DSR, DCD and DTR together. This way, when the computer wants a certain handshaking signal, it will get it (from itself).

Now that you have the right kind of cable, connect your terminal to your computer. If you can, tell you terminal to ignore modem control signals. Try using 9600 bps, 8 data bits, 1 stop bit, no parity bits for the terminal's setup.

8.2 Setting up getty

Install `getty_ps` as described in section 7.2. Add an entry for `getty` to use for your terminal in `/etc/gettydefs`:

```
# 38400 bps Dumb Terminal entry
DT38400# B38400 CS8 CLOCAL # B38400 SANE -ISTRIP CLOCAL #@S @L login: #DT38400
# 19200 bps Dumb Terminal entry
DT19200# B19200 CS8 CLOCAL # B19200 SANE -ISTRIP CLOCAL #@S @L login: #DT19200
# 9600 bps Dumb Terminal entry
DT9600# B9600 CS8 CLOCAL # B9600 SANE -ISTRIP CLOCAL #@S @L login: #DT9600
```

If you want, you can make `getty` print interesting things in the login banner. In my examples, I have the system name and the serial line printed. You can add other things:

```
@B    The current (evaluated at the time the @B is seen) bps rate.
@D    The current date, in MM/DD/YY.
@L    The serial line to which getty is attached.
@S    The system name.
@T    The current time, in HH:MM:SS (24-hour).
@U    The number of currently signed-on users.  This is  a
      count of the number of entries in the /etc/utmp file
      that have a non-null ut_name field.
@V    The value of VERSION, as given in the defaults file.
To display a single '@' character, use either '\@' or '@@'.
```

Edit your `/etc/inittab` file to run `getty` on the serial port (substituting in the correct information for your environment - port, speed, and default terminal type):

```
S1:456:respawn:/sbin/getty ttyS1 DT9600 vt100
```

Restart `init`:

```
linux# init q
```

At this point, you should see a login prompt on your terminal. You may have to hit return to get the terminal's attention. Rejoice.

8.3 Notes on setting up a PC as a terminal

Many people set up other PCs as terminals connected to Linux boxen. For example, old 8088 or 286 PCs are perfect for this purpose. All you need is a DOS boot disk containing a version of DOS suitable for your terminal-PC, and a communications program for your terminal-PC to run. `kermit` works very well for this purpose. You can find precompiled versions of `kermit` for nearly every OS in existence at `watsun.cc.columbia.edu:/pub/ftp/kermit`. Other popular DOS comm programs such as `telix` and `procomm` will work equally well. Be sure to input correct serial port information into your terminal-PC's communications setup.

9 Can I use more than 2 serial devices?

You don't need to read this section, unless you want to use 3 or more serial devices... (assuming you don't have a multiport board).

Providing you have another spare serial port, yes, you can.

The number of serial ports you can use is limited by the number of interrupts (IRQ) and port I/O addresses we have to use. This is not a Linux limitation, but a limitation of the PC bus. Each serial devices must be assigned it's own interrupt and address. A serial device can be a serial port, an internal modem, or a multiport serial board.

Multiport serial boards are specially designed to have multiple serial ports that share the same IRQ for all serial ports on the board. Linux gets data from them by using a different I/O address for each port on the card.

9.1 Choosing serial device interrupts

Your PC will normally come with `ttyS0` and `ttyS2` at IRQ 4, and `ttyS1` and `ttyS3` at IRQ 3. To use more than 2 serial devices, you will have to give up an interrupt to use. A good choice is to reassign an interrupt from your parallel port. Your PC normally comes with IRQ 5 and IRQ 7 set up as interrupts for your parallel ports, but few people use 2 parallel ports. You can reassign one of the interrupts to a serial device, and still happily use a parallel port. You will need the `setserial` program to do this. In addition, you have to play with the jumpers on your boards, check the docs for your board. Set the jumpers to the IRQ you want for each port.

You will need to set things up so that there is one, and only one interrupt for each serial device. Here is how I set mine up in `/etc/rc.d/rc.local` - you should do it upon startup somewhere:

```
/etc/setserial /dev/cua0 irq 3          # my serial mouse
/etc/setserial /dev/cua1 irq 4          # my Wyse dumb terminal
/etc/setserial /dev/cua2 irq 5          # my Zoom modem
/etc/setserial /dev/cua3 irq 9          # my USR modem
```

Standard IRQ assignments:

```
IRQ  0    Timer channel 0
IRQ  1    Keyboard
IRQ  2    Cascade for controller 2
IRQ  3    Serial port 2
IRQ  4    Serial port 1
IRQ  5    Parallel port 2
IRQ  6    Floppy diskette
IRQ  7    Parallel port 1
IRQ  8    Real-time clock
IRQ  9    Redirected to IRQ2
IRQ 10    not assigned
IRQ 11    not assigned
```

```
IRQ 12    not assigned
IRQ 13    Math coprocessor
IRQ 14    Hard disk controller
IRQ 15    not assigned
```

There is really no Right Thing to do when choosing interrupts. Just make sure it isn't being used by the motherboard, or your other cards. 2, 3, 4, 5, or 7 is a good choice. "not assigned" means that currently nothing standard uses these IRQs. Also note that IRQ 2 is the same as IRQ 9. You can call it either 2 or 9, the serial driver is very understanding.

If you have a serial card with a 16-bit bus connector, you can also use IRQ 10, 11, 12 or 15.

Just make sure you don't use IRQ 0, 1, 6, 8, 13 or 14! These are used by your mother board. You will make her very unhappy by taking her IRQs.

9.2 Setting serial device addresses

Next, you must set the port address. Check the manual on your board for the jumper settings. Like interrupts, there can only be one serial device at each address. Your ports will usually come configured as follows:

```
ttyS0 address 0x3f8
ttyS1 address 0x2f8
ttyS2 address 0x3e8
ttyS3 address 0x2e8
```

Choose which address you want each serial device to have and set the jumpers accordingly. I have my modem on `ttyS3`, my mouse on `ttyS0`, and my terminal on `ttyS2`.

When you reboot, Linux should see your serial ports at the address you set them. The IRQ Linux sees may not correspond to the IRQ you set with the jumpers. Don't worry about this. Linux does not do any IRQ detection when it boots, because IRQ detection is dicy and can be fooled. Use `setserial` to tell Linux what IRQ the port is using.

10 How do I set up my serial ports for higher speeds? What speed should I use with my modem?

This section should help you figure out what speed to use when using your modem with a communications program, or with a `getty` program.

- If you have something slower than a 9600 bps (V.32) modem, set your speed to the highest speed your modem supports. For example 300, 1200, or 2400 bps.

- If you have a 9600 bps (V.32) modem, with V.42bis data compression. use 38400 as your speed. V.42bis compression has a *theoretical* rate of 4:1, thus "4 * 9600 = 38400".

- If you have a 14400 bps (V.32bis) modem, with V.42bis data compression, use `setserial`, with the `spd_hi` flag to configure your serial port to use 57600 bps (4 * 14400 = 57600). Use the `spd_vhi` flag if you have a 28800 (V.FC or V.34) modem (4 * 28800 = 115200).

  Then, use 38400 as the speed in your comm program, or `inittab`. This is now the high speed you have set. There is no speed named 57600 or 115200 (although support was added in 1.1.65 and will be used soon). Make sure you have 16550A UARTs :-).

Put your modifications into `/etc/rc.d/rc.serial` or `/etc/rc.d/rc.local` so that they are done at startup. In my `/etc/rc.d/rc.local`, I set `ttyS3` to 115200 bps by doing:

```
/sbin/setserial /dev/cua3 spd_vhi
```

11 Communications programs and utilities

Once you get everything working, you may want to check out these more advanced programs, all are available on the usual FTP sites, if they didn't come with your distribution.

- `ecu` - a communications program
- `minicom` - `telix`-like comm program
- `procomm` - procommish comm program with zmodem
- `seyon` - X based comm program
- `xc` - xcomm communications package

These programs offer more features than just `kermit` alone, including telephone directories, auto-dialing and so on.

- Another useful program is `term`. `term` multiplexes many connections over one serial line. It is somewhat similar to SLIP, and offers some SLIP functionality. These include `rlogin`, `telnet`, `ftp`, `finger`, `rdate`, `xmosaic` and `tredir`. `tredir` is a special program which lets you redirect remote TCP/IP ports to your local machine. This allows for remote NNTP, and SMTP access. The good thing about `term` is that is runs entirely in user space, meaning it requires no kernel support, or sysadmin support (like SLIP does).
- `screen` is another multi-session program. This one behaves like the virtual consoles.
- `callback` is a program that will have your modem call you back immediately from where you just called.
- `mgetty+fax` handles FAX stuff, and provides an alternate `getty`
- `ZyXEL` is a control program for ZyXEL U-1496 modems. It handles dialin, dialout, dial back security, FAXing, and voice mailbox functions.

- Other things can be found on `sunsite.unc.edu:/pub/Linux/system/Serial` and `sunsite.unc.edu:/pub/Linux/apps/comm` or one of the many mirrors. These are the directories where all the serial type things are kept.

12 Serial Tips

Here are some serial tips you might find helpful...

12.1 kermit and zmodem

To use zmodem with **kermit**, add the following to your **.kermrc**:

```
define rz !rz < /dev/cua3 > /dev/cua3
define sz !sz \%0 > /dev/cua3 < /dev/cua3
```

Be sure to put in the correct port your modem is on. Then, to use it, just type **rz** or **sz <filename>** at the **kermit** prompt.

12.2 Setting terminal types automagically

To set your terminal type automagically when you log in, add the terminal type to the entry in **/etc/inittab**. If I have a vt100 terminal on **ttyS1**, I would add "vt100" to the **getty** command:

```
S1:456:respawn:/sbin/getty ttyS1 DT9600 vt100
```

You can also get **tset** from **sunsite.unc.edu:/pub/Linux/system/Terminal-management** or a mirror site. See the docs that come with **tset** to learn how to use it. **tset** can establish terminal characteristics when you log in, and doesn't depend on any defaults.

12.3 Color ls on serial connections

If **ls** is screwing up your terminal emulation with the color feature, turn it off. **ls --color**, and **ls --colour** all use the color feature. Some installations have **ls** set to use color by default. Check **/etc/profile** and **/etc/csh.cshrc** for **ls** aliases. You can also alias **ls** to **ls --no-color**, if you don't want to change the system defaults.

12.4 Printing to a printer connected to a terminal

There is a program called **vtprint** that will do this, written by Garrett D'Amore **garrett@sdsu.edu**.
It is available from **ftp.sdsu.edu:/pub/vtprint**, and also from **http://www.sdsu.edu/~garrett/**. The following is from the **README** file that comes with the program:

vtprint is a program that allows users to print from a remote UNIX host to a printer attached to their local terminal or emulator, which makes it great for printing files at home, etc. (It only does text files, though.)

12.5 Can Linux configure the serial devices automagically?

Yes. To get Linux to detect and set up the serial devices automatically on startup, add the line:

```
/sbin/setserial /dev/cuaN auto_irq skip_test autoconfig
```

to your `/etc/rc.d/rc.local` or `/etc/rc.d/rc.serial` file. Do this for every serial port you want to auto configure. Be sure to give a device name that really does exist on your machine.

12.5.1 Notes for multiport boards

For board addresses, and IRQs, look at the `rc.serial` that comes with the `setserial` program. It has a lot of detail on multiport boards, including I/O addresses and device names.

13 Linux FTP sites

```
sunsite.unc.edu [152.2.22.81]:/pub/Linux        (NC, USA)
tsx-11.mit.edu [18.172.1.2]:/pub/linux          (MA, USA)
nic.funet.fi [128.214.6.100]:/pub/OS/Linux      (Finland, Europe)
```

`sunsite.unc.edu` is the official Linux FTP site, and has many mirrors. Please use a mirror site if at all possible to save `sunsite` some traffic.

`sunsite` mirrors (`sunsite.unc.edu:/pub/Linux/MIRRORS`):

```
* CONTINENT
- COUNTRY
   CITY... FTP Site  Directory
-------------------------------------------------------------------------
* Africa
- None so far
* Asia
- Thailand
   Bangkok... ftp.nectec.or.th  /pub/mirrors/linux/
- Hong Kong
    ... ftp.cs.cuhk.hk  /pub/Linux/
- Republic of Singapore
   Singapore... ftp.nus.sg  /pub/unix/Linux/
- Japan
   Unknown... ftp.spin.ad.jp  /pub/linux/sunsite.unc.edu/
* Australia
```

```
  Adelaide...  smug.student.adelaide.edu.au  /pub/sunsite.linux/
  Brisbane...  ftp.dstc.edu.au  /pub/linux/
* Europe
- Austria
  Graz...  ftp.tu-graz.ac.at  /pub/Linux/
- Czech Republic
  Brno...  ftp.fi.muni.cz  /pub/UNIX/linux/
  Prague...  pub.vse.cz  /pub/386-unix/linux/
- France
  Angers...  ftp.univ-angers.fr  /pub/linux/
  Nancy...  ftp.loria.fr  /pub/linux/sunsite/
- Germany (Deutschland)
  Aachen...  ftp.dfv.rwth-aachen.de  /pub/linux/sunsite/
  Dortmund...  ftp.germany.eu.net  /pub/os/Linux/Mirror.SunSITE/
  Dresden...  ftp.tu-dresden.de  /pub/Linux/sunsite/
  Erlangen...  ftp.uni-erlangen.de  /pub/Linux/MIRROR.sunsite/
  Mannheim...  ftp.ba-mannheim.de  /pub/linux/mirror.sunsite/
  Paderborn...  ftp.uni-paderborn.de  /pub/Mirrors/sunsite.unc.edu/
  Rostock...  ftp.uni-rostock.de  /Linux/sunsite/
  Stuttgart...  ftp.rus.uni-stuttgart.de  /pub/unix/systems/linux/MIRROR.sunsite/
  Tuebingen...  ftp.uni-tuebingen.de  /pub/linux/Mirror.sunsite/
  Ulm...  ftp.rz.uni-ulm.de  /pub/mirrors/linux/sunsite/
  Unknown...  ftp.gwdg.de  /pub/linux/mirrors/sunsite/
- Hungary
  Budapest...  ftp.kfki.hu  /pub/linux/
- Italy
  Pisa...  cnuce-arch.cnr.it  /pub/Linux/
- Switzerland
  Zurich...  ftp.switch.ch  /mirror/linux/
- Turkey (Turkiye)
  Ankara...  ftp.metu.edu.tr  /pub/linux/sunsite/
- United Kingdom
  Coventry...  ftp.maths.warwick.ac.uk  /mirrors/linux/sunsite.unc-mirror/
  London...  src.doc.ic.ac.uk  /packages/linux/sunsite.unc-mirror/
  Mildenhall...  ftp.dungeon.com  /pub/linux/sunsite-mirror/
* North America
- United States
  Atlanta, GA...  ftp.cc.gatech.edu  /pub/linux/
  Chapel Hill, NC...  sunsite.unc.edu  /pub/Linux/
  Fayetteville, AR...  ftp.engr.uark.edu  /pub/linux/sunsite/
  Flagstaff, AZ...  ftp.infomagic.com  /pub/mirrors/linux/sunsite/
  Midwest...  ftp.wit.com  /systems/unix/linux/
  Mt. Pleasant, MI...  ftp.cps.cmich.edu  /pub/linux/sunsite/
  Rochester, NY...  ftp.rge.com  /pub/linux/sunsite/
  Salt Lake City, UT...  ftp.pht.com  /mirrors/linux/sunsite/
  Urbana, IL...  mrcnext.cso.uiuc.edu  /pub/linux/
  Unknown...  ftp.linux.org  /pub/mirrors/sunsite/
  Unknown...  ftp.orst.edu  /pub/mirrors/sunsite.unc.edu/linux/
```

```
   Unknown...  ftp.iquest.com  /pub/linux/sunsite/
   Unknown...  ftp.yggdrasil.com  mirrors/sunsite/
* South America
- Chile
  ftp.inf.utfsm.cl              /pub/Linux
* Unknown
- If you know where these sites are, please mail ewt@sunsite.unc.edu.
    ...  ftp.linux.org  /pub/mirrors/sunsite/
    ...  ftp.gwdg.de  /pub/linux/mirrors/sunsite/
    ...  ftp.orst.edu  /pub/mirrors/sunsite.unc.edu/linux/
    ...  ftp.iquest.com  /pub/linux/sunsite/
    ...  ftp.spin.ad.jp  /pub/linux/sunsite.unc.edu/
    ...  ftp.yggdrasil.com  mirrors/sunsite/
```

These FTP sites support anonymous FTP, which means login as `ftp`, and password as your email address (ie `logname@yourhost.yourdomain`).

14 One step further...

This section is not required reading, but may give you some further insight into Unix, and the world of telecommunications.

14.1 What are lock files?

Lock file are simply a file saying that a particular device is in use. They are kept in `/usr/spool/uucp`, or `/var/lock`. Linux lock files are named `LCK..`*name*, where *name* is either a device name, or a UUCP site name. Certain processes create these locks so that they can have exclusive access to devices, for instance if you dial out on your modem, a lock will appear telling other processes that someone is using the modem already. Locks mainly contain the PID of the process that has locked the device. Most programs look at the lock, and try to determine if that lock is still valid by checking the process table for the process that has locked the device. If the lock is found to be valid, the program (should) exit. If not, some programs remove the stale lock, and use the device, creating their own lock in the process. Other programs just exit and tell you that the device is in use.

14.2 "baud" vs. "bps"

"baud" and "bps" are perhaps one of the most misused terms in the computing/telecom field. Many people use these terms interchangeably, when in fact they are not!

baud

The baud rate is a measure of how many times per second the signal sent by a modem (**mo**dulator-**dem**odulator) changes. For example, a baud rate of 1200 implies one signal change every 833 microseconds. Common baud rates are 50, 75, 110, 300, 600, 1200, and 2400. Most high speed modems run at

2400 baud. Because of the bandwidth limitations on voice-grade phone lines, baud rates greater than 2400 are harder to achieve, and only work under very pristine phone line quality. “baud” is named after Emile Baudot, the inventor of the asynchronous telegraph printer.

bps

The bps rate is a measure of how many bits per second are transmitted. Common bps rates are 50, 75, 110, 300, 1200, 2400, 9600, ... 115200. With modems using V.42bis compression (4:1 compression), *theoretical* bps rates are possible up to 115200. This is what most people mean when they misuse the word “baud”.

So, if high speed modems are running at 2400 baud, how can they send 14400 bps? The modems achieve a bps > baud rate by encoding a number of bits per baud. Thus, when 2 or more bits are encoded per baud, the bps rate exceeds the baud rate. If your modem connects at 14400 bps, it's going to be sending 6 bits per baud.

How did this confusion start? Well, back when today's low speed modems were yesterday's high speed modems, the bps rate actually did equal the baud rate. One bit would be encoded per baud. People would use bps and baud interchangeably, because they were the same number. For example, a 300 bps modem also had a baud rate of 300. This all changed when faster modems came around, and the bit rate exceeded the baud rate.

14.3 What are UARTs? How do they affect performance?

UARTs (**U**niversal **A**syncronous **R**eceiver **T**ransmitter) are chips on your PC serial card. Their purpose is to convert data to bits, send the bits down the serial line, and then rebuild the data again on the other end. UARTs deal with data in byte sized pieces, which is conveniently also the size of ASCII characters.

Say you have a terminal hooked up to your PC. When you type a character, the terminal gives that character to it's transmitter (also a UART of some sort). The transmitter sends that byte out onto the serial line, one bit at a time, at a specific rate. On the PC end, the receiving UART takes all the bits and rebuilds the byte and puts it in a buffer.

There are two different types of UARTs. You have probably heard of dumb UARTs - the 8250 and 16450, and FIFO UARTs - the 16550A. To understand their differences, first let's examine what happens when a UART has sent or received a byte.

The UART itself can't do anything with the data, it just sends and receives it. The CPU gets an interrupt from the serial device every time a byte has been sent or received. The CPU then moves the received byte out of the UART's buffer and into memory somewhere, or gives the UART another byte to send. The 8250 and 16450 UARTs only have a 1 byte buffer. That means, that every time 1 byte is sent or received, the CPU is interrupted. At low rates, this is OK. But, at high transfer rates, the CPU gets so busy dealing with the UART, that is doesn't have time to tend to other tasks. In some cases, the CPU does not get around to servicing the interrupt in time, and the byte is overwritten, because they are coming in so fast.

That's where the 16550A UARTs are useful. These chips come with 16 byte FIFOs. This means that it can receive or transmit up to 16 bytes before it has to interrupt the CPU. Not only can it wait, but the CPU then can transfer all 16 bytes at a time. Although the interrupt threshold is seldom set at 16, this is still a significant advantage over the other UARTs, which only have the 1 byte buffer. The CPU receives less interrupts, and is free to do other things. Data is not lost, and everyone is happy. (There is also a 16550 UART, but it is treated as a 16450)

In general, the 8250 and 16450 UARTs should be fine for speeds up to 38400 bps. At speeds greater than 38400 bps, you might start seeing data loss, and a reduction in interactive response time. Other PC operating systems (definition used loosely here), like DOS aren't multitasking, so they might be able to cope better with 8250 or 16450s. That's why some people don't see data loss, until they switch to Linux.

Non-UART, and intelligent multiport boards use DSP chips to do additional buffering and control, thus relieving the CPU even more. For example, the Cyclades Cyclom, and Stallion EasyIO boards use a Cirrus Logic CD-1400 RISC chip.

Keep in mind that these dumb UART types are not bad, they just aren't good for high speeds. You should have no problem connecting a terminal, or a mouse to these UARTs. But, for a high speed modem, the 16550A is definitely a must.

You can buy serial cards with the 16550A UARTs for a little more money, just ask your dealer what type of UART is on the card. Or if you want to upgrade your existing card, you can simply purchase 16550A chips and replace your existing 16450 UARTs. They are pin-to-pin compatible. Some cards come with socketed UARTs for this purpose, if not you can solder. Note, that you'll probably save yourself a lot of trouble by just getting a new card, if you've got the money, they are under US$ 50.

14.4 What's the real difference between the `/dev/cua`*N* and `/dev/ttyS`*N* devices?

The only difference is the way that the devices are opened. The dialin devices `/dev/ttyS`*N* are opened in blocking mode, until CD is asserted (ie someone connects). So, when someone wants to use a `/dev/cua`*N* device, there is no conflict with a program watching the `/dev/ttyS`*N* device.

The distinction is made to allow dialin and dialout use of the same serial port.

15 Troubleshooting

15.1 I keep getting "line *NNN* of inittab invalid".

Make sure you are using the correct syntax for your version of `init`. The different `init`'s that are out there use different syntax in the `/etc/inittab` file. Make sure you are using the correct syntax for your version of `getty`.

15.2 When I try to dial out, it says "/dev/cua*N*: Device or resource busy".

This problem can arise when DCD or DTR are not set correctly. DCD should only be set when there is an actual connection (ie someone is dialed in), not when `getty` is watching the port. Check to make sure that your modem is configured to only set DCD when there is a connection. DTR should be set whenever something is using, or watching the line, like `getty`, `kermit`, or some other comm program.

Another common cause of "device busy" errors, is that you set up your serial port with an interrupt already taken by something else. As each device initializes, it asks Linux for permission to use its hardware interrupt. Linux keeps track of which interrupt is assigned to whom, and if your interrupt is already taken, your device won't be able to initialize properly. The device really doesn't have much of any way to tell you that this happened, except that when you try to use it, it will return a "device-busy" error. Check the interrupts on all of your cards (serial, ethernet, SCSI, etc.). Look for IRQ conflicts.

15.3 I keep getting "Id SN respawning too fast: disabled for 5 minutes".

Make sure your modem is configured correctly. Look at registers `E` and `Q`. This can occur when your modem is chatting with `getty`.

Make sure you are calling `getty` correctly from your `/etc/inittab`. Using the wrong syntax or device names will cause serious problems.

This can also happen when the `uugetty` initialization is failing. Go to the "`getty` or `uugetty` still doesn't work" question.

15.4 Serial devices are slow or serial devices can only send in one direction.

You probably have an IRQ conflict. Make sure there are no IRQs being shared. Check all your cards (serial, ethernet, SCSI, etc...). Make sure the jumper settings, and the `setserial` parameters are correct for all your serial devices.

15.5 My modem is hosed after someone hangs up, or `uugetty` doesn't respawn.

This can happen when your modem doesn't reset when DTR is dropped. I saw my RD and SD LEDs go nuts when this happened to me. You need to have your modem reset. Most Hayes compatible modems do this with `&D3`, but on my USR Courier, I had to set `&D2` and `S13=1`. Check your modem manual.

15.6 I have my terminal connected to my PC, but after I type in a login name, it just locks up.

You probably don't have `CLOCAL` in your `/etc/gettydefs` entry for the terminal, and you're probably not using a full null modem cable. You need `CLOCAL`, which tells Linux to ignore modem control signals. Here is what it should look like:

```
# 38400 bps Dumb Terminal entry
DT38400# B38400 CS8 CLOCAL # B38400 SANE -ISTRIP CLOCAL #@S @L login: #DT38400
# 19200 bps Dumb Terminal entry
DT19200# B19200 CS8 CLOCAL # B19200 SANE -ISTRIP CLOCAL #@S @L login: #DT19200
# 9600 bps Dumb Terminal entry
DT9600# B9600 CS8 CLOCAL # B9600 SANE -ISTRIP CLOCAL #@S @L login: #DT9600
```

Next, `kill` the `getty` process so a new one will be spawned with the new entry.

15.7 At high speeds, my modem loses data.

If you are trying to run your modem at > 19200 bps, and you don't have 16550A UARTs, you should upgrade them. See section 14.3 about UARTs.

15.8 On startup, Linux doesn't report the serial devices the way I have them configured.

This is true. Linux does not do any IRQ detection on startup, it only does serial device detection. Thus, disregard what it says about the IRQ, because it's just assuming the standard IRQs. This is done, because IRQ detection is unreliable, and can be fooled.

So, even though I have my `ttyS2` set at IRQ 5, I still see

```
Jan 23 22:25:28 misfits vmunix: tty02 at 0x03e8 (irq = 4) is a 16550A
```

You have to use `setserial` to tell Linux the IRQ you are using.

15.9 `rz` and/or `sz` don't work when I call my Linux box on my modem.

If Linux looks for `/dev/modem` when you try to transfer files, look at `/etc/profile`, and `/etc/csh.cshrc`. There are a bunch of aliases defined there on some distributions, most notably Slackware. These aliases mess up the zmodem programs. Take them out, or correct them.

15.10 My screen is printing funny looking characters.

This happens on virtual consoles when you send binary data to your screen, or sometimes on serial connections. The way to fix this is to type `echo ^v^[c`. For the control-character-impaired, thats `echo` `<ctrl>v<esc>c`.

15.11 `getty` or `uugetty` still doesn't work.

There is a `DEBUG` option that comes with `getty_ps`. Edit your config file `/etc/conf.{uu}getty.ttyS`*N* and add `DEBUG=`*NNN*. Where *NNN* is one of the following combination of numbers according to what you are trying to debug:

```
D_OPT   001          option settings
D_DEF   002          defaults file processing
D_UTMP  004          utmp/wtmp processing
D_INIT  010          line initialization (INIT)
D_GTAB  020          gettytab file processing
D_RUN   040          other runtime diagnostics
D_RB    100          ringback debugging
D_LOCK  200          uugetty lockfile processing
```

```
D_SCH   400         schedule processing
D_ALL   777         everything
```

Setting `DEBUG=010` is a good place to start.

If you are running `syslogd`, debugging info will appear in your log files. If you aren't running `syslogd` info will appear in `/tmp/getty:ttyS`*N* for debugging `getty` and `/tmp/uugetty:ttyS`*N* for `uugetty`, and in `/var/adm/getty.log`. Look at the debugging info and see what is going on. Most likely, you will need to tune some of the parameters in your config file, and reconfigure your modem.

You could also try `mgetty`. Some people have better luck with it.

16 Contributions

There was no possible way to write this HOWTO alone. Although a lot of the HOWTO is my writing, I have rewritten many contributions to maintain continuity in the writing style and flow. Thanks to everyone who has contributed or commented, the list of people has gotten too long to list (somewhere over fifty). Special thanks to Ted T'so for answering questions about the serial drivers, Kris Gleason who maintains `getty_ps`, and Gert Doering who maintains `mgetty`.

END OF Serial-HOWTO