The Linux Serial HOWTO

by Greg Hankins, greg.hankins@cc.gatech.edu v1.10, 31 March 1997

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

Contents

1 Introduction **3**

1.1 Copyright . . . 3

1.2 New Versions Of This Document . . . 4

1.3 Feedback . . . 4

1.4 Disclaimer . . . 4

2 Supported Serial Hardware **4**

2.1 Standard PC Serial Boards . . . 5

2.2 Plug And Play (PnP) Modems . . . 5

2.3 Dumb Multiport Serial Boards (with 8250/16450/16550A UARTs) . . . 5

2.4 Intelligent Multiport Serial Boards . . . 6

2.5 Unsupported Serial Hardware . . . 8

3 What Are The Names Of The Serial Ports? **8**

3.1 Serial Port Devices and Numbers In `/dev` . . . 9

3.1.1 Creating Devices In `/dev` . . . 9

3.2 Notes For Dumb Multiport Boards . . . 9

3.3 Notes For Intelligent Multiport Boards . . . 10

4 Interesting Programs You Should Know About **10**

4.1 What is `getty`? . . . 10

4.1.1 About `getty_ps` . . . 10

4.1.2 About `mgetty` . . . 10

4.1.3 About `agetty` . . . 11

4.2 What is `setserial`? . . . 11

5 How Do I Dial Out With My Modem? **11**

5.1 Hardware Requirements . . . 11

5.1.1 Notes On Internal Modems . . . 11
5.2 Talking To Your Modem . . . 12
5.3 Dial Out Modem Configuration . . . 13
5.4 Hardware Flow Control . . . 13

6 How Do I Dial In And Out With My Modem Using `getty_ps`? 14
6.1 Dial In And Out Modem Configuration . . . 14
6.2 Installing `getty_ps` . . . 14
6.3 Setting up `uugetty` . . . 15
6.4 Customizing `uugetty` . . . 16

7 How Do I Set Up A Terminal Connected To My PC? 17
7.1 Hardware Requirements . . . 17
7.2 Setting up `getty` . . . 19
7.3 Notes On Setting Up A PC As A Terminal . . . 20

8 Can I Use More Than Two Serial Devices? 20
8.1 Choosing Serial Device Interrupts . . . 20
8.2 Setting Serial Device Addresses . . . 21

9 How Do I Set Up My Serial Ports For Higher Speeds? What Speed Should I Use With My Modem? 22

10 Communications Programs And Utilities 23

11 Serial Tips And Miscellany 23
11.1 `kermit` and zmodem . . . 23
11.2 Setting Terminal Types Automagically . . . 24
11.3 Color `ls` On Serial Connections . . . 24
11.4 Printing To A Printer Connected To A Terminal . . . 24
11.5 Can Linux Configure The Serial Devices Automagically? . . . 24
11.5.1 Notes For Multiport Boards . . . 24
11.6 Using A Serial Console . . . 24
11.7 Higher Serial Throughput . . . 25

12 One Step Further... 25

12.1 What Are Lock Files? . . . 25

12.2 "baud" Vs. "bps" . . . 25

12.3 What Are UARTs? How Do They Affect Performance? . . . 26

13 Troubleshooting 27

13.1 I Keep Getting "line *NNN* of inittab invalid" . . . 27

13.2 When I Try To Dial Out, It Says "/dev/cua*N*: Device or resource busy" . . . 27

13.3 I Keep Getting "Id S*N* respawning too fast: disabled for 5 minutes" . . . 27

13.4 Serial Devices Are Slow, Or Serial Devices Can Only Send In One Direction . . . 28

13.5 My Modem Is Hosed After Someone Hangs Up, Or `uugetty` doesn't respawn . . . 28

13.6 I Have My Terminal Connected To My PC, But After I Type In A Login Name, It Just Locks Up . . . 28

13.7 At High Speeds, My Modem Looses Data . . . 29

13.8 On Startup, Linux Doesn't Report The Serial Devices The Way I Have Them Configured . . . 29

13.9 `rz` And/Or `sz` Don't Work When I Call My Linux Box On A Modem . . . 29

13.10My Screen Is Printing Funny Looking Characters . . . 29

13.11`getty` Or `uugetty` Still Doesn't Work . . . 29

14 Other Sources Of Information 30

15 Contributions 31

1 Introduction

This is the Linux Serial HOWTO. All about how to set up modems and terminals under Linux, some serial tips, and troubleshooting. This HOWTO addresses Linux running on x86 hardware, although it might work for other architectures.

1.1 Copyright

The Linux Serial HOWTO is copyright (C) 1993 - 1997 by Greg Hankins. Linux HOWTO documents may be reproduced and distributed in whole or in part, in any medium physical or electronic, as long as this copyright notice is retained on all copies. Commercial redistribution is allowed and encouraged; however, the author would *like* to be notified of any such distributions.

All translations, derivative works, or aggregate works incorporating any Linux HOWTO documents must be covered under this copyright notice. That is, you may not produce a derivative work from a HOWTO

and impose additional restrictions on its distribution. Exceptions to these rules may be granted under certain conditions; please contact the Linux HOWTO coordinator at the address given below.

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.

1.2 New Versions Of This Document

New versions of the Serial HOWTO will be available at `ftp://sunsite.unc.edu:/pub/Linux/docs/HOWTO/Serial-HOWTO` and mirror sites. There are other formats, such as PostScript and DVI versions in the `other-formats` directory. The Serial HOWTO is also available at `http://sunsite.unc.edu/LDP/HOWTO/Serial-HOWTO.html` and will be posted to `comp.os.linux.answers` monthly.

1.3 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. All email will be answered, although it may take me a week or so depending on how busy I am. I get a lot of email from `root` and misconfigured sites. Please send me email as a real person, and make sure that your email system is working so that I will be able to reply to you. If you don't hear back from me within two weeks, please try sending your email again.

I can also be reached at:
Greg Hankins
College of Computing
801 Atlantic Drive
Atlanta, GA 30332-0280
via snail mail, and at `http://www.cc.gatech.edu/staff/h/Greg.Hankins/`.

Please include the version number of the Serial HOWTO when writing, this is version 1.10.

1.4 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.

2.1 Standard PC Serial Boards

- 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)

Note: due to address conflicts, you cannot use COM4 and IBM8514 video board simultaneously. This is due to a bug in the IBM8514 board.

2.2 Plug And Play (PnP) Modems

PnP modems will work with Linux. However, I recommend against using them because they are difficult. There are a couple things you can try;

- Try to disable the PnP features in the BIOS on your computer and modem.
- Boot Win95, configure the modem, and see if the settings are preserved when you boot Linux. If not, then:
- Boot Win95, configure the modem, and then *warm boot* Linux so that the settings are preserved. You can do this with the `loadlin` boot loader.
- Get the PnP tools from `http://www.redhat.com/linux-info/pnp`.
- And of course you can get a real modem.

2.3 Dumb Multiport Serial Boards (with 8250/16450/16550A UARTs)

- AST FourPort and clones (4 ports)
- Accent Async-4 (4 ports)
- Arnet Multiport-8 (8 ports)
- Bell Technologies HUB6 (6 ports)
- Boca BB-1004 (4 ports), BB-1008 (8 ports), BB-2016 (16 ports)
- Boca IOAT66 (6 ports)
- Boca 2by4 (4 serial ports, 2 parallel ports)
- Computone ValuePort V4-ISA (AST FourPort compatible)
- Digi PC/8 (8 ports)
- GTEK BBS-550 (8 ports)

- Longshine LCS-8880, Longshine LCS-8880+ (AST FourPort compatible)
- Moxa C104, Moxa C104+ (AST FourPort compatible)
- PC-COMM (4 ports)
- SIIG I/O Expander 2S IO1812 (4 ports)
- STB-4COM (4 ports)
- Twincom ACI/550
- Usenet Serial Board II (4 ports)

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.

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

2.4 Intelligent Multiport Serial Boards

- Comtrol RocketPort (36MHz ASIC; 4, 8, 16 or 32 ports)
  contact: `info@comtrol.com` or `http://www.comtrol.com`
  driver status: supported by Comtrol
  driver location: `ftp://tsx-11.mit.edu/pub/linux/packages/comtrol`
- Computone IntelliPort II (16MHz 80186; 4, 8, or 16 ports),
  IntelliPort II EXpandable (20MHz 80186; 16 - 64 ports)
  contact: Michael H. Warfield, `mhw@wittsend.atl.ga.us`
  driver status: pre-ALPHA
- Cyclades Cyclom-Y (Cirrus Logic CD1400 UARTs; 8 - 32 ports),
  Cyclom-Z (25MHz MIPS R3000; 8 - 128 ports)
  contact: `sales@cyclades.com` or `http://www.cyclades.com`
  driver status: supported by Cyclades
  driver location: `ftp://ftp.cyclades.com/pub/cyclades` and included in Linux kernel since version 1.1.75
- Decision PCCOM8 (8 ports)
  contact: `pccom8@signum.se`
  driver location: `ftp://ftp.signum.se/pub/pccom8`
- Digi PC/Xi (12.5MHz 80186; 4, 8, or 16 ports),
  PC/Xe (12.5/16MHz 80186; 2, 4, or 8 ports),
  PC/Xr (16MHz IDT3041; 4 or 8 ports),
  PC/Xem (20MHz IDT3051; 8 - 64 ports)
  contact: `sales@dgii.com` or `http://www.dgii.com`
  driver status: supported by Digi
  driver location: `ftp://ftp.dgii.com/drivers/linux` and included in Linux kernel since version 2.0

- Digi COM/Xi (10MHz 80188; 4 or 8 ports)
  contact: Simon Park, si@wimpol.demon.co.uk
  driver status: ALPHA
  note: Simon is often away from email for months at a time due to his job. Mark Hatle, fray@krypton.mankato.msus.edu has graciously volunteered to make the driver available if you need it. Mark is not maintaining or supporting the driver.

- Equinox SuperSerial Technology (30MHz ASIC; 2 - 128 ports)
  contact: sales@equinox.com or http://www.equinox.com
  driver status: supported by Equinox
  driver location: ftp://ftp.equinox.com/library/sst

- GTEK Cyclone (16C654 UARTs; 6, 16 and 32 ports),
  SmartCard (24MHz Dallas DS80C320; 8 ports),
  BlackBoard-8A (16C654 UARTs; 8 ports),
  PCSS (15/24MHz 8032; 8 ports)
  contact: spot@gtek.com or http://www.gtek.com
  driver status: supported by GTEK
  driver location: ftp://ftp.gtek.com/pub

- Hayes ESP (COM-bic; 1 - 8 ports)
  contact: Andrew J. Robinson, arobinso@nyx.net or http://www.nyx.net/~arobinso
  driver status: supported by author
  driver location: http://www.nyx.net/~arobinso and included in Linux kernel since version 2.1.15

- Maxpeed SS (Toshiba; 4, 8 and 16 ports)
  contact: info@maxpeed.com or http://www.maxpeed.com
  driver status: supported by Maxpeed
  driver location: ftp://maxpeed.com/pub/ss

- Moxa C218 (12MHz 80286; 8 ports),
  Moxa C320 (40MHz TMS320; 8 - 32 ports)
  contact: info@moxa.com.tw or http://www.moxa.com.tw
  driver status: supported by Moxa
  driver location: ftp://ftp.moxa.com.tw/drivers/c218-320/linux

- SDL RISCom/8 (Cirrus Logic CD180; 8 ports)
  contact: sales@sdlcomm.com or http://www.sdlcomm.com
  driver status: supported by SDL
  driver location: ftp://ftp.sdlcomm.com/pub/drivers

- Omega COMM-8 (8 ports)
  contact: Vance Petree, vpetree@infi.net

- Specialix SIO (20MHz Zilog Z280; 4 - 32 ports),
  XIO (25MHz Zilog Z280; 4 - 32 ports)
  contact: Simon Allen, simonallen@cix.compulink.co.uk
  driver status: BETA

driver location: `ftp://sunsite.unc.edu/pub/Linux/kernel/patches/serial`

- Stallion EasyIO-4 (4 ports), EasyIO-8 (8 ports), and
  EasyConnection (8 - 32 ports) - each with Cirrus Logic CD1400 UARTs,
  Stallion (8MHz 80186 CPU; 8 or 16 ports),
  Brumby (10/12 MHz 80186 CPU; 4, 8 or 16 ports),
  ONboard (16MHz 80186 CPU; 4, 8, 12, 16 or 32 ports),
  EasyConnection 8/64 (25MHz 80186 CPU; 8 - 64 ports)
  contact: `sales@stallion.com` or `http://www.stallion.com`
  driver status: supported by Stallion
  driver location: `ftp://ftp.stallion.com/drivers/ata5/Linux` and included in linux kernel since 1.3.27

A review of Comtrol, Cyclades, Digi, and Stallion products was printed in the June 1995 issue of the *Linux Journal*. The article is available at `http://www.ssc.com/lj/issue14`.

2.5 Unsupported Serial Hardware

Modems that claim to be "Winmodems", for example the USR Sportster Winmodem and IBM Aptiva MWAVE, are *not* supported under Linux. These modems use proprietary designs that require special Windows drivers. Avoid these types of modems. In addition, stay away from modems that require Rockwell RPI drivers for the same reason.

Intelligent serial boards that require drivers not available for Linux won't work either.

3 What Are The Names Of The Serial Ports?

An I/O port is a way to get data into and out of a computer. There are many types of I/O ports such as serial ports, parallel ports, disk drive controllers, ethernet boards, etc. We will be dealing with serial ports since modems and terminals are serial devices. Each serial port must have an I/O address, and an interrupt (IRQ). There are the four serial ports corresponding to COM1 - COM4:

```
ttyS0 (COM1) address 0x3f8 IRQ 4
ttyS1 (COM2) address 0x2f8 IRQ 3
ttyS2 (COM3) address 0x3e8 IRQ 4
ttyS3 (COM4) address 0x2e8 IRQ 3
```

If Linux does not detect any serial ports when it boots, then make sure that serial support is enabled and compiled into the kernel. In this document, I refer to COM1 as `ttyS0`, COM2 as `ttyS1`, COM3 as `ttyS2`, and COM4 as `ttyS3`. Notice that by default these devices have overlapping IRQs. You cannot use all of the ports in this default configuration, and you must reassign different IRQs. See section 8 on setting IRQs.

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 device in `/dev` 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 may run into problems because the lock files may not work correctly if you use `/dev/modem`. Use them if you like, but *be sure they point to the right device*. However, if you change or remove this link, some applications (`minicom` for example) might need reconfiguration.

3.1 Serial Port Devices and Numbers 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. You can verify this by typing:

```
linux% ls -l /dev/cua*
linux% ls -l /dev/ttyS*
```

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 use the `MAKEDEV` script, which lives in `/dev`. 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, and should set the correct permissions.

3.2 Notes For Dumb Multiport Boards

The devices your multiport board uses depends on what kind of board you have. These are listed in detail in `rc.serial` which comes with the `setserial` program. I highly recommend getting the latest version of `setserial` if you are trying to use multiport boards. You will probably need to create these devices. Either use the `mknod` command, or the `MAKEDEV` script. Devices for multiport boards are made by adding "64 + 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
```

Note: the SIIG manual for the IO1812 listing for COM5-COM8 is wrong. They should be COM5=0x250, COM6=0x258, COM7=0x260, and COM8=0x268.

Note: the Digi PC/8 Interrupt Status Register is at 0x140.

Note: for an AST Fourport, you might need to specify `skip_test` in `rc.serial`.

3.3 Notes For Intelligent Multiport Boards

Read the information that comes with the driver. These boards use special devices, and not the standard ones. This information varies depending on your hardware.

4 Interesting Programs You Should Know About

4.1 What is getty?

`getty` is a program that handles the login process when you log onto a Unix box. You will need to use `getty` if you want to be able to dial in to your Linux machine with a modem. You do not need to use `getty` if you only want to dial out with your modem. There are three versions that are commonly used with Linux: `getty_ps`, `mgetty` and `agetty`. The syntax for these programs differs, so be sure to check and make sure that you are using the correct syntax for whatever `getty` you use.

4.1.1 About getty_ps

Most distributions come with the `getty_ps` package installed. It contains two programs: `getty` is used for console and terminal devices, and `uugetty` for modems. I use this version of `getty`, so that is what I will focus on.

4.1.2 About mgetty

`mgetty` is a version of `getty` for use with modems. In addition to allowing dialup logins, `mgetty` also provides FAX support and auto PPP detection. The `mgetty` documentation is very good, and does not need supplementing. Please refer to it for installation instructions. You can find the latest information on `mgetty` at `http://www.leo.org/~doering/mgetty/`.

4.1.3 About agetty

`agetty` is the third variation of `getty`. It's a simple, completely functional implementation of `getty` which is best suited for virtual consoles or terminals rather than modems.

4.2 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. 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.

5 How Do I Dial Out With My Modem?

5.1 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.

5.1.1 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 8 if you need help on choosing interrupts or addresses.

On some motherboards you will have to disable the serial port that the modem is replacing in order to avoid conflicts. This may be done with jumpers or in the BIOS settings, depending on your motherboard. Consult your motherboard manual.

Due to a bug in IBM8514 video boards, 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. Linux does not do any autoconfiguration on `ttyS3` due to this video board bug.

5.2 Talking To Your Modem

Use `kermit` or some other communications program to test the setup, before you go jumping into complex things SLIP or PPP. You can find the latest version of `kermit` at `http://www.columbia.edu/kermit/`. For example, say your modem was on `ttyS3`, and it's speed was 115200 bps. You would do the following:

```
linux# kermit
C-Kermit 6.0.192, 6 Sep 96, for Linux
 Copyright (C) 1985, 1996,
  Trustees of Columbia University in the City of New York.
Default file-transfer mode is BINARY
Type ? or HELP for help.
C-Kermit>set line /dev/ttyS3
C-Kermit>set carrier-watch off
C-Kermit>set speed 115200
/dev/ttyS3, 115200 bps
C-Kermit>c
Connecting to /dev/ttyS3, speed 115200.
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.
ATE1Q0V1                          ; you type this and then the Enter key
OK                                ; modem should respond with this
```

If your modem responds to `AT` commands, you can assume your modem is working correctly on the Linux side. Now try calling another modem by typing:

```
ATDT7654321
```

where 7654321 is a phone number. Use ATDP instead of ATDT if you have a pulse line. If the call goes through, your modem is working.

To get back to the `kermit` prompt, hold down the Ctrl key, press the backslash key, then let go of the Ctrl key, then press the C key:

```
Ctrl-\-C
(Back at linux)
C-Kermit>quit
linux#
```

This was just a test using the primitive "by-hand" dialing method. The normal method is to let `kermit` do the dialing for you with its built-in modem database and automatic dialing features, for example using a US Robotics (USR) modem:

```
linux# kermit
C-Kermit 6.0.192, 6 Sep 1997, for Linux
 Copyright (C) 1985, 1996,
  Trustees of Columbia University in the City of New York.
```

```
Default file-transfer mode is BINARY
Type ? or HELP for help
C-Kermit>set modem type usr        ; Select modem type
C-Kermit>set line /dev/ttyS3       ; Select communication device
C-Kermit>set speed 115200          ; Set the dialing speed
C-Kermit>dial 7654321              ; Dial
 Number: 7654321
 Device=/dev/ttyS3, modem=usr, speed=115200
 Call completed.<BEEP>
Connecting to /dev/ttyS3, speed 115200
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.

Welcome to ...

login:
```

See section 10 about communications 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. Versions of Linux with a `libc` version greater then 5.x have support for speeds up to 115200 bps. `libc` usually lives in `/lib`, so look there for what version you have. If Linux does not recognize a speed of 57600 or 115200 bps, then you must use the `setserial` program to set your serial port to a higher speed. See section 9 for how to do this. Then, set the speed to 38400 bps in your comm program.

5.3 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. In general, factory defaults that enable error correction and hardware flow control are the best setting for dial out modems, consult your modem manual for these settings.

5.4 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.sysinit` file! You need to do the following for each serial port you want to enable hardware flow control on:

```
stty crtscts < /dev/ttyS3
```

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.

6 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 5 go *read it now*! It contains *very* important setup information. You *do not need* to read this section if you only want to *dial out* with your modem.

6.1 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! You can get them at `ftp://ftp.cc.gatech.edu/pub/people/gregh/modem-configs`.

Note: to get my USR Courier V.34 modem to reset correctly when DTR drops, I had to set `&D2` and `S13=1` (this sets bit 0 of register S13). This has been confirmed to work on USR Sportster V.34 modems as well.

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

6.2 Installing getty_ps

Get the latest version from `sunsite.unc.edu:/pub/Linux/system/serial`. In particular, if you want to use high speeds (57600 and 115200 bps), you must get version 2.0.7j or later. You must also have libc 5.x

or greater.

By default, `getty_ps` will be configured to be Linux FSSTND (File System Standard) compliant, which means that the binaries will be in `/sbin`, and the config files will be named `/etc/conf.{uu}getty.ttyS`N. 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.ttyS`N, and lock files will go in `/usr/spool/uucp`. I recommend doing things this way if you are using UUCP, because UUCP will have problems if you move the lock files to where it isn't looking for them.

`getty_ps` can also use `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.

Decide on if you want FSSTND compliance and syslog capability. You can also choose a combination of the two. Edit the `Makefile`, `tune.h` and `config.h` to reflect your decisions. Then compile and install according to the instructions included with the package.

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. These instructions will not work for `mgetty` or `agetty`.

6.3 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 if they aren't already there (note that the entries point to each other, these are not for fixed speed - blank lines are needed between each entry):

```
# Modem entries
115200# B115200 CS8 # B115200 SANE -ISTRIP HUPCL #@S @L @B login: #57600

57600# B57600 CS8 # B57600 SANE -ISTRIP HUPCL #@S @L @B login: #38400

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: #115200
```

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 speeds. Then, instead of the step down series of lines listed above `/etc/gettydefs` only needs to contain one line for the modem, for example:

```
# 115200 fixed speed
F115200# B115200 CS8 # B115200 SANE -ISTRIP HUPCL #@S @L @B login: #F115200
```

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

```
# 115200 fixed speed with hardware flow control
F115200# B115200 CS8 CRTSCTS # B115200 SANE -ISTRIP HUPCL CRTSCTS #@S @L @B login: #F115200
```

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. If you want to be able to dial out with your modem while `uugetty` is watching the port for logins, use the `/dev/cua`*N* device instead of the `/dev/ttyS`*N* device.

When you are done editing `/etc/gettydefs`, you can verify that the syntax is correct by doing:

```
linux# getty -c /etc/gettydefs
```

6.4 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 - config file location, port, speed, and default terminal type):

```
S3:456:respawn:/sbin/uugetty -d /etc/default/uugetty.ttyS3 ttyS3 F115200 vt100
```

Restart `init`:

```
linux# init q
```

For the speed parameter in your `/etc/inittab`, you want to use the highest bps rate that your modem supports.

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

`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.

7 How Do I Set Up A Terminal Connected To My PC?

The instructions in this section will work for connecting terminals, as well as other computers to the serial port on your Linux box.

7.1 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! Many computer stores advertise this kind of cable as a serial printer cable. Make sure you are using your serial port, the male DB25 or the DB9, and not your parallel port (female DB25 or centronics).

For a DB25 connector, you need a minimum of:

```
PC male DB25                              Terminal DB25
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:

```
PC male DB25                              Terminal DB25
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
                            |
DCD   Carrier Detect        8 <-- 20    DTR   Data Terminal Ready
SG    Signal Ground         7 --- 7     SG    Signal Ground
                                  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:

```
PC DB9                                    Terminal 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:

```
PC DB9                                    Terminal DB25
RxD   Receive Data          2 <-- 2     TxD   Transmit Data
TxD   Transmit Data         3 --> 3     RxD   Receive Data
                                  6     DSR   Data Set Ready
                                  |
DTR   Data Terminal Ready   4 --> 8     DCD   Carrier Detect
GND   Signal Ground         5 --- 7     GND   Signal Ground
DCD   Carrier Detect        1
                            |
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)
```

(Yes, the pins 2 and 3 *really do* have the opposite meanings in DB9 connectors than in DB25 connectors!)

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 the terminal to ignore modem control signals. Try using 9600 bps, 8 data bits, 1 stop bit, no parity bits for the terminal's setup.

7.2 Setting up `getty`

Add entries for `getty` to use for your terminal in `/etc/gettydefs` if there they aren't already there:

```
# 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 '@@'.
```

When you are done editing `/etc/gettydefs`, you can verify that the syntax is correct by doing:

```
linux# getty -c /etc/gettydefs
```

Make sure there is no `getty` or `uugetty` config file for the serial port that your terminal is attached to (`/etc/default/{uu}getty.ttyS`*N* or `/etc/conf.{uu}getty.ttyS`*N*), as this will probably interfere with running `getty` on a terminal. Remove the file if it exits.

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.

7.3 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 80286 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 every OS in existence at `http://www.columbia.edu/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.

8 Can I Use More Than Two Serial Devices?

You don't need to read this section, unless you want to use three 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 (IRQs) 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 board.

8.1 Choosing Serial Device Interrupts

Your PC will normally come with `ttyS0` and `ttyS2` at IRQ 4, and `ttyS1` and `ttyS3` at IRQ 3. You can see what IRQs are in use by looking at `/proc/interrupts`. To use more than two serial devices, you will have to reassign an interrupt. 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 two 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:

```
/sbin/setserial /dev/ttyS0 irq 3        # my serial mouse
/sbin/setserial /dev/ttyS1 irq 4        # my Wyse dumb terminal
/sbin/setserial /dev/ttyS2 irq 5        # my Zoom modem
/sbin/setserial /dev/ttyS3 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 1
IRQ 15    Hard disk controller 2
```

There is really no Right Thing to do when choosing interrupts. Just make sure it isn't being used by the motherboard, or any other boards. 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 board 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. When you are done, double-check `/proc/interrupts` and make sure there are no conflicts.

8.2 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. You can check `/proc/ioports` to see what I/O port addresses are in use after Linux boots.

9 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 bps (V.21/Bell 103), 1200 bps (V.22/Bell 212A), or 2400 bps (V.22bis).
- If you have a 9600 bps (V.32) modem, with V.42bis data compression, use 38400 bps 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 or 33600 bps (V.FC or V.34) modem (4 * 28800 = 115200).

  Then, use 38400 bps as the speed in your communication program, or `/etc/inittab`. This is now the high speed you have set. Make sure you have 16550A UARTs.

  If your libc version is at least 5.x, there will be speeds named 57600 and 115200. `libc` lives in `/lib`, so look there for what version you have. You can use these directly (without using `setserial`), if your applications have been compiled to take advantage of this. There are many distributions out there, so the best thing to do would be to try using these higher speeds if you have a recent version of a Linux distribution.

Test your `setserial` setting on the command line first, and then when you have them working, put them 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/ttyS3 spd_vhi
```

Make sure that you are using a valid path for `setserial`, and a valid device name. You can check the settings of a serial port by running:

```
setserial -a /dev/ttyS3
```

10 Communications Programs And Utilities

Here is a list of some communication software you can choose from, available via FTP, if they didn't come with your distribution.

- `ecu` - a communications program
- *C-Kermit* <`http://www.columbia.edu/kermit/`> - portable, scriptable, serial and TCP/IP communications including file transfer and character-set translation
- `minicom` - `telix`-like communications program
- `procomm` - `procomm`-like communications program with zmodem
- `seyon` - X based communication program
- `xc` - xcomm communication package
- Other useful programs are `term` and `SLiRP`. They offer TCP/IP functionality using a shell account.
- `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.
- `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.
- SLIP and PPP software can be found at `ftp://sunsite.unc.edu/pub/Linux/system/network/serial`.
- Other things can be found on `ftp://sunsite.unc.edu/pub/Linux/system/serial` and `ftp://sunsite.unc.edu/pub/Linux/apps/comm` or one of the many mirrors. These are the directories where serial programs are kept.

11 Serial Tips And Miscellany

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

11.1 `kermit` and zmodem

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

```
define rz !rz < /dev/ttyS3 > /dev/ttyS3
define sz !sz \%0 > /dev/ttyS3 < /dev/ttyS3
```

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.

11.2 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 use `tset`, which can establish terminal characteristics when you log in, and doesn't depend on any defaults.

11.3 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.

11.4 Printing To A Printer Connected To A Terminal

There is a program called `vtprint`, available from `ftp://ftp.sdsu.edu/pub/vtprint`, and from `http://www.sdsu.edu/~garrett`.

Another program that will do this is called `xprt`. It can be found at `ftp://sunsite.unc.edu/pub/Linux/system/printing`.

11.5 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/ttyS3 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.

11.5.1 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.

11.6 Using A Serial Console

There is an article in issue 36 of the *Linux Journal*, `http://www.ssc.com/lj/issue36/index.html` that explains how to use a serial console. Unfortunately, email to the author bounces. I hope that the article will be on the *Linux Journal* web site at the above URL soon.

11.7 Higher Serial Throughput

If you are seeing slow throughput and serial port overruns on a system with (E)IDE disk drives, you can get `hdparm`. This is a utility that can modify (E)IDE parameters, including unmasking other IRQs during a disk IRQ. This will improve repsonsiveness and will help eliminate overruns. Be sure to read the man page very carefully, since some drive/controller combinations don't like this and may corrupt the filesystem.

Also have a look at a utility called `irqtune` that will change the IRQ priority of a device, for example the serial port that your modem is on. This may improve the serial throughput on your system. The `irqtune` FAQ is at `http://www.best.com/ cae/irqtune`.

12 One Step Further...

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

12.1 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.

12.2 "baud" Vs. "bps"

"baud" and "bps" are perhaps one of the most misused terms in the computing and telecommunications 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 a signal, for instance one sent by a modem (**mo**dulator-**dem**odulator) changes. For example, a baud rate of 1200 implies one signal change every 833 microseconds. Common modem 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. Multiple bits can be encoded per baud, to get bit rates that exceed the baud rate. "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 modem bps rates are 50, 75, 110, 300, 1200, 2400, 9600, ... 115200. Using modems with V.42bis compression (max 4:1 compression), *theoretical* rates up to 115200 bps are possible. 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 achive a bps rate greater than baud rate by encoding many bits in each signal change, or phase change. 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 phase change, at 2400 baud.

How did this confusion start? Well, back when low speed modems were high speed modems, the bps rate actually did equal the baud rate. One bit would be encoded per phase change. 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.

12.3 What Are UARTs? How Do They Affect Performance?

UARTs (**U**niversal **A**syncronous **R**eceiver **T**ransmitter) are chips on your PC serial board. 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). 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 since it is broken.)

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. 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 CD1400 RISC UART, and many boards use 80186 CPUs or even special RISC CPUs, to handle the serial I/O.

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 boards with the 16550A UARTs for a little more money, just ask your dealer what type of UART is on the board. Or if you want to upgrade your existing board, you can simply purchase 16550A chips and replace your existing 16450 UARTs. They are pin-to-pin compatible. Some boards 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 board, if you've got the money, they are under US$50.

13 Troubleshooting

13.1 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`.

13.2 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 boards (serial, ethernet, SCSI, etc.). Look for IRQ conflicts.

13.3 I Keep Getting "Id S*N* 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.

Verify that your `/etc/gettydefs` syntax is correct by doing the following:

```
linux# getty -c /etc/gettydefs
```

This can also happen when the `uugetty` initialization is failing. See section 13.11.

13.4 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 boards (serial, ethernet, SCSI, etc...). Make sure the jumper settings, and the `setserial` parameters are correct for all your serial devices. Also check `/proc/ioports` and `/proc/interrups` for conflicts.

13.5 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 crazy 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.

13.6 I Have My Terminal Connected To My PC, But After I Type In A Login Name, It Just Locks Up

- If you are using `getty`: 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.

- If you are using `agetty`: Add the `-L` flag to the `agetty` line in your `/etc/initab`. This will make it ignore modem control signals. Then restart `init` by typing `init q`. The entry should look like this:

  ```
  s1:345:respawn:/sbin/agetty -L 9600 ttyS1 vt100
  ```

13.7 At High Speeds, My Modem Looses Data

If you are trying to run your modem faster than 38400 bps, and you don't have 16550A UARTs, you should upgrade them. See section 12.3 about UARTs.

13.8 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
```

when Linux boots. You have to use `setserial` to tell Linux the IRQ you are using. After Linux boots, you can look at the `/proc/interrupts` file, to see what IRQs are really configured.

13.9 rz And/Or sz Don't Work When I Call My Linux Box On A 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.

13.10 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:

```
linux% echo <ctrl>v<esc>c
```

13.11 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.

14 Other Sources Of Information

- man pages for: `agetty(8)`, `getty(1m)`, `gettydefs(5)`, `init(1)`, `login(1)`, `mgetty(8)`, `setserial(8)`
- Your modem manual
- NET-3 HOWTO: all about networking, including SLIP, CSLIP, and PPP
- PPP HOWTO: help with PPP
- Printing HOWTO: for setting up a serial printer
- Term HOWTO: everything you wanted to know about the `term` program
- UPS HOWTO: setting up UPS boxen connected to your serial port
- UUCP HOWTO: for information on setting up UUCP
- Usenet newsgroups:

```
comp.os.linux.answers
        FAQs, How-To's, READMEs, etc. about Linux.
comp.os.linux.hardware
        Hardware compatibility with the Linux operating system.
comp.os.linux.networking
        Networking and communications under Linux.
comp.os.linux.setup
        Linux installation and system administration.
```

- 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. The server also serves many other Linux lists. Send the "`lists`" command for a list of mailing lists.
- A white paper discussing serial communications and multiport serial boards is available from Cyclades at `http://www.cyclades.com`.

- Modem FAQs:
  Navas 28800 Modem FAQ <http://web.aimnet.com/~jnavas/modem/faq.html>
  Curt's High Speed Modem Page <http://www.teleport.com/~curt/modems.html>
- Serial programming: *Advanced Programming in the UNIX Environment*
  <http://heg-school.aw.com/cseng/authors/stevens/advanced/advanced.nclk>, by W. Richard Stevens (ISBN 0-201-56317-7; Addison-Wesley)
  <http://www.ora.com/catalog/posix/> NAME="POSIX Programmer's Guide"¿, by Donald Lewine (ISBN 0-937175-73-0; O'Reilly)

15 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 one hundred). Special thanks to Ted T'so for answering questions about the serial drivers, Kris Gleason who used to maintain `getty_ps`, and Gert Döring who maintains `mgetty`.

END OF SERIAL HOWTO