Remote Serial Console HOWTO

Glen Turner

Australian Academic and Research Network

glen.turner+howto@aarnet.edu.au

Mark F. Komarinski

mkomarinski@valinux.com

v2.0 2002/02/02

Revision History

Revision 2.0	2002−02−02	Revised by: gdt
Second edition.		
Revision d1.0	2001−03−20	Revised by: mfk
First edition.		

An RS−232 serial console allows Linux to be controlled from a terminal or modem attached to an asynchronous serial port. The monitor, mouse and keyboard are no longer required for system administration. Serial consoles are useful where Linux systems are deployed in remote sites or are deployed in high−density racks.

This *HOWTO* describes how to configure Linux to attach a serial console.

Table of Contents

Table of Contents

Table of Contents

Chapter 1. Preliminaries

1.1. Copyright

1.1.1. Linux Documentation Project License

1.2. Disclaimer

No liability for the contents of this documents can be accepted. Use the concepts, examples and other content at your own risk. As this is a new edition of this document, there may be errors and inaccuracies, that may of course be damaging to your system. Proceed with caution, and although this is highly unlikely, the author(s) do not take any responsibility for that.

All copyrights are held by their by their respective owners, unless specifically noted otherwise. Use of a term in this document should not be regarded as affecting the validity of any trademark or service mark.

Naming of particular products or brands should not be seen as endorsements.

You are strongly recommended to take a backup of your system before major installation and backups at regular intervals.

1.3. Acknowledgments

The first edition of this *HOWTO* was written by Mark Komarinski. It was based upon `/usr/src/linux/Documentation/serial-console.txt`, which was written by Miquel van Smoorenburg.

The second edition of this *HOWTO* was written by the staff of the Australian Academic and Research Network, mainly Glen Turner and David Vu. The revised text was proof read by the members of the LinuxSA mailing list. LinuxSA is a Linux user group based in South Australia.

David Lawyer, author of the *Text−Terminal−HOWTO*, reviewed the second edition. Many thanks to David for his useful feedback.

Glen Turner would like to thank his family for allowing him to work on this project for the surprisingly large number of evenings which it took to write this *HOWTO*. Thanks Karen, Kayla and Ella.

1.4. Comments and corrections

Without your submissions this document would not exist. Linux is continually improving and your author is not a professional editor. Please send corrections, additions, comments and criticisms to the maintainer, currently `<glen.turner+howto@aarnet.edu.au>`.

Chapter 2. Introduction

console n. [From latin consolatio(n) "comfort, spiritual solace."] A device for displaying or printing condolances or obituaries for the operator.

Stan Kelly−Bootle, The Computer Contradictionary.

2.1. What is a console?

The console is the text output device for system administration messages. These messages come from the kernel, from the init system and from the system logger.

On modern small computers the console is usually the computer's attached monitor and keyboard.

On many older computers the console is an RS−232 link to a terminal such as a DEC VT100. This terminal is in a locked room and is continually observed by the minicomputer's operators. Large systems from Sun, Hewlett−Packard and IBM still use serial consoles.

It is usually possible to login from the console. A login session from the console is treated by many parts of the operating system as being more trustworthy than a login session from other sources. Logging in as the root super−user from the console is the Command Line of Last Resort when faced with a misbehaving system.

2.2. Why use a serial console?

For the average user a serial console has no advantage over using a directly attached keyboard and screen. Serial consoles are much slower, taking up to a second to fill a 80 column by 24 line screen. Serial consoles generally only support non−proportional ASCII text, with limited support for languages other than English. A new terminal can be more expensive than an old PC.

There are some scenarios where serial consoles are useful. These are:

Systems administration of remote computers

> Linux is an good operating system for deployment at unstaffed sites. Linux is also good at hosting critical network infrastructure such as DNS and DHCP servers. These servers are generally installed at every site of an organisation including sites which may be too small or too remote to have IT staff.
>
> System administration of these remote computers is usually done using SSH, but there are times when access to the console is the only way to diagnose and correct software failures. Major upgrades to the installed distribution may also require console access.
>
> In these cases the serial console is attached to a modem. Access to the console is gained from a remote computer by dialing into the modem. This allows the console to be reached from any telephone socket.

High density racks of computers

Clusters of personal computers can outperform mainframe computers and form competitive supercomputers for some applications. See the *Cluster−HOWTO* for more information on clustering.

These clusters are typically assembled into 19−inch telecommunications equipment racks and the system unit of each computer is typically one rack unit (or 1.75 inches) tall. It is not desirable to put a keyboard and monitor on each computer, as a small cathode ray tube monitor would consume the space used by sixteen rack units.

A first glance it seems that a monitor and keyboard switch is the best solution. However the VGA signal to the monitor is small, so even with the switch the monitor cannot be placed very far away from the rack of computers.

It is desirable to allow the consoles to be monitored in the operators' room of the computer center, rather than in the very expensive space of the machine room. Although monitor switches with remote control and fiber optical extensions are available, this solution can be expensive.

A standard RS−232 cable can be 15 meters in length. Longer distances are easily possible. The cabling is cheap. Terminal servers can be used to allow one terminal to be access up to 90 serial consoles.

Recording console messages

This is useful in two very different cases.

Kernel programmers are often faced with a kernel error message that is displayed a split second before the computer reboots. A serial console can be used to record that message. Another Linux machine can be used as the serial terminal.

Some secure installations require all security events to be unalterably logged. A way to meet this requirement is to print all console messages. Connecting the serial console to a serial printer can achieve this.[1]

Embedded software development

Linux is increasingly being as the operating system in embedded applications. These computers do not have keyboards or screens.

A serial port is a cheap way to allow software developers to directly access the embedded computer. This is invaluable for debugging. Most chip sets designed for embedded computers have a serial port precisely for this purpose.

The shipping product need not present the RS−232 port on an external connector. Alternatively the RS−232 port is often used for downloading software updates.

Unlike minicomputer systems, the IBM PC was not designed to use a serial console. This has two consequences.

Firstly, Power On Self−Test messages and Basic Input/Output System (BIOS) messages are sent to the screen and received from the keyboard. This makes it difficult to reconfigure the BIOS and seeing makes it impossible to see Power On Self−Test errors.

An increasing number of manufacturers of rackable *server* equipment are altering their BIOSs to optionally use the RS−232 port as the console. If you are buying a machine specifically for use with serial console you should seek this feature. If you have an existing machine that definitely requires access to the BIOS from the serial port then there are hardware solutions such as PC Weasel 2000.

Secondly, the RS−232 port on the IBM PC is designed for connecting to a modem. Thus a null modem cable is needed when connecting the PC's serial port to a terminal.

2.3. Alternative meanings of console

Some authors use the word console to refer to the keyboard and monitor that are attached to the system unit. This is described as a physical console by some Linux documentation. The console where system messages appear is described as the logical console by that documentation.

As an illustration of the difference, X Windows should start on the physical console but system messages issued by failures when starting X Windows should be written to the logical console.

To avoid confusion this *HOWTO* uses the word console to describe the place where system messages are printed. This *HOWTO* uses the phrase "attached monitor and keyboard" rather than the confusing words "physical console".

These distinctions are also made in the naming of devices. The device `/dev/console` is used to send messages to the console. The symbolic link `/dev/systty` points to the device which is used by the attached monitor and keyboard, often `/dev/tty0`.

Table 2−1. Different ways of referring to the console

This *HOWTO*	console	Attached monitor and keyboard
Some Linux documentation	Logical console	Physical console
Device names	`/dev/console`	`/dev/systty`

2.4. Configuration overview

There are four major steps to configuring a serial console.

1. The boot loader must be configured to use the serial port.
2. The Linux kernel must be configured to use the serial port as its console. This is done by passing the kernel the `console` parameter when the kernel is started by the boot loader.

3. The init system should keep a process running to monitor the serial console for logins. The monitoring process is traditionally called getty.
4. A number of system utilities need to be configured to make them aware of the console, or configured to prevent them from disrupting the console.

Examples in this *HOWTO* are from Red Hat Linux versions 7.1 and 7.2, which were released in 2001. The maintainer would appreciate updates for later versions of Red Hat Linux. The maintainer would very much appreciate examples for Linux distributions that are dissimilar to Red Hat Linux; particularly Debian GNU/Linux and Slackware Linux.

Chapter 3. Preparation

This chapter ensures that access the existing console can be restored should the serial console fail to start.

This chapter then discusses the selection of the RS−232 port and its parameters.

3.1. Create fallback position

Good system administrators always have a viable fallback plan to cope with failures. A mistake configuring the serial console can make both the serial console and the attached monitor and keyboard unusable. A fallback plan is needed to retrieve console access.

Many Linux distributions allow boot diskettes to be created. Writing a boot diskette before altering the console configuration results in a boot diskette that passes good parameters to the kernel rather than parameters that may contain an error.

Under Red Hat Linux a boot diskette is created by determining the running kernel version

```
bash$ uname −r
2.4.2−2
```

and then using that version to create the boot diskette

```
bash# mkbootdisk −−device /dev/fd0 2.4.2−2
```

An alternative fallback position is have a rescue diskette with the machine. A common choice is Tom's root boot.

3.2. Select a serial port

Linux names its serial ports in the UNIX tradition. The first serial port has the file name `/dev/ttyS0`, the second serial port has the file name `/dev/ttyS1`, and so on.

This differs from the IBM PC tradition. The first serial port is named `COM1:`, the second serial port is named `COM2:`, and so on. Up to four serial ports can be present on a IBM PC/AT computer and its successors.

Most boot loaders have yet another naming scheme. The first serial port is numbered `0`, the second serial port is numbered `1`, and so on.

The result is that the first serial port is labeled `COM1:` on the chassis of the IBM PC; is known as `/dev/ttyS0` to Linux; and is known as port `0` to the boot loader.

The examples in this *HOWTO* use this first serial port, as that is the serial port which most readers will wish to use.

Table 3−1. Many names for the same serial port

IBM PC	Linux kernel	Most boot loaders
COM1:	/dev/ttyS0	0
COM2:	/dev/ttyS1	1
COM3:	/dev/ttyS2	2
COM4:	/dev/ttyS3	3

3.3. Select a serial speed and parameters

This *HOWTO* does not discuss the RS−232 standard, which is formally known as *ANSI/TIA/EIA−232−F−1997 Interface Between Data Terminal Equipment and Data Circuit−Terminating Equipment Employing Serial Data Interchange*. For an explanation of bits per second , start bits , data bits , parity and stop bits refer to the *Serial−HOWTO* and the *Modem−HOWTO*.

The Linux kernel uses the syntax in Figure 3−1 to describe the serial parameters. Many boot loaders use a variation of the syntax used by the Linux kernel.

Figure 3−1. Serial parameter syntax, in extended Backus−Naur form

```
<mode> ::= <speed><parity><data><stop>
<speed> ::=  <digits>
<parity> ::= n | e | o
<data> ::= 7 | 8
<stop> ::= 1 | 2
<digits> ::= <digit> | <digit><digits>
<digit> ::= 0 | 1 | & | 9
```

The variables and their values are:

<*speed*>

The speed of the serial link in bits per second.

The Linux kernel on a modern PC supports 50, 75, 110, 134.5, 150, 200, 300, 600, 1200, 1800, 2400, 4800, 9600, 19200, 38400, 57600 and 115200 bits per second. Higher bit rates may be possible depending upon the model of the serial port's semiconductor.

Most boot loaders only support a subset of this range. LILO 21.7.5 supports 110, 150, 300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 56000, 57600 and 115200 bits per second. SYSLINUX 1.67 supports 75 to 56000 bits per second. GRUB 0.90 supports 2400, 4800, 9600, 19200, 38400, 57600 and 115200 bits per second.

You must chose the same speed for both the boot loader and for the Linux kernel. An operating system may use more than one boot loader. For example, Red Hat Linux uses SYSLINUX to install or upgrade the operating system; LILO as the boot loader for Red Hat Linux 7.1 and earlier; and GRUB as the boot loader for Red Hat Linux 7.2 and later.

If you are using a serial terminal or if you are using a dumb modem then the bit rate of the terminal or dumb modem must also match the bit rate selected in the boot loader and kernel.

If the serial console is connected to a Hayes−style modem slower than 9600bps then configure the serial console with the same speed as the modem. Modems faster than 9600bps will generally automatically synchronize to the speed of the serial port.

The selected bit rate must also be supported by the serial port's semiconductor. Early model UARTs such as the 8250 series and the 16450 could only reliably recieve at up to 14400bps. The 16550 series and later models will work at all bit rates.

Unless you have good reason, use the popular bit rate of 9600 bits per second. This is the default bit rate of a great many devices.

The speeds that are supported by the kernel, the three common boot loaders, and all IBM PCs capable of running Linux are: 2400, 4800, 9600 and 19200 bits per second. This is a depressingly small selection: not slow enough to support a call over an international phone circuit and not fast enough to upload large files. You may need to choose a speed that will result in a less robust software configuration.

<parity>

Number of parity bits and the interpretation of a parity bit if one is present.

Allowed values are `n` for no parity bit, `e` for one bit of even parity and `o` for one bit of odd parity.

Using no parity bit and eight data bits is recommended.

If parity is used then even parity is the common choice.

Parity is a simple form of error detection. Modern modems have much better error detection and correction. As a result the parity bit guards only the data on the cable between the modem and the serial port. If this cable has a low error rate, and it should, then the parity bit is not required.

<data>

The number of data bits per character.

Allowed values are `7` bits or `8` bits, as Linux uses the ASCII character set which requires at least seven bits.

Eight data bits are recommended. This allows the link to easily be used for file transfers and allows non−English text to be presented.

<stop>

The number of stop bit−times.[2]

Allowed values are `1` or `2`.

One stop bit−time is recommended.

If the RS−232 cable is very long then two stop bit−times may be needed.

You may occassionally see 1.5 stop bit−times. The intent is to gain 4% more data throughput when a link is too long for one stop bit−time but is too short to require two stop bit−times. 1.5 stop bit−times is now rare enough to be a hazard to use.

Most boot loaders default to `9600n81`. A common default found on older terminals is `9600e71`.

Use `9600n81` if possible, as this is the default for most Linux software and modern devices.

This *HOWTO* always configures the serial speed and parameters, even where not strictly necessary. This is so that people configuring parameters other than the recommended and common default value `9600n81` will know what to alter.

3.4. Configure the modem or the null−modem cable

If a modem is used, configure it to be a dumb modem at the port speed selected in Section 3.3. If the modem accepts Hayes AT commands see Chapter 12 to dumb−down the modem.

Alternatively if a terminal and a null−modem cable are used see Section 11.3, which discusses the pinout of the null modem cable.

3.5. Configure the terminal or the terminal emulator

Configure the terminal to match the serial parameters. The data bits, parity bits and stop bits must match. If a modern smart modem is used then the bit speeds need not match. If a dumb modem or a null modem cable is used then the bit speeds must match.

Set CTS/RTS handshaking on, DTR/DSR handshaking off and XON/XOFF handshaking off. Your equipment may call CTS/RTS handshaking or DTR/DSR handshaking hardware handshaking and may call XON/XOFF handshaking software handshaking .

Set automatic line wrapping on. This allows all of a long console message to be read.

If you are using a terminal emulator then it is best to choose to emulate the popular DEC VT100 terminal. Later terminals in the DEC VT range are compatible with the VT100. If this terminal is not available then try to emulate another terminal that implements *ANSI X3.64−1979 Additional Controls for Use with American National Standard Code for Information Interchange* (or its successor *ISO/IEC 6429:1992 ISO Information technology Control functions for coded character sets*). For example, many emulators have a terminal called `ANSI BBS` which uses the IBM PC character set, IBM PC colors and a selection of *X3.64−1979* control sequences.

See the *Text−Terminal−HOWTO* for much more information on configuring terminals.

Chapter 4. Configure the boot loader

When a PC boots the CPU it runs code from Read−Only Memory. This code is the Basic Input/Output System, or BIOS. The BIOS then loads a boot loader from the Master Boot Record of the first hard disk. In turn, the boot loader reads the operating system into memory and then runs it.

Neither the BIOS nor the boot loader are strictly necessary. For example, there are versions of Linux that run directly from the flash memory which usually contains the BIOS.

The benefits of using a boot loader are:

- Multiple operating systems can be booted. See the *Linux + Windows HOWTO* for more information.
- Parameters can be passed to the kernel interactively. This is useful for solving hardware problems; for example, some interrupt lines can be disabled, direct memory access to some drives can be disabled, and so on. See the *Linux BootPrompt−HOWTO* for a list of kernel parameters.
- Differing kernels can be loaded interactively. This is useful when deploying a new kernel, as it provides a simple fallback to an older proven kernel.

For these reasons systems administrators want to be able to interactively control the boot loader from the serial console.

LILO, GRUB and SYSLINUX are popular boot loaders for IBM PCs. Find which of these boot loaders your Linux installation uses and then follow the instructions for your boot loader in the following section.

4.1. Configure the LILO boot loader

LILO is the Linux Boot Loader used on Intel machines. Other boot loaders for Intel machines exist, common alternatives are GRUB and SYSLINUX. Equivalents to LILO exist for other processor architectures, their names are usually some play upon LILO .

LILO is documented in the *lilo(8)* and *lilo.conf(5)* manual pages; the *LILO Generic boot loader for Linux & User's Guide* found in the file `/usr/share/doc/lilo&/doc/User_Guide.ps`; and the *LILO mini−HOWTO*.

The LILO configuration is kept in the file `/etc/lilo.conf`. The first part of the file applies to all images. The following parts are `image` descriptions for each kernel.

Set LILO to use the serial port. The syntax of the serial line parameters follows that used by the kernel, except that one stop bit is assumed.

Figure 4−1. Syntax of LILO serial command, in EBNF

```
serial=<port>[,<speed>[<parity>[<data>]]]
```

Where the variables are the same as used by the kernel (shown in Figure 3−1) and:

Figure 4−2. LILO serial EBNF variables

<*port*> ::= 0 | 1| & | 3

Our examples use /dev/ttyS0, which LILO knows as port 0.

Figure 4−3. LILO boot loader sample configuration

```
serial=0,9600n8
timeout=100
restricted
password=PASSWORD
```

The parameters restricted and password are used to avoid someone dialing in, booting the machine, and stepping around the Linux access permissions by typing:

Example 4−1. Using kernel parameters to avoid access permissions

```
LILO: linux init=/sbin/sash
```

The password should be good, as it can be used to gain root access. The LILO password is stored in plain text in the configuration file, so it should never be the same as any other password. The permissions on the configuration file should be set so that only root can read /etc/lilo.conf.

```
bash# chmod u=rw,go= /etc/lilo.conf
```

LILO has an option to display a boot message. This does not work with serial consoles. Remove any lines like:

```
message=/boot/message
```

LILO is now configured to use the serial console. The kernels booted from LILO are yet to be configured to use the serial console.

4.2. Configure the GRUB boot loader

GRUB is a boot loader designed to boot a wide range of operating systems from a wide range of filesystems. GRUB is becoming popular due to the increasing number of possible root filesystems that can Linux can reside upon.

GRUB is documented in a GNU info file. Type **info grub** to view the documentation.

The GRUB configuration file is /boot/grub/menu.lst, although some distributions use another configuration file. For example, Red Hat Linux uses the file /boot/grub/grub.conf.

GRUB configuration files are interpreted. Syntax errors will not be detected until the machine is rebooted, so take care not to make typing errors.

Edit the GRUB configuration file and remove any **splashimage** entries. If these entries are not removed GRUB 0.90 behaves very oddly, transferring control between the serial console and the attached monitor and keyboard.

If there is not already a **password** command in the GRUB configuration file then create a hashed password, see configure−boot−loader−grub−md5. The password should be good, as it can be used to gain root access.

Figure 4−4. Using md5crypt to create a hashed password for GRUB

```
grub> md5crypt
Password: **********
Encrypted: $1$U$JK7xFegdxWH6VuppCUSIb.
```

Use that hashed password in the GRUB configuration file, this is shown in Figure 4−5.

Figure 4−5. GRUB configuration to require a password

```
password --md5 $1$U$JK7xFegdxWH6VuppCUSIb.
```

Define the serial port and configure GRUB to use the serial port, as shown in Figure 4−6.

Figure 4−6. GRUB configuration for serial console

```
serial --unit=0 --speed=9600 --word=8 --parity=no --stop=1
terminal serial
```

`--unit` is the number of the serial port, counting from zero, unit 0 being `COM1`.

Note that the values of `--parity` are spelt out in full: `no`, `even` and `odd`. The common abbreviations `n`, `e` and `o` are *not* accepted.

If there is mysteriously no output on the serial port then suspect a syntax error in the **serial** or **terminal** commands.

If you also want to use and attached monitor and keyboard as well as the serial port to control the GRUB boot loader then use the alternative configuration in Figure 4−7.

Figure 4−7. GRUB configuration for serial console and attached monitor and keybaord console

```
password --md5 $1$U$JK7xFegdxWH6VuppCUSIb.
serial --unit=0 --speed=9600 --word=8 --parity=no --stop=1
terminal --timeout=10 serial console
```

When both the serial port and the attached monitor and keyboard are configured they will both ask for a key to be pressed until the timeout expires. If a key is pressed then the boot menu is displayed to that device. The other device sees nothing.

If no key is pressed then the boot menu is displayed on the whichever of `serial` or `console` is listed first in the **terminal** command. After the timeout set by the **timeout** the default option set by **default** is booted.

```
Press any key to continue.
Press any key to continue.
Press any key to continue.
Press any key to continue.
Press any key to continue.
Press any key to continue.
Press any key to continue.
Press any key to continue.
Press any key to continue.
Press any key to continue.

    GRUB  version 0.90  (639K lower / 162752K upper memory)

 +-------------------------------------------------------------------------+
 | [ Red Hat Linux (2.4.9-21)   ]                                          |
 |                                                                         |
 |                                                                         |
 +-------------------------------------------------------------------------+
      Use the ^ and v keys to select which entry is highlighted.
      Press enter to boot the selected OS or 'p' to enter a
      password to unlock the next set of features.

   The highlighted entry will be booted automatically in 10 seconds.
```

Note that there are two timeouts involved. `Press any key to continue` is printed for **terminal −−timeout=10** seconds, waiting for someone on the keyboard or terminal to press a key to get the input focus. Then the menu is displayed for **timeout 10** seconds before the default boot option is taken.

If the terminal attached to the serial port is not a real or emulated VT100, then force GRUB to use it's command line interface. This interface is much more difficult to use than GRUB's menu interface; however, the command line interface does not assume the VT100's terminal language.

Figure 4−8. GRUB configuration for command line interface for terminals other than VT100

```
terminal --timeout=10 --dumb serial console
```

This *HOWTO* does not discuss the use of GRUB's command line. It is far too complex and error−prone to recommend for use on production machines. Wizards will know to consult GRUB's info manual for the commands required to boot the kernel.

GRUB's menu's can be edited interactively after **P** is pressed and the password supplied. A better approach is to add menu items to boot the machine into alternative run levels. A sample configuration showing a menu entry for the default run level and an alternative menu entry for single user mode (run level *s*) is shown in Figure 4−9. Remember to use the **lock** command to require a password for single user mode, as single user mode does not ask for a Linux password.

Figure 4−9. Adding a single user mode option to the GRUB menu

```
password --md5 $1$U$JK7xFegdxWH6VuppCUSIb.
default 0
title Red Hat Linux (2.4.9-21)
        root (hd0,0)
        kernel /vmlinuz-2.4.9-21 ro root=/dev/hda6
        initrd /initrd-2.4.9-21.img
title Red Hat Linux (2.4.9-21) single user mode
        lock
        root (hd0,0)
        kernel /vmlinuz-2.4.9-21 ro root=/dev/hda6 s
        initrd /initrd-2.4.9-21.img
```

File names in the **kernel** and **initrd** commands are relative to the GRUB installation directory, which is usually `/boot/grub`. So `/vmlinuz-2.4.9-21` is actually the file `/boot/grub/vmlinuz-2.4.9-21`.

GRUB is now configured to use the serial console. The kernels booted from GRUB are yet to be configured to use the serial console.

4.3. Configure the SYSLINUX boot loader

SYSLINUX is a boot loader that is installed on a MS−DOS floppy disk. As directed by it's configuration file `\SYSLINUX.CFG` it will load one of the files from the floppy disk as a Linux kernel.

SYSLINUX presents a simple text interface that can be used to select between canned configurations defined in the configuration file and can be used to add parameters to the kernel.

ISOLINUX and PXELINUX are variants of SYSLINUX for CD−ROMs and Intel's Preboot Execution Environment.

SYSLINUX supports a variety of serial port speeds, but it only supports eight data bits, no parity and one stop bit. SYSLINUX supports the serial ports `COM1:` through to `COM4:`, as with most boot loaders these are written as port `0` through to port `3`.

For SYSLINUX to support a serial console add a new *first line* to `\SYSLINUX.CFG`:

Figure 4−10. Syntax of SYSLINUX serial command, in EBNF

```
serial <space> <port> [ <space> <speed> [ <space> <flow_control> ] ]
```

The variables are the same as used by syntax descriptions in Figure 3−1 and Figure 4−2 plus those in Figure 4−11.

Figure 4−11. SYSLINUX serial EBNF variables

```
<space> ::=
<flow_control> ::= <hex_digits>
<hex_digits> ::= 0x<hex_digit><hex_digit><hex_digit>
<hex_digit> ::= 0 | 1 | & | 9 | a | b | & | f
```

The <*flow_control*> variable controlling the RS−232 status and flow control signals is optional. If your null−modem cable does not present any status or handshaking signals then do not use it. The value of <*flow_control*> is calculated by adding the hexadecimal values for the desired flow control behaviours listed in Table 4−1.

The behaviours for a correctly−wired null−modem cable or a correctly configured modem are marked "Required for full RS−232 compliance" in the table. The sum of these values is `0xab3`.

Table 4−1. SYSLINUX flow control bitmap

Flow control behaviour	Hex value	Required for full RS−232 compliance?
Assert DTR	0x001	Yes
Assert RTS	0x002	Yes
Wait for CTS assertion	0x010	Yes
Wait for DSR assertion	0x020	Yes
Wait for RI assertion	0x040	No
Wait for DCD assertion	0x080	Yes
Ignore input unless CTS asserted	0x100	No
Ignore input unless DSR asserted	0x200	Yes
Ignore input unless RI asserted	0x400	No
Ignore input unless DCD asserted	0x800	Yes

Our preferred configuration of 9600bps, port `0`, full RS−232 status signals and CTS/RTS flow control is written as:

```
serial 0 9600 0xab3
```

When using this configuration SYSLINUX will not display anything and will not accept any typed character until the RS−232 status signals show a connected modem call (or a connected terminal if you are using a null−modem cable).

If you have a null modem cable with no RS−232 status signals and no flow control then use:

```
serial 0 9600
```

Remember that the **serial** command must be the first line in `\SYSLINUX.CFG`.

Chapter 5. Configure Linux kernel

The Linux kernel is configured to use a serial console by passing it the `console` parameter. The `console` parameter can be given repeatedly; in that case output is sent to all consoles and input is taken from the last listed console. The last `console` is the one Linux uses as the `/dev/console` device.

The syntax of the `console` parameter is given in Figure 5–1.

Figure 5–1. Kernel `console` syntax, in EBNF

```
console=ttyS<port>[,<mode>]
console=tty<virtual_terminal>
console=lp<parallel_port>
```

<port> is the number of the serial port. This is defined in Figure 4–2 and discussed in Section 3.2. The examples in this *HOWTO* use the first serial port, giving *<port>* the value `ttyS0`.

With no `console` parameter the kernel will use the first virtual terminal, which is `/dev/tty0`. A user at the keyboard uses this virtual terminal by pressing **Ctrl–Alt–F1**.

<mode> is defined in Figure 3–1 and is discussed in Section 3.3. The examples in this *HOWTO* use 9600 bits per second, eight data bits, no parity and one stop bit, giving *<mode>* the value of `9600n81`.

If your computer contains a video card then we suggest that you also configure it as a console. This is done with the kernel parameter `console=tty0`.

For PCs with a video card and a serial console in the port marked COM1: this *HOWTO* suggests the kernel parameters:

Figure 5–2. Recommended kernel parameters, PCs with video card

```
console=tty0 console=ttyS0,9600n81
```

For PCs without a video card, this HOWTO suggests the kernel parameters:

Figure 5–3. Recommended kernel parameters, PCs without video card

```
console=ttyS0,9600n81
```

These parameters are passed to the booting kernel by the boot loader. Next we will configure the boot loader used by your Linux installation to pass the `console` parameters to the kernel.

5.1. Configure Linux kernel using LILO

For each `image` entry in `/etc/lilo.conf` add the line:

Figure 5−4. Recommended kernel parameters, LILO configuration

```
append="console=tty0 console=ttyS0,9600n8"
```

Sometimes the `append` line will already exist. For example

```
append="mem=1024M"
```

In this case, the existing `append` line is modified to pass all the parameters. The result is:

```
append="mem=1024M console=tty0 console=ttyS0,9600n8"
```

As a complete example, a typical `/etc/lilo.conf` configuration from Red Hat Linux 7.1 is:

Example 5−1. Complete LILO configuration, as installed by vendor

```
boot=/dev/hda
map=/boot/map
install=/boot/boot.b
prompt
timeout=50
message=/boot/message
default=linux

image=/boot/vmlinuz-2.4.2-2
  label=linux
  read-only
  root=/dev/hda6
  initrd=/boot/initrd-2.4.2-2.img
```

This is modified to

Example 5−2. Complete LILO configuration, modified for serial console

```
boot=/dev/hda
map=/boot/map
install=/boot/boot.b
prompt
default=linux
# Changes for serial console on COM1: in global section
#   Deleted: message=/boot/message
timeout=200
serial=0,9600n8
timeout=100
restricted
password=de7mGPe3i8

image=/boot/vmlinuz-2.4.2-2
  label=linux
  read-only
  root=/dev/hda6
  initrd=/boot/initrd-2.4.2-2.img
  # Changes for serial console on COM1: in each image section
```

```
  append="console=tty0 console=ttyS0,9600n81"
```

Now that we have finished configuring LILO, use the **lilo** command to install the new boot record onto the disk:

```
bash# chown root:root /etc/lilo.conf
bash# chmod u=rw,g=,o= /etc/lilo.conf
bash# lilo
Added linux *
```

5.2. Configure Linux kernel using GRUB

Find each `title` entry in the GRUB configuration file. It will be followed by a `kernel` line. For example

```
title Red Hat Linux (2.4.9-21)
  root (hd0,0)
  kernel /vmlinuz-2.4.9-21 ro root=/dev/hda6
  initrd /initrd-2.4.9-21.img
```

Modify each of the `kernel` lines to append the parameters that inform the kernel to use a serial console.

Figure 5–5. Recommened kernel parameters, GRUB configuration

```
title Red Hat Linux (2.4.9-21)
  root (hd0,0)
  kernel /vmlinuz-2.4.9-21 ro root=/dev/hda6 console=tty0 console=ttyS0,9600n81
  initrd /initrd-2.4.9-21.img
```

As a complete example, Example 5–3 is a typical GRUB configuration from Red Hat Linux 7.2.

Example 5–3. Complete GRUB configuration, as installed by vendor

```
default=0
timeout=10
splashimage=(hd0,0)/grub/splash.xpm.gz
password --md5 $1$wwmIq64O$2vofKBDL9vZKeJyaKwIeT.
title Red Hat Linux (2.4.9-21)
root (hd0,0)
  kernel /vmlinuz-2.4.9-21 ro root=/dev/hda6
  initrd /initrd-2.4.9-21.img
```

The modified configuration file is shown in Example 5–4.

Example 5–4. Complete GRUB configuration, modified for serial console

```
default=0
timeout=10
password --md5 $1$wwmIq64O$2vofKBDL9vZKeJyaKwIeT.
```

```
serial --unit=0 --speed=9600 -word=8 --parity=no --stop=1
terminal --timeout=10 serial console
title Red Hat Linux (2.4.9-21)
  root (hd0,0)
  kernel /vmlinuz-2.4.9-21 ro root=/dev/hda6 console=tty0 console=ttyS0,9600n81
  initrd /initrd-2.4.9-21.img
title Red Hat Linux (2.4.9-21) single user mode
  lock
  root (hd0,0)
  kernel /vmlinuz-2.4.9-21 ro root=/dev/hda6 console=tty0 console=ttyS0,9600n81 s
  initrd /initrd-2.4.9-21.img
```

5.3. Configure Linux kernel using SYSLINUX

Edit each `LABEL` entry to add an `APPEND` line containing the serial console parameter to pass to the Linux kernel. Like LILO, if a kernel already has parameters, then add our parameters to the list after `APPEND`.

For example:

Figure 5−6. Recommended kernel parameters, SYSLINUX configuration

```
APPEND console=tty0 console=ttyS0,9600n81
```

There are some traps for beginners in the differences between LILO and SYSLINUX. LILO uses `append=`, whereas SYSLINUX uses just `append`. **lilo** needs to be run after each change to `/etc/lilo.conf`, whereas **syslinux** does not need to be run after changing `\SYSLINUX.CFG`.

Chapter 6. Configure getty

getty monitors serial lines, waiting for a connection. It then configures the serial link, sends the contents of `/etc/issue`, and asks the person connecting for their login name. getty then starts login and login asks the person for their password. If the user does nothing, getty or login hang up and getty goes back to waiting.

The getty command has been re–implemented numerous times. There is a wide selection of getty clones, each with slight differences in behavior and syntax. We will describe the traditional getty, and then some popular alternatives.

One of the jobs of a getty is to set the `TERM` environment variable to indicate the make and model of the terminal which is connecting. In this *HOWTO* we set the terminal to the commonly emulated DEC VT100. If you occassionally connect using a different terminal emulation then you can interactively change your choice of terminal by setting `TERM` to the appropiate terminal listed in `/etc/termcap`.

Figure 6–1. Interactively altering the connecting terminal's make and model

```
bash$ TERM=kermit
bash$ tset -r
```

But first, let's see how getty gets started in the first place.

6.1. init system

The file `/etc/inittab` contains the background programs that used to keep the system running. One of these programs is one getty process per serial port.

Figure 6–2. getty is started by init, based upon an entry in `/etc/inittab`

```
s0:2345:respawn:/sbin/getty ttyS0 CON9600
```

Each field in `inittab` is separated by a colon and contains:

s0

Arbitrary entry for `inittab`. As long as this entry doesn't appear anywhere else in `inittab`, you're okay. We named this entry `s0` because it's for `/dev/ttyS0`.

2345

Run levels where this entry gets started. Run levels 2, 3, 4 and 5 can be used for an operational system, getty should not be used in other run levels. The serial console still works in run level 1 (or single user mode) even without a getty.

respawn

Re−run the program if it dies. We want this to happen so that a new `login` prompt will appear when you log out of the console.

/sbin/getty ttyS0 CON9600 vt102

The command to run. In this case, we're telling getty to connect to `/dev/ttyS0` using the settings for `CON9600` which exists in `/etc/gettydefs`. This entry represents a terminal running at 9600bps. Initially assume that the terminal is a later−model VT100.

After changing `/etc/inittab` restart init with

```
telinit q
```

An alternative is to send the hangup signal to init with the command **kill −HUP 1**. This is not recommended: if you make a typing mistake and actually kill init then your system will suddenly halt.

6.2. Traditional getty

Traditional getty implementations include uugetty and getty_ps.

The traditional getty is listed in `/etc/inittab` with the name of a section in `/etc/gettydefs` to use for its configuration. Our example in Figure 6−2 used the section `CON9600`.

There is no `CON9600` in the standard `gettydefs`. This is deliberate, as serial consoles sometimes require slight tweaking. Copy the `DT9600` entry and use it as your model.

Figure 6−3. Define `CON9600` in `gettydefs`

```
# Serial console 9600n81, CTS/RTS flow control
CON9600# B9600 CS8 −PARENB −ISTRIP CRTSCTS HUPCL # B9600 SANE CS8 −PARENB −ISTRIP CRTSCTS HUPCL #@S @L login: #CON9600
```

Separate each line with a blank line.

Each configuration line has the syntax:

Figure 6−4. Syntax of entries in `/etc/gettydefs`, in EBNF

<label># <initial_flags> # <final_flags> #<login_prompt>#<next_label>

The *<label>* is referred to on the getty command line.

The *<next_label>* is the definition used if a RS−232 Break is sent. As the console is always 9600bps, this points back to the original *label*. See Section 9.9 if you ever intend to have more one line for `CON9600` in `gettydefs`.

<*initial_flags*> are the serial line parameters used by getty. These are modeled on the *stty(1)* and *termios(3)* options and the full list varies depending upon your getty variant. The parameters in Figure 6−3 ensure that a line at 9600bps with eight data bits and no parity is configured.

<*final_flags*> are the serial line parameters set by getty before it calls login. You will usually want to set a 9600bps line, `SANE` terminal handling, eight data bits, no parity and to hang up the modem when the login session is finished.

The <*login_prompt*> for serial lines is traditionally the name of the machine, followed by the serial port, followed by `login:` and a space. The macro that inserts the name of the machine and the serial port varies, see the documentation for your getty.

6.3. agetty

agetty is an "alternative getty". It takes all of its parameters on the command line, with no use of `/etc/gettydefs` or any other configuration file. agetty is documented in the manual page *agetty(8)*.

Figure 6−5 shows how to invoke agetty for use with a serial console.

Figure 6−5. `/etc/inittab` entry for agetty

```
s0:2345:respawn:/sbin/agetty -h -t 60 ttyS0 9600 vt102
```

`ttyS0` refers to the serial device `/dev/ttyS0`.

`9600` is the bits per second of the serial link. agetty will support multiple values, using the modem's `CONNECT` message or the RS−232 Break signal to select between them. Only use one value, as serial consoles only have only one data rate.

`vt102` sets the `TERM` environment variable to indicate that a VT100 terminal is connecting.

`-h` activates CTS/RTS handshaking.

`-t 60` allows 60 seconds for someone to attempt to log in before the modem is hung up.

agetty uses escape sequences in `/etc/issue` to insert information. For example, `\n.\o \l` will appear as `remote.example.edu.au ttyS0`.

6.4. mgetty

mgetty is a modem−aware getty. It supports modems with the Hayes AT command set and is especially designed for supporting modems that are used to send faxes and to dial out as well as dial in. These features are not required for a serial console.

mgetty does not require the traditional `/etc/gettydefs` file. As a result mgetty is invoked from `/etc/inittab` without supplying an entry in `/etc/gettydefs`.

Figure 6–6. `/etc/inittab` entry for mgetty

```
s0:2345:respawn:/sbin/mgetty ttyS0
```

mgetty is configured using the file `/etc/mgetty+sendfax/mgetty.config`. It should contain an entry for the port used by the serial console.

Figure 6–7. mgetty configuration file `mgetty.config`

```
port ttyS0
 speed 9600
 direct yes
 data-only yes
 toggle-dtr yes
 need-dsr yes
 port-owner root
 port-group root
 port-mode 600
 login-prompt @ \P login:\040
 login-time 60
 term vt102
```

All the options are documented in the PostScript file `/usr/share/doc/mgetty&/mgetty.ps`.

We set `direct`, `data-only`, `need-dsr` and `toggle-dtr` so that the RS–232 control lines are used correctly for a dumb modem.

`port-owner`, `port-group` and `port-mode` set the serial device to be accessible only by the root user. Modem applications, which normally use the uucp group, cannot now accidentally use the serial console.

`login-prompt` shows the machine (`@`) and serial port (`\P`) being used. The text `\040` is simply the octal code for a space after `login:`.

`term vt102` gives the make and model of the terminal most likely to dial in. This sets the `TERM` environment variable, which you can change if you are dialling in from another terminal type.

The remaining configuration files, `/etc/mgetty+sendfax/dialin.config` and `/etc/mgetty+sendfax/login.config`, do not need to be altered.

If you wish to alter the suggested configuration then note that mgetty's `blocking` and `toggle-dtr` parameters do not co–exist well.

If you have difficulties, activate debugging by adding `debug 8` to `mgetty.config`. mgetty's actions are then visible in the file `/var/log/mgetty.log.ttyS0`.

6.5. mingetty

mingetty is designed to be a minimal getty for the virtual terminals on the the workstation's monitor and keyboard. It has no support for serial lines.

You must not use mingetty for the serial line in `/etc/inittab`, but the current mingetty entries for the virtual terminals can remain.

Each virtual terminal uses about 8KB of kernel memory. If this matters, it is easy to allocate fewer virtual terminals. In the Linux 2.4 kernel virtual terminals are created on demand, so not starting mingetty on the virtual terminal will not create the virtual terminal. If the machine does not have a video card then remove all the mingetty entries from `/etc/inittab`.

Figure 6–8. Fewer virtual terminals. Removing mingetty entries from `/etc/inittab`

```
1:2345:respawn:/sbin/mingetty tty1
# Additional virtual terminals are not used
# 2:2345:respawn:/sbin/mingetty tty2
# 3:2345:respawn:/sbin/mingetty tty3
# 4:2345:respawn:/sbin/mingetty tty4
# 5:2345:respawn:/sbin/mingetty tty5
# 6:2345:respawn:/sbin/mingetty tty6
```

After restarting init it would be wise to remove the unused device files.

Figure 6–9. Fewer virtual terminals. Deallocating unused virtual terminals and removing their device files.

```
bash# telinit q
bash# deallocvt /dev/tty[2-9] /dev/tty[0-9][0-9]
bash# rm /dev/tty[2-9] /dev/tty[0-9][0-9]
```

6.6. No getty

If you are using serial console simply to print console messages then do not run a getty process on the serial port.

getty follows a locking convention that prevents other serial port applications from using the serial port. Since we do not want other processes to use the serial port, but are not running getty, manually create the lock file.

Create a file `/var/lock/LCK..ttyS0` to contain the text `1`. This lets other potential serial port applications know that process 1 has the serial port in use. Process 1 is always the init process, and init is always running, so the serial port is always locked.

The file is created upon each system boot, as lock files are often cleared when the system boots. A convenient place to create the lock file is from `/etc/rc.serial`. It should contain:

Figure 6–10. Contents of `/etc/rc.serial` to lock console serial port when no getty used

```
# Lock /dev/ttyS0 as it is used by an output-only console
(umask 022 && \
 rm -f '/var/lock/LCK..ttyS0' && \
```

```
echo '1' > '/var/lock/LCK..ttyS0')
```

Chapter 7. Configure incidentals

A surprising number of other configuration files need small modifications before the serial console works well.

The configuration of many items depends upon your security requirements, especially depending upon the level of trust and corresponding need for security at the remote site. By assuming a high need for security at the remote site this *HOWTO* can illustrate a large number of configuration items.

7.1. Allow root to login from serial console

The file `/etc/securetty` controls the devices that the root user can log in upon.

It is usually desirable to have root be able to log in from the console, so add the basename of the serial console device to `/etc/securetty`.

Figure 7–1. Alter `securetty` to allow root to log in from the serial console

```
ttyS0
```

Almost anyone can now dial into the modem and attempt to guess the root password. Normally we do not allow root to log in from a remote site, rather we have a normal user log in and then use **su** or **sudo** to become root. This gives some traceability.

Unfortunately, the root user needs to be able to log in from the console to fix a full disk. Disk subsystems typically reserve 5% of their space for root's exclusive use.[3] This is enough space for the root user to log in and start deleting the files that filled the disk.

7.2. Change init level to textual

There is little point in running the X Window System on a server with no screen. Edit `/etc/inittab` finding the line containing `initdefault`, such as

```
id:5:initdefault:
```

Alter the default from run level 5 (multiuser with X Window System) to run level 3 (multiuser).

```
id:3:initdefault:
```

The **startx** command can be used if an occassional X Windows session is required upon an attached keyboard and monitor.

7.3. Remove saved console settings

The file /etc/ioctl.save contains the serial and terminal parameters to use in single user mode. The serial and terminal parameters are usually set by getty during single user mode no getty runs and the contents of /etc/ioctl.save are used to set the serial and terminal parameters.

As we are changing consoles, the saved settings are no longer correct.

Figure 7–2. Removal of ioctl.save containing the saved console parameters

```
bash# rm -f /etc/ioctl.save
```

We re–create this file once we can log in from the serial console.

7.4. Serial console is not /dev/modem

In many Linux distributions the file /dev/modem is a symbolic link to the serial port containing a modem which is available for use.

Although the serial console is a serial port with a modem, we certainly don't want it used to place an outgoing call.

Check that /dev/modem does not point to the serial port being used for the console, say /dev/ttyS0. If it does, then remove the symbolic link.

Figure 7–3. Remove /dev/modem if it points to the serial console's port

```
bash$ ls -l /dev/modem
lrwxrwxrwx 1 root root 10 Jan 01 00:00 /dev/modem -> /dev/ttyS0
bash# rm /dev/modem
```

7.5. Alter target of /dev/systty

In many Linux distributions the file /dev/systty is a symbolic link to the device which is used as the by the attached monitor and keyboard. See Section 2.3 for a fuller description.

If there is no attached keyboard and monitor or no wish to give the attached keyboard and monitor greater capabilities then a text terminal, then alter /dev/systty to point to the serial console.

Rather than directly altering this symbolic link, it is better to modify the configuration file used by **MAKEDEV**, which is then run to recreate the symbolic link. The configuration file is in the directory /etc/makedev.d. The default configuration will point to the first virtual terminal, as shown in Figure 7–4.

Figure 7–4. Default value of /dev/systty in /etc/makedev.d/linux–2.4.x

```
l systty tty0
```

Modify this to point to the serial port being used by the console, as shown in Figure 7−5.

Figure 7−5. Alter value of `/dev/systty` in MAKEDEV configuration file

```
bash# cd /etc/makedev.d
bash# fgrep systty *
linux-2.4.x:l systty tty0
bash# vi linux-2.4.x
```

```
l systty ttyS0
```

Now re−create `/dev/systty` using its new definition, as shown in Figure 7−6.

Figure 7−6. Installing new value of `/dev/systty`

```
bash# cd /dev
bash# rm systty
bash# ./MAKEDEV systty
```

7.6. Configure Pluggable Authentication Modules

The Pluggable Authentication Module system can be used to give special privileges to users that logged in through the console. It is used to make devices like the floppy disk mountable by the console's user; usually they would need to become the super−user to mount a disk.

The PAM configuration file `/etc/security/console.perms` contains the `<console>` variable. For Red Hat Linux 7.1 `<console>` is the regular expression:

Figure 7−7. Default `<console>` in `console.perms` refers to attached keyboard and screen

```
<console>=tty[0-9][0-9]* vc/[0-9][0-9]* :[0-9]\.[0-9] :[0-9]
```

Later in the file the `<console>` user is granted permission to use some devices. This is done by altering the devices' permissions upon login and logout.

Figure 7−8. Default device listing in `console.perms`

```
<console>  0660 <floppy>     0660 root.floppy
<console>  0600 <sound>      0600 root
<console>  0600 <cdrom>      0660 root.disk
<console>  0600 <pilot>      0660 root.uucp
<console>  0600 <jaz>        0660 root.disk
<console>  0600 <zip>        0660 root.disk
<console>  0600 <ls120>      0660 root.disk
<console>  0600 <scanner>    0600 root
<console>  0600 <camera>     0600 root
<console>  0600 <memstick>   0600 root
```

```
<console>  0600 <flash>      0600 root
<console>  0600 <fb>         0600 root
<console>  0600 <kbd>        0600 root
<console>  0600 <joystick>   0600 root
<console>  0600 <v4l>        0600 root
<console>  0700 <gpm>        0700 root
```

There are two types of devices listed above: those devices required by someone connecting from an attached keyboard and monitor and those devices that allow convenient access to devices. The configuration file fails to make the distionction between logical and physical console noted in Section 2.3. The configuration file is modified to create that distinction.

Figure 7–9. Devices in `console.perms` required for attached keyboard and screen

```
<console>  0600 <fb>         0600 root
<console>  0600 <kbd>        0600 root
<console>  0600 <joystick>   0600 root
<console>  0600 <v4l>        0600 root
<console>  0700 <gpm>        0700 root
```

The remaining devices should be altered to give control only to people attaching from the serial console. For example, we don't want an unprivileged user at a co–location site mounting a floppy disk. Define a new console type for the serial console, say `<sconsole>`.

Figure 7–10. Add `<sconsole>` in `console.perms` to refer to serial console

```
<sconsole>=ttyS0
```

Now modify the remaining entries from `<console>` to `<sconsole>`.

Figure 7–11. Remaining devices in `console.perms` altered to refer to serial console

```
<sconsole>  0660 <floppy>     0660 root.floppy
<sconsole>  0600 <sound>      0600 root
<sconsole>  0600 <cdrom>      0660 root.disk
<sconsole>  0600 <pilot>      0660 root.uucp
<sconsole>  0600 <jaz>        0660 root.disk
<sconsole>  0600 <zip>        0660 root.disk
<sconsole>  0600 <ls120>      0660 root.disk
<sconsole>  0600 <scanner>    0600 root
<sconsole>  0600 <camera>     0600 root
<sconsole>  0600 <memstick>   0600 root
<sconsole>  0600 <flash>      0600 root
```

7.7. Configure Red Hat Linux

Red Hat Linux stores parameters concerning system start up in the file `/etc/sysconfig/init`.

Alter the parameter `BOOTUP` to use terminal–independent commands to write the `OK`, `PASSED` and `FAILED` messages. These messages will no longer appear in green, yellow or red. The comments in

/etc/sysconfig/init suggest that any value other than color will do, but it seems that BOOTUP must be set to serial.

Alter the PROMPT parameter to disallow interactive start up. Allowing an unauthenticated keystroke to stop system services is not robust against line noise and allows anyone that dials in during system boot to deny services.

Figure 7−12. Alterations to /etc/sysconfig/init for Red Hat Linux

```
BOOTUP=serial
PROMPT=no
```

Red Hat Linux runs a hardware discoverer, named kudzu. When attempting to identify a serial port Kudzu resets the serial port. This stops the serial console. Kudzu is configured from the file /etc/sysconfig/kudzu.

Kudzu can be prevented from resetting hardware by setting the configuration parameter SAFE to yes.

Figure 7−13. Alterations to /etc/sysconfig/kudzu for Red Hat Linux

```
SAFE=yes
```

Chapter 8. Reboot and test

8.1. Verify console operation

If possible, plug an RS−232 breakout box into the serial port. During reboot the Data Terminal Ready line should become active and then the Transmit Data lights should flash as console messages appear.

Attach a modem, or a null modem cable and a terminal. Configure them to match the serial parameters used by the serial console port. If using a modem, dial in to it from a terminal emulator.

```
+++
AT Z
AT DT 1234−5678
CONNECT 9600
```

Configure the terminal or terminal emulator to match the serial parameters used by the serial console. If using a modern Hayes AT−style modem then the speed need not match. If using a directly−attached terminal then the speed must match.

Reboot the computer.

```
bash# shutdown −h now
```

During reboot the terminal should see the usual boot loader text, and then the default kernel booting, then the init output, and finally the contents of `/etc/issue` and getty asking you to login.

```
LILO:

Linux version &
Kernel command line: auto BOOT_IMAGE=linux ro root=306 BOOT_FILE=/boot/vmlinuz−2.4.3−12 console=tty0 console=ttyS0,9600n81
&
INIT version &
&
/etc/issue says "All your base are belong to us".
remote.example.edu.au ttyS0 login:
```

If you do not see the `login:` message then press **Return** or **Enter**.

8.2. Re−create saved console settings

Log in as root from the serial console and send the console into single user mode. The modem may hang up whilst doing this and you may need to re−connect.

```
remote.example.edu.au ttyS0 login: root
Password: &
```

```
bash# telinit 1
&Telling INIT to go to single user mode.
INIT: Going single user
INIT: Sending processes the TERM signal
sh# rm -f /etc/ioctl.save
sh# stty sane -parenb cs8 crtscts brkint -istrip -ixoff -ixon
```

Exiting from single user mode back to the default run level will save the serial console configuration into `/etc/ioctl.save`.

```
sh# exit
&
bash# ls -l /etc/ioctl.save
-rw------- 1 root root 60 Jan 1 00:00 /etc/ioctl.save
```

This file will be used if the machine boots into single user mode for any reason.

8.3. Test the console

Dial in from a machine, perhaps using Minicom.

Example 8-1. Dialing into a serial console

```
localhost bash$ minicom
```

```
Initializing modem
Welcome to minicom 1.83.1
Press ALT-Z for help on special keys
AT S7=45 S0=0 L1 V1 X4 &C1 E1 Q0
OK
Alt-D remote.example.edu.au-ttyS0
Dialing: remote.example.edu.au-ttyS0 At: 1234-5678
Connected. Press any key to continue
Any
CONNECT 115200/V34/LAPM/V42BIS/33600:TX/33600:RX
```

```
Enter
/etc/issue says "All your base are belong to us".
remote.example.edu.au ttyS0 login: user
Password: ********
Message of the day is "be careful out there".
remote bash$ stty -a
speed 9600 baud; rows 0; columns 0; line = 0;
intr = ^C; quit = ^\; erase = ^?; kill = ^U; eof = ^D; eol = <undef>;
eol2 = <undef>; start = ^Q; stop = ^S; susp = ^Z; rprnt = ^R; werase = ^W;
lnext = ^V; flush = ^O; min = 1; time = 0;
-parenb -parodd cs8 hupcl -cstopb cread -clocal crtscts
-ignbrk brkint ignpar -parmrk -inpck -istrip -inlcr -igncr icrnl -ixon -ixoff
-iuclc -ixany -imaxbel
opost -olcuc -ocrnl onlcr -onocr -onlret -ofill -ofdel nl0 cr0 tab3 bs0 vt0 ff0
isig icanon -iexten echo echoe echok -echonl -noflsh -xcase -tostop -echoprt
-echoctl -echoke
&
remote bash$ logout
```

```
NO CARRIER
Alt-X
Leave Minicom? Yes
Resetting modem
```

```
localhost bash$
```

Interestingly the **stty −a** command, used to display the terminal settings, reports that the link from the modem to the serial console is 9600bps. The `CONNECT` message reports that link between the two modems operates at 33600bps. The constant speed modem−computer link is a very useful feature of the Hayes AT−style modems: the calling computer need not know line speed of the called serial console in advance.

8.4. Where to next from here?

The serial console is now configured. Check the security pointers given in Chapter 9 to complete the job.

Chapter 9. Security

Using serial console with a modem gives anyone the opportunity to connect to the console port. This connection is not mediated by firewalls or intrusion detection sniffers. It is important to prevent the misuse of the serial console by unauthorized people.

9.1. Use good passwords

Anyone that can guess the BIOS password, the boot loader password, or the root password can get full control of the machine. These should be different, unrelated, excellent passwords. Random text and digits are by far the best choice. You should never use a password that you think would return a hit from a search engine.[4]

Guessing a user's password is only slightly less severe. If possible, severely limit the number of users on the machine. Ensure that only good passwords are chosen by using a fascist password checker such as a cracklib−based PAM module.

You should write down the BIOS password, the boot loader password and the root password. Now you don't need to remember them, so there is no reason for them not to be totally random, unrelated, excellent passwords. Fold the page, put it in an envelope and seal it.

Now we have turned a computer security problem into a physical security problem. We know how to solve those problems: locks, keys, alarms, safes, guards, regular inspections. If your site has staffed security then a good option is to leave the envelope in the care of the guard post with instructions to treat the envelope with the same procedures used for the site's master keys. Smaller sites can use a safe, a cash box or a locked drawer. A thief forcing a locked drawer still leaves shows more apparent signs of entry and more clues to their identity than is left by a hacker behind a modem.

These three passwords are an important corporate asset. If the machine is secure then forgetting the major passwords for the machine should result in a machine whose configuration cannot be altered by actions short of disassembly. You should have written procedures controlling the generation, storage, lifetime and use of major passwords.

9.2. Obey Data Terminal Ready and Data Carrier Detect

The RS−232 Data Terminal Ready signal is lowered when the computer wishes the modem to hang up. The computer wishes to hang up when people have ended their login session ends or when they fail to respond to the `login:` prompt.

Using a modem cable that has DTR wired and a modem that is configured to obey DTR is essential to prevent denial of service attacks upon the access to the console.

Without DTR a caller can simply hold the modem line open, denying system administrators access to the console.

The RS−232 Data Carrier Detect signal is lowered when the user hangs up.

Using a modem cable that has DCD wired and a modem that is configured to assert DCD is essential to prevent people dialling in after a user has hang up and from carrying on their session.

Without DCD the session is not cleared when an accidental disconnection occurs. This allows any subsequent caller to resume the previous session. The machine is totally compromised if the previous user was root.

9.3. Use or configure a dumb modem

Most modems use the Hayes AT command set. The modem's attention is gained by sending +++ surrounded by some idle time. Commands are then sent prefixed by `AT`.

Unfortunately, if the modem sees +++ during a call it may revert to command mode. The modem can then be configured by the caller. For example, the modem could be set to permit incoming calls only from the number À , this would deny the system administrators access to the modem.

The attention command can be removed using **AT S2=255**. Of course once that is done no more AT commands can be given to the modem, so any other configuration of the modem needs to be done prior to that command.

Unfortunately, when power to the modem is applied the modem starts in command mode. So a carefully chosen console message could disable the modem.

The best solution is to select a modem that has a dumb or "select profile" DIP switch or jumper. These switches disable command mode and load the modem's saved configuration when they start.

9.4. Restrict console messages

Generating a stready stream of console messages can easily overwhelm a 9600bps link.

Although displaying all syslog messages on the console appears to be a good idea, this actually provides a nice method to deny effective use of the remote console.

Configure log messages to the console to the bare minimum, especially if the machine accepts remotely generated syslog messages. Look in `/etc/syslog.conf` for lines ending with `/dev/console`.

Users that are logged into the serial console should not accept broadcast messages. Add new files to `/etc/profile.d` to do this. Figure 9−1 shows a file for use by the Bourne shell.

Figure 9−1. Restrict sending of messages to console user

```
#
# Do we have files referred to?
if [ -x /usr/bin/mesg -a -x /usr/bin/tty ]
then
  # Are we on serial console?
  if [ `/usr/bin/tty` = /dev/ttyS0 ]
  then
    # Do not accept broadcast messages
```

```
    /usr/bin/mesg n
  fi
fi
```

As this file is run frequently, we use a faster but less readable version of the above, shown in Figure 9−2.

Figure 9−2. Restrict sending of messages to console user, `/etc/profile.d/mesg.sh`

```
#
# /etc/profile.d/mesg.sh -- prevent people hassling the serial console user
[ -x /usr/bin/mesg -a -x /usr/bin/tty -a `/usr/bin/tty` = /dev/ttyS0 ] && /usr/bin/mesg n
```

We also need a C shell version, shown in Figure 9−3.

Figure 9−3. Restrict sending of messages to console user, `/etc/profile.d/mesg.csh`

```
#
# /etc/profile.d/mesg.csh -- prevent people hassling the serial console user
if (-X mesg && -X tty && `tty` == /dev/ttyS0) then
  mesg n
endif
```

Although `mesg.sh` and `mesg.csh` are included by the parent shell rather than executed, the files need the execute permission set. The procedure in Figure 9−4 installs the files and sets the permissions.

Figure 9−4. Install files into `/etc/profile.d`

```
bash# cp mesg.*sh /etc/profile.d/
bash# chown root:root /etc/profile.d/mesg.*sh
bash# chmod u=rwx,g=rx,o=rx /etc/profile.d/mesg.*sh
```

9.5. Modem features to restrict usage

Most modems support the addition of a password. This is not particularly useful as it has the same strenghts and weaknesses of all other password authentication schemes. We already have password authentication in the BIOS, in the boot loader and in login.

Many modems support call−back. The modem is called and a few seconds after hang−up it calls a pre−configured number. This limits the locations that can gain access to the console.

Many modems support checking the calling line identification (CLI) against a predefined list. If the calling number is not on the list then the call is cleared. The phone line to the modem must be configured to send CLI, this may incur an additional charge from the phone company. Not all calling phones can send CLI and some valid callers may have asked their phone company to suppress the sending of CLI.

Many modems can be configured to log the calling line identification. This is useful when tracing misuse.

Many modems support encryption. Some modems allow multiple keys. This gives a neat solution: only authorized people can dial in, but they can do so from any location. The modems usually need to be of the same make, and perhaps of the same model.

Encryption dual−use technology

Possessing, using, buying, selling, importing or exporting modems with encryption features is a serious criminal offense in some countries.

You should acquiant yourself with the laws in your jurisdiction and the laws of jurisdictions you may travel through.

Many telephone services or PBX lines can be set to allow only incoming calls. This is useful as it prevents misuse of the modem should the computer be compromised. A demon dialler can call many numbers seeking an answering modem and the cost of these calls can be significant.

9.6. BIOS features

Most BIOSs can be configured with a configuration password . This should set and tested. Some motherboards will require a jumper to be set to allow the password to take effect. Some BIOSs have well−known master passwords , use a search engine to ensure that your BIOS is not one of these. The password should not be the same as the boot loader or root passwords.

The BIOS configuration will have a boot order setting. It should be set to boot from the hard disk before any other media. This prevents someone inserting a rescue diskette, booting the machine, and gaining access to the filesystems as root.

9.7. Use a boot loader password

Configure the boot loader to request a password when booting a non−default image or when supplying parameters from the command line.

This prevents someone from dialing in during the boot sequence and booting the kernel with options to take control of the machine, as in Example 4−1.

The password should not be the same as the BIOS or root passwords.

9.8. Non−interactive boot sequence

Red Hat Linux has an interactive boot option that can be used to prevent services from being started. This may not be pleasant if the purpose of the machine is web serving and the HTTP daemon is interactively prevented from starting by an unauthenticated person.

Edit `/etc/sysconfig/init` to contain the line

```
PROMPT=no
```

9.9. Magic SysRq key

The magic **SysRq** key is a key sequence that allows some basic commands to be passed directly to the kernel. Kernel software developers use this interface to debug their software. Under most circumstances it can also be used to uncleanly reboot the computer, something that is otherwise difficult or expensive to do remotely.

For computers that are not used for kernel software development the magic **SysRq** key makes an ideal denial of service device. A few unauthenticated keystrokes and the computer is dead in the water. The console, serial or otherwise, must be in a area with access limited to trusted people.

The serial console uses the RS−232 break function as the magic **SysRq** key . A break is a period of no transmission on the serial line, on traditional terminals it is activated by pressing a key labeled **Break**.

Anyone can dial into a modem and send a break, so if the serial console is attached to a modem we need to disable the magic **SysRq** key . If the serial console is attached to a terminal server which asks for authentication, or is attached directly to another terminal using a null modem cable then you may decide to activate the magic **SysRq** key.

The magic **SysRq** key can be disabled by setting a kernel variable or by not compiling support for the key.

Writing a `0` into `/proc/sys/kernel/sysrq` will disable the magic **SysRq** key. The command **sysctl** can also be used:

Figure 9−5. Using sysctl to defeat the magic SysRq key

```
bash# sysctl −w kernel.sysrq=0
```

Your Linux distribution may have a file `/etc/sysctl.conf` which is used to run **sysctl** during the boot of the machine. Add the lines:

Figure 9−6. Configuring `/etc/sysctl.conf` to defeat the magic SysRq key

```
# Disables the magic SysRq key
kernel.sysrq = 0
```

Even when setting the magic **SysRq** key off in `/etc/sysctl.conf` there is a period of vulnerability after the kernel boots but before contents of the file are applied.

It is much better to compile your own kernel and set the following configuration parameter:

Figure 9−7. Kernel make menuconfig showing disabled SysRq key

```
Kernel hacking  --->
 [ ] Magic SysRq key
```

This should place the following configuration parameter in `/usr/src/linux/.config`.

Figure 9−8. Kernel `.config` showing disabled SysRq key

```
# CONFIG_MAGIC_SYSRQ is not set
```

9.10. Adjust behaviour of Ctrl−Alt−Delete

The IBM PC used **Ctrl−Alt−Delete** to launch a reboot of the computer. Linux traps this key chord and makes it available to the init system. This is done by sending the init process a `SIGINT` signal (although **ctrlaltdel hard** can undo this trap and make the key chord reboot the comptuer immediately). The init system uses `/etc/inittab` to determine how to handle the signal generated by the **Ctrl−Alt−Delete** key chord.

Most distributions cleanly reboot the system, mimicing the behaviour that most users expect. Figure 9−9 shows how this is done.

Figure 9−9. Default handling of Ctrl−Alt−Delete in `/etc/inittab`

```
# Trap CTRL-ALT-DELETE
ca::ctrlaltdel:/sbin/shutdown -t3 -r now
```

Depending upon each individual site you may wish to disable **Ctrl−Alt−Delete**. This is shown in Figure 9−10.

Figure 9−10. Ignoring Ctrl−Alt−Delete in `/etc/inittab`

```
# Trap CTRL-ALT-DELETE and do nothing
ca::ctrlaltdel:
```

Alternatively, you may wish to cleanly shut down the computer. This is very easy to explain to operators and instructions can be displayed on the monitor using `/etc/issue` or a Post−it Note. If the computer uses Advanced Power Management (or APM) then shutting down the computer will also remove the power.

Figure 9−11. Shut down cleanly upon Ctrl−Alt−Delete in `/etc/inittab`

```
# Trap CTRL-ALT-DELETE and shut down
ca::ctrlaltdel:/sbin/shutdown -t3 -h now
```

9.11. Log attempted access

Look in the system logs for the output of getty. Add the monitoring of these messages to your log−watching procedures.

Chapter 10. Configuring a kernel to support serial console

Most Linux kernels shipped by distributors are configured to allow the serial console to be enabled. However system administrators will almost certainly encounter some problems best solved by recompiling a kernel. In these cases configure the kernel to support the serial console. The usual virtual terminal console is also configured, as we normally want console messages to go a monitor as well as the serial port.

10.1. Linux kernel version 2.4

When configuring the kernel set the following configuration parameters:

Figure 10–1. Kernel configuration for serial console using make menuconfig

```
Character devices  --->
 [*] Virtual terminal
  [*]   Support for console on virtual terminal
 <*> Standard/generic (8250/16550 and compatible UARTs) serial support
  [*]   Support for console on serial port
```

This should set the following configuration parameters in `/usr/src/linux/.config`.

Figure 10–2. Kernel configuration for serial console using `.config`

```
CONFIG_VT=y
CONFIG_VT_CONSOLE=y
CONFIG_SERIAL=y
CONFIG_SERIAL_CONSOLE=y
```

You should also configure the kernel without the magic **SysRq** key, as described in Section 9.9.

10.2. Linux kernel version 2.2

The later Linux 2.2 kernels use the same build parameters and parameter syntax as the Linux version 2.4 kernels.

For earlier kernels see the article by Francesco Conti in issue 36 of *Linux Journal* published in April 1997.

This article included some patches for the kernel, which have been extended in the notes below to use a broader range of serial port speeds.

Choose to use the serial console by adding a couple of `#defines` at the start of `/usr/src/linux/drivers/char/console.c`:

```
#define CONFIG_SERIAL_ECHO
#define SERIAL_ECHO_PORT 0x3f8  /* COM1 port address */
```

Alternatively, to use `ttyS1` use these lines:

```
#define CONFIG_SERIAL_ECHO
#define SERIAL_ECHO_PORT 0x2f8  /* COM2 port address */
```

The kernel assumes a serial link speed of 9600bps. If you are using a differing bit rate then find these two lines:

```
serial_echo_outb(0x00, UART_DLM); /* 9600 baud */
serial_echo_outb(0x0c, UART_DLL);
```

and change `0x0c` to one of the values in Table 10−1.

Table 10−1. IBM−PC/AT serial port bit rates and their bit−clock divisors

Bit Rate	Divisor
115200bps	`0x01`
57600bps	`0x02`
38400bps	`0x03`
19200bps	`0x06`
9600bps	`0x0c`
4800bps	`0x18`
2400bps	`0x30`
1200bps	`0x60`

Chapter 11. Serial cabling

11.1. Jargon

RS−232 cables were originally intended to link terminals to modems. The terminal is formally named a Data Terminal Equipment, abbreviated to DTE. The modem is formally named a Data Communications Equipment, abbreviated to DCE.

A standard RS−232 cable as a 25−pin D−type socket which connects to the DTE and a 25−pin D−type plug which connects to the DCE. All 25 pins are connected, with pin 1 on the plug wired to pin 1 on the socket, pin 2 on the plug wired to pin 2 on the socket, and so on. The shielding of the cable is attached to the metallic cover on the socket.

RS−232 signaling is much more robust than many other communications standards. Pins can be shorted, not connected or drive more than one output.

Signals are named from the point of view of the Data Terminal Equipment. So Transmit Data on the DTE is connected to Transmit Data on the DCE. Transmit Data on the DTE actually transmits data, whereas Transmit Data on the DCE actually recieves data.

11.2. Cable from console port to modem

The RS−232 standard defines the interconnection of computers and modems, so there is little to go wrong here by simply purchasing a pre−assembled cable. There are two types of cable: cables with connectors for a standard 25−pin D connector on the computer; and cables with connectors for a proprietary 9−pin D connector used on the IBM PC/AT and many other computers. The cables have titles like *RS−232 25−pin computer (DTE) to 25−pin modem (DCE)* or *RS−232 9−pin IBM PC/AT computer (DTE) to 25−pin modem (DCE)*. Most modems are packaged with a suitable cable.

If you need to manufacture your own cables, see the *Serial−HOWTO* for the RS−232 pinout for your computer. Connect Transmit Data on the computer to Transmit Data on the modem, Receive Data on the computer to Receive Data on the modem, and so on for Signal Ground, Clear to Send, Ready to Send, Data Set Ready, Data Terminal Ready, Data Carrier Detect and Ring Indication.

For professional computer room installations consider routing the serial cable through an RJ−45 patch panel. There are two common pinouts on used on the RJ−45 connector: Yost and Cisco 2500−series console.

If you create your own pinout for unshielded twisted pair cable then be sure that your pinout twists a Signal Ground wire with the Transmit Data wire and another Signal Ground wire with the Receive Data wire. Although the RS−232 signals are not balanced, this twist will result in the least amount of signal degradation and noise pickup.

11.3. Cable from console port to terminal (or another PC)

The RS−232 standard allows for, but does not specify, the interconnection of two computers without intervening modems. A special cable is required, called a null modem cable.

The wiring within the null modem cable depends upon the handshaking and control signals that are needed. Differing manufacturers have differing views on this topic, so don't buy a null modem cable that does not come with a wiring diagram.

Linux needs all of the flow control and modem control signals to be correctly wired. The correct wiring of a null modem cable is shown in Figure 11−1.

Linux uses CTS and RTS to do handshaking, preventing the computer from overrunning the terminal and preventing the terminal from overrunning the computer. If you are connecting two computers together, then you will not get reliable file transfers without CTS/RTS handshaking.

Linux uses DSR and DCD to sense that a terminal is connected. It will then request a login. If a session is established and DCD falls then Linux will log out the user.

Linux uses DTR to force the link to be cleared. It does this after a user logs off to free up the communications channel.

Major security exposures can occur with incorrectly wired null modem cables.

Figure 11−1. Null modem cable with full status and handshaking

```
      Signal ground ---------------------- Signal ground

       Receive data ---------------------- Transmit data

      Transmit data ---------------------- Receive data

      Ready to send ---------------------- Clear to send

      Clear to send ---------------------- Ready to send

Data terminal ready -----------------+---- Data carrier detect
                                     |
                                     +---- Data set ready

Data carrier detect ----+----------------- Data terminal ready
                        |
     Data set ready ----+

    Ring indication -- not connected

                          not connected -- Ring indication
```

Unfortunately not all Linux boot loaders support the control signals required by the Linux operating system. This odd state of affairs may force you to do away with control signals and handshaking if you need to issue commands to the boot loader.

There are two ways of defeating the RS−232 handshaking: software and hardware.

If you have a modem then by far the best technique is to disable the control signals and handshaking by using AT commands to configure the modem's software. This allows the handshaking to be restored when the boot loader authors correct their support for serial terminals.

For a null modem cable the best approach is to disable handshaking in your terminal emulation software.

In the worst case for a null modem you will need a cable that falsifies the handshaking and control signals.

Figure 11−2. Null modem cable with falsified status and handshaking

```
      Signal ground ---------------------- Signal ground

       Receive data ---------------------- Transmit data

      Transmit data ---------------------- Receive data

Data terminal ready ---+              +--- Data terminal ready
                       |              |
      Clear to send ---+              +--- Clear to send
                       |              |
Data carrier detect ---+              +--- Data terminal ready
                       |              |
     Data set ready ---+              +--- Data set ready

      Ready to send -- not connected

                          not connected -- Ready to send

    Ring indication -- not connected

                          not connected -- Ring indication
```

If you are happy with a quick hack, perhaps just to use a serial console to grab a kernel oops message, then you can configure some getty programs to ignore the RS−232 status signals. For example, mgetty has the direct option in `mgetty.conf`. In this case only a three−wire RS−232 null modem cable is needed.

Figure 11−3. Null modem cable with no status or handshaking

```
Signal ground ---------------------- Signal ground

 Receive data ---------------------- Transmit data

Transmit data ---------------------- Receive data
```

Don't use this cable in a production environment.

11.4. Making serial cables

If you use a serial console for densely−racked computers you will end up making a lot of null−modem serial cables. This section has some hints on making serial cables. If you are making more than ten cables and live in a city you will probably find it economic to have the cables made by a specialty cabling firm.

The RS−232 standard will drive at least 15 meters of shielded cable. Longer distances are possible with better cable; 100 meter cables are advertised by some specialty firms. Distances longer than 15m are also be possible with the high−quality unshielded twisted pair used for 100Base−TX ethernet. Be wary of long unshielded cables, as the RS−232 signals are not balanced and thus pick up noise easily. For distances beyond 100m use an RS−232 line driver; these will typically drive up to 2000 meters over category 3 UTP cable. For greater distances consider using fiber optical modems, the global telephony system, the mobile telephony system, satellite or radio.

If the environment has a lot of radio frequency noise then use shielded cable and connectors. Connect the shield in the cable to the computer at *one* end. This can be done by connecting the drain wire of the shield it to the Protective Ground (if present) or by soldering the drain wire to the body of the connector. If there is a substantial amount of noise also place a ferrite core over the shielded cable at both ends of the cable. Follow the usual good practices of making the cable to the correct length and screwing home the D connectors into the chassis.

If you are making one of these cables and have some soldering skill, you can easily do the jumpering of the signal wires within the backshell of the DB9 or DB25 connector.

If you are making a large number of cables then crimping systems are much faster than soldering. Again, pin jumpering can be done within the backshell.

No matter what system is adopted use the Resistance setting of a multimeter to check for dead and shorted pins. A minute here can save hours later.

For structured cabling systems, space is tight within DB9/RJ−45 backshells, so the jumpering is better done behind the patch panel. The DB9/RJ−45 connectors present the IBM PC pinout at the DB9 connector and present the Yost or Cisco pinout at the RJ−45 connector.

Incompatible devices in structured cabling systems

Take care to connect only RS−232 devices to RS−232 devices when patching structured cabling systems. Other cables may be carrying ethernet, ISDN, telephony, alarm and DC power voltages. Connecting incompatible voltages may destroy equipment.

Chapter 12. Modem configuration

12.1. Using Minicom to give commands to a modem

Minicom is a full–screen serial terminal emulation package, very much like the classic Telix terminal emulator for MS–DOS.

Firstly, start Minicom in configuration mode with the command:

```
bash# minicom -o -s
```

The following menu appears:

```
Filenames and paths
File transfer protocols
Serial port setup
Modem and dialing
Screen and keyboard
Save setup as dfl
Save setup as..
Exit
Exit from Minicom
```

Select Serial port setup and set

```
A - Serial Device: /dev/ttyS0
B - Lockfile Location: /var/lock
C - Callin Program:
D - Callout Program:
E - Bps/Par/Bits: 9600 8N1
F - Hardware Flow Control: Yes
G - Software Flow Control: No
```

Now save the configuration

```
Give name to save this configuration?
> console
```

and exit Minicom.

To configure a modem use the command **minicom –o console** to start Minicom without sending an initialization string to the modem. Now issue the `AT` commands to configure the modem.

When finished use the Quit option to leave Minicom without sending a reset string to the modem; this option is **Alt–Q**.

Sometimes Minicom will use **Ctrl–A** rather than **Alt–Z** to access the menu system, look for a hint in Minicom's start up message:

Press ALT−Z for help on special keys
Press CTRL−A Z for help on special keys

12.2. Configure dumb modem

Linux, like most UNIX−like operating systems, expects a serial console to be connected to a dumb modem. Dumb modems are not seen much these days, perhaps only on exotic hardware such as ISDN terminal adapters or satellite ground terminals.

A dumb modem is configured using hardware. Figure 12−1 shows the front panel of a fanciful dumb modem. In reality the speed and mode settings are likely to be done using jumpers or DIP switches.

Figure 12−1. Front panel of a dumb modem

```
+-----------------------------+
|                             |
|    SPEED       MODE         |
|  [ ]  300    [ ] Originate  |
|  [ ]  600    [X] Answer     |
|  [ ] 2400                   |
|  [X] 9600                   |
|                             |
+-----------------------------+
```

The modem's speed is set the the desired bit rate, in our case 9600bps. The modem's mode is set to Answer, that is, to wait for incoming calls and to answer them.

If the RS−232 control line Data Terminal Ready is low, the modem will not answer a call. The computer is off or the computer's serial interface is not yet initialized. Once DTR is high the modem will answer incoming calls.

Once an incoming call is established the modem raises the Data Carrier Detect control line.

getty on the Linux computer has been waiting for DCD to come high, and getty welcomes the user and requests them to log in.

Whilst the user is logged in and data is flowing, Clear to Send and Ready to Send are used between the modem and the computer to prevent data being sent too soon. The computer lowers Clear to Send when it is too busy to receive a character. The modem lowers Ready to Send when it is too busy to receive a character.

When the user hangs up, Data Carrier Detect falls and the hang up signal is sent to all processes associated with the dial in session.

Alternatively, the user can log out. When the shell dies, the computer pulls Data Terminal Ready low, causing the modem to hang up. When the getty brings Data Terminal Ready high again, the modem will accept more incoming calls.

We have not yet described Data Set Ready. This line is low if the modem is off or if the modem has not yet initialized.

12.3. Configure modem with AT commands

Most modems today are smart modems based upon the Hayes modems and their command sets. But as discussed above, the remote serial console is designed to operate with a dumb modem.

Thus the smart modem is dumbed−down until it resembles a dumb modem. Some expensive modems will have a DIP switch or board jumper to put them into dumb mode.

It is essential to have a manual for the modem which describes that modem's `AT` commands. Although most modems agree on the more popular `AT` commands, they differ in the more technical commands.

12.3.1. Configure port speed

Hayes AT−style modems can maintain a static speed between the computer and the modem, no matter what speed the dialing modem uses.

For most modems this is set automatically based upon the speed of the first characters sent after power−on.

Power cycle the modem and connect to it with the command **minicom −o console**. Press **Enter** a few times. The modem should now be running at the same bit rate used by Minicom, which we set to the speed of the serial console in Section 12.1.

You can check the port speed by asking the modem to generate some output.

Figure 12−2. Testing the modem's port speed

```
bash# minicom −o console
Welcome to minicom
Press CTRL−A Z for help on special keys

Enter Enter Enter

ATI Enter
56k V.90 Series 3 External V2.20

Ctrl−A Q
Leave without reset? Yes
```

Some modems have an AT command to re−establish the port speed, look in your modem's manual for the **AT&B1** command. Some modems have a command to explicitly set the port speed, look in you modem's manual for the **ATB** command.

12.3.2. Configure answer mode

The modem will answer an incoming call on the second ring using the command **ATS0=2**.

Don't answer the phone on the first ring as this may invalidate the certification of the modem in some telephony jurisdictions.

12.3.3. Configure CTS/RTS handshaking

CTS/RTS handshaking prevents lost characters.

The `AT` command is **AT&K3**.

12.3.4. Configure Data Carrier Detect

Data Carrier Detect should follow the presence or absence of a calling modem.

The AT command is **AT&C1**.

12.3.5. Configure Data Terminal Ready

Data Terminal Ready should control the modem. If DTR is high the modem is ready to receive calls. If DTR is low the modem should not receive any More calls and should hang up any existing call.

The AT command is **AT&D2**.

12.3.6. Configure no `CONNECT` messages

An Hayes AT–style modem usually outputs a message when a call is received. For example:

```
CONNECT 9600
```

The modem has a quiet mode that disables these messages.

The AT command is **ATQ1**. There will be no `OK` printed in response to this command.

12.3.7. Configure no echo of commands

Echoing commands can confuse the console.

The `AT` command is **ATE0**.

12.3.8. Configure silent connection

Most modems have a speaker. By default this is connected whilst a modem is connecting and negotiating a common protocol and speed. This is very useful for a dialing modem, as it prevents a human being accidentally repeatedly called. The speaker can be annoying on answering modems.

If wanted, use the **ATM0** command to turn off the speaker.

12.3.9. Configure DTR delay

Data Terminal Ready drops when the semiconductor that supports the RS−232 link is reset. This then hangs up the modem. This can be annoying. If the getty supports a parameter similar to mgetty's `toggle−dtr−waittime` then it is possible to extend the time that the modem will ignore DTR. The time that getty holds DTR low to force a hang up is extended beyond the modem's setting. The result is that resetting the semiconductor does not hang up the modem, but getty can still hang up the modem at the end of a login session.

Check your modem's documentation. Our example modem uses S−register 25 to contain the threshold for noticing a change in DTR. The value is in one−hundreds of a second. By setting the modem with **ATS25=150** (1.5 seconds) and setting mgetty with `toggle−dtr−waittime 2000` (2 seconds) we ignore small blips in DTR.

12.3.10. Configure no attention sequence

Once the modem is correctly configured and works well, disable the **+++** sequence that gives access to the modem's command mode.

The AT command is **ATS2=255**.

If this command is accidentally given see Section 12.3.12 to reset the modem to its factory default parameters and start again.

12.3.11. Configuration example

Figure 12−3. Configure modem using AT commands

```
bash# minicom −o console
Welcome to minicom
Press CTRL−A Z for help on special keys

AT &F Enter
OK

AT Z Enter
OK

AT &C1 &D2 &K3 S0=2 M0 Enter
```

```
OK

AT E0 Q1 S2=255 &W Enter

Alt-A Q
Leave without reset? Yes
```

12.3.12. Resetting the modem

If you need to issue more AT commands to the modem then power cycle the modem. This should place the modem into command mode.

Now issue the following commands to restore the modem's factory configuration.

Figure 12−4. Resetting a Hayes AT−style modem

```
bash# minicom -o console
Welcome to minicom
Press CTRL-A Z for help on special keys

AT &F &Y0 &W &W1 Enter
OK
AT Z Enter
OK

Alt-A Q
Leave without reset? Yes
```

If this fails then you will need to clear the modem's configuration memory. The procedure for this varies by manufacturer, and probably requires the disassembly of the modem.

12.4. Internal modems

An internal modem is basically an external modem and serial port mounted upon a PC bus card. These are cheaper than external modems as they do not require a power supply or a chassis.

Internal modems work fine for remote serial console applications. They are especially attractive for computers at co−location sites, as those site charge according to space and power consumption.

Check that your internal modem preserves its setting across a power cycle.

Ensure that the interrupt line and port address space used by the internal modem's serial port does not conflict with that used by any other pre−existing serial ports. Alternatively, ensure that the internal serial port can be disabled, freeing its interrupt line and port address space for use by the internal modem.

Be careful not to confuse an internal modem with a WinModem. An internal modem does not need a special device driver, but appears to Linux as a stardard serial port.

12.5. WinModems

If you look at a modem, with it's small central processing unit and special−purpose digital signal processor, and then look at a modern PC, with it's large CPU and general−purpose DSP on the sound card, you may wonder if the hardware duplication of an external modem is necessary.

A WinModem incorporates the CPU and DSP of the modem into the slightly−enhanced fabric of a PC. They are called "WinModems" because they originally only shipped with Microsoft Windows device drivers. These device drivers presented the illusion of a serial port attached to a Hayes AT−style modem. For a long time only Windows versions of these drivers where available. Some manufacturers now provide Linux versions of their device drivers as well, these are jokingly called LinModems .

It is probably possible to use a LinModem as a Linux console. At the most this would require altering the source code to dumb−down the AT command emulation of the modem and recompiling the kernel.

Boot loaders, however, work in a very confined software environment and struggle to support a simple serial chip. Considering that some boot loaders do not even handle interrupts, handling the complex DSP of a LinModem is well beyond what is practical.

Appendix A. Bugs and annoyances

A.1. Red Hat Linux 7.1 and SysVinit

The System V init system shipped with Red Hat Linux 7.1 does not support serial console correctly in single user mode. See Red Hat advisory *RHBA−2001:085−02 New SysVinit package to fix hangs on serial console*. The advisory announces an update to the package `SysVinit−2.78−15.i386.rpm` that is shipped on the Red Hat Linux 7.1 CD.

A.2. BIOSs, keyboards and video cards

Some BIOSs will not boot if the keyboard is not installed.

```
Keyboard faulty, press F1
```

Most BIOSs have settings that will allow them to boot without a keybaord.

Some odd BIOSs will not boot if no video card is installed.

A.3. Modem hangs up upon reboot

During reboot the serial controller is reset. This drops the modem control line Data Terminal Ready. This in turn instructs the modem to hang up.

Avoid the temptation to configure the modem to ignore DTR. This leads to a worse bug, where the telephone line does not clear down correctly, the modem is engaged, and there is no way to clear it. Ignoring DTR also gives no way to clear hostile callers from the line.

You may wish to record the amount of time that the computer takes from `Restarting system` to the boot loader prompt.

The modem may also hang up during the boot process (as the serial chip is reset) or when the init run level is changed (as getty is restarted).

A.4. init and syslog output does not display on secondary consoles

The kernel can be configured to output messages to the serial port and to the attached monitor. However messages from init and syslog only appear on the last−listed console device, in our case the serial port.

This can confuse someone looking at the attached monitor, as the messages on the monitor suggest that the machine has hung just before starting init. Eventually the machine will finish booting and getty will display a `login:` request. A Post−it Note on the monitor may reassure the impatient.

A.5. The console is unresponsive after connecting

The terminal's screen may be blank after connecting to the machine. Pressing **Enter** will usually bring up a `login:` request.

If no characters appear upon the screen after pressing **Enter** do not panic. The machine must have power and the operating system must have booted: for our call to be answered by the modem Data Terminal Ready must be active.

The most likely thing is that the machine booted and is running a **fsck** filesystem check. These checks can take some considerable time, all with no or very little output.

It will help your peace of mind considerably to record in the system log book the time **fsck** takes to check each filesystem.

If you see garbled text after pressing **Enter** then there are mismatched bit rates or parity parameters. Correct your terminal emulator's configuration.

A.6. Modem hangs up during initialization

Using **setserial** will reset the serial port. This will hang up the modem.

setserial is sometimes used during the boot process, resulting in the output seen in Figure A−1. Look into the file `/etc/rc.serial` and remove any references to the port which is being used as the serial console.

Figure A−1. setserial causes a modem to hang up as the machine initializes

```
&
Mounting local filesystems:  [  OK  ]
Turning on user and group quotas for local filesystems:  [  OK  ]
Enabling swap space:  [  OK  ]
/dev/ttyS0 at 0x03f8 (irq = 4) is a 16550A

NO CARRIER
```

A.7. Boot loader has no flow control

Most boot loaders do not support CTS/RTS flow control. This can cause some data loss where large speed mis−matches exist, as is often the case with a modern modem connected into a 9600bps fixed−speed port.

SYSLINUX 1.66 supports flow control.

A.8. Boot loaders are vulnerable to line noise

Most boot loaders will sit at their prompt forever after receiving a single character of line noise.

Some modems will let the RS−232 signals "float", sending noise when their is no caller. Because the modem is not asserting Data Carrier Detect it expects the receiver to discard the noise characters.

The combination of an unfortunate boot loader with an unfortunate modem can result in a machine that will regularly hang during booting.

If you cannot configure your boot loader to obey DCD then be careful to test any modem you intend to purchase to ensure that it does not generate characters when their is no caller. At the present only SYSLINUX implements full RS−232 status signals.

A.9. Advanced Power Management

APM allows control of the power from software. This can be a blessing and a curse.

The blessing is that the machine can be cleanly and totally shut down remotely. You may want to do this if the remote site is maintaining their power supply.

The curse is that once powered down the machine will not start up again until the **Power** button is physically pressed. Some machines have a BIOS or motherboard setting to defeat this unhelpful behaviour.

Errors when typing shutdown are worse with APM

Be careful not to confuse **shutdown −r now**, which cleanly reboots the machine, with **shutdown −h now**, which cleanly powers down the machine. Someone will need to physically press the **Power** button if you choose wrongly.

If you are serious about remote site computing then you should investigate remote power switches from companies like Western Telematic, Server Technology and many others. Some models include built−in terminal servers, built−in modems and RS−232 lines to simulate a UPS input power failure (and thus shut the Linux system down cleanly before removing power).

A.10. Modems and overseas telecommunications requirements

There is no world−wide approval processes to certify that a modem is suitable for connection to the telephone network. This is despite the presence of a common set of technical standards that modems must meet for use on the global switched telephone network. There is little or no recognition of one nation's approvals by other national regulators.

There are national technical requirements concerning the use of modems. Common requirements are to set the modem and its software to answer after the second ring and never to dial the same engaged or faulty number more than five times in a row.

Telecommunications device approvals

Using or importing unapproved telecommications equipment is a criminal offense in most countries.

Additionally, the operator of some types of equipment may require certification.

Privacy laws may control what can be done with calling line identification records.

Do not assume that Touch Tone dialling is globally available. There is no common standard for decadic dialling: some countries have the longest sequence for zero, other countries have the shortest sequence for zero.

There is little coordination of national numbering plans. Be careful not to call a national emergency services number when intending to dial the international access code. Common emergency services numbers are: 112, 911, 000. International access codes vary by country.

Intelligent network features such as toll−free numbers are usually not available to calls originating from abroad.

International calls may be routed through fiber optical submarine cable, satelite or High Frequency radio. The possible bit rates vary considerably between these options. Expect the maximum throughput with no errors from fiber optical submarine cable. Expect 1200bps to 2400bps with some errors from satelite. Expect 75bps to 300bps with many errors from HF radio.

There will be considerable latency depending upon the distance. If the latency becomes greater than the modem's error correction window then you will get better Zmodem file transfer performance if you disable the HDLC−based error correction in the modems.

International voice calls may be placed through analogue conditioning circuits or may be digitally compressed. These alterations to the modem's signals considerably lower modem throughput, if a connection can be established at all. Most international calling cards use digital compression. You may be able to program a *guard tone* to disable analogue conditioning, this will vary by carrier and the commands to send the guard tone vary by modem.

Appendix B. Uploading files from a serial console

There are many scenarios where the machine is dead in the water and you need to upload a file to correct that. In many of these scenarios the only way to upload the file is via the serial port being used as the console.

Moving files about over serial links has a long history in microcomputing and this section goes back in time to uncover the tools commonly used in the pre−Internet age of the Bulletin Board System.

B.1. ASCII upload and cat

cat is available on every UNIX−like system. It copies the data received from the keyboard to a file. Minicom and other terminal emulators have an ASCII upload facility that will send a file up the serial link as though it had been typed.

```
remote bash$ cat > upload.txt
```

```
Alt-S Upload ascii
[ascii upload - Press CTRL-C to quit]
```

`Wait for upload to complete&`

```
ASCII upload of "upload.txt"
10.0 Kbytes transferred at 3900 CPS... Done.
READY: press any key to continue...
```

```
Ctrl-D
remote bash$
```

Without hardware flow control ASCII upload will drop the occassional character.

To upload binary files encode them into ASCII, upload them, and then decode them into binary again.

```
localhost bash$ uuencode upload.bin < upload.bin > upload.txt
```

```
Alt-S Upload ascii
[ascii upload - Press CTRL-C to quit]
```

`Wait for upload to complete&`

```
ASCII upload of "upload.txt"
10.0 Kbytes transferred at 3900 CPS... Done.
READY: press any key to continue...
```

```
Ctrl−D
remote bash$
```

```
remote bash$ uudecode < upload.txt
```

You can detect transmission errors by using a checksum program such as **sum**, **cksum** or **md5sum**. Print the ckecksum of the file before it is sent from the local machine and after it is recieved upon the remote machine.

```
localhost bash$ cksum upload.bin
1719761190 76 upload.bin
```

```
remote bash$ cksum upload.bin
1719761190 76 upload.bin
```

There are a number of checksumming programs. The **sum** command should be used with caution, as there are versions for BSD and System V UNIX which give differing results. **cksum** is the attempt by the POSIX standards developers to correct that mess: it gives the same result for the same file on all POSIX machines.

If the checksums of the original and uploaded files do not match then the file will have to be uploaded again. If the link is noisy and the file is big then you may never get a successful upload. What is needed in this case is to divide the file into many small parts, upload a part, check its checksum, and if it is fine proceed to the next part.

This sounds like something that should be automated. Entering from stage left is Xmodem.

B.2. Disable logging to console

Console messages will disturb two−way communications protocols such as Xmodem, Zmodem or Kermit. The protocols are robust enough to cope with the occassional message, they are just another type of line noise. But regularly repeated messages, such as those from a failing disk, will diminish performance to the extent that uploading a file will be impossible.

Alter your system's configuration to give klogd the `−c 1` parameter, inhibiting the display of kernel messages directly to the console. Kernel messages will still go to the system logger.

Figure B−1. Supressing kernel messages to the console in Red Hat Linux

```
bash# vi /etc/sysconfig/syslog
```

```
KLOGD_OPTIONS="−2 −c 1"
```

```
bash# /etc/init.d/syslog restart
```

Also modify the system logger's configuration not to send messages to the console. Edit `/etc/syslog.conf`, altering lines sending output to `/dev/console`. Send this output to a file instead.

B.3. Xmodem, Ymodem and Zmodem

Xmodem sends 128 bytes and a checksum, waits for a Acknowledgment to say all is well and sends the next block. If a negative acknowledgement is received or if no ACK or NAK ever appears then the block is sent again.

Xmodem is a simple protocol, as you would expect of a program written for 8−bit computers runing CP/M. It has lots of inefficiencies and minor problems, such as rounding up the file size to the next 128 byte boundary. These deficiencies lead to an evolution of the protocol with revisions of Xmodem, then Ymodem and finishing with Zmodem. Zmodem is substantially faster than Xmodem and has no niggling problems. The Zmodem protocol is substantially more complex than the Xmodem protocol, but since we only seek to at most compile the code, that complexity need not concern us.

```
remote bash$ rz
... waiting to receive.**B0100000023be50
```

```
Alt-S Upload zmodem
[zmodem upload - Press CTRL-C to quit]
Sending: upload.bin
Bytes Sent:   3072/  10000   BPS:2185     ETA 00:09
```

If an upload fails and you are left with **rz** waiting to recieve a file then typing **Ctrl−X** a number of times will return you to the command prompt. This also works for Xmodem's **rx** and Ymodem's **ry**.

A useful Zmodem capability is the ability to resume failed uploads and to send multiple files in a single upload session.

An implementation of Xmodem, Ymodem and Zmodem for POSIX computers is available from http://www.ohse.de/uwe/software/lrzsz.html. Red Hat Linux distribute this in the `lrzsz` RPM package. lrzsz is a enhanced free software branch of the public domain version of rzsz from Omen Technology.

B.4. Kermit

Kermit is a terminal emulator and file transfer program delevoped by Columbia University. It's popularity springs from the large range of computers that Kermit could be used to access, from IBM mainframes to MS−DOS PCs.

A Kermit variant named G−Kermit was released under the*GNU Public License*. This is available in most Linux distributions.

The recent Kermit and Zmodem protocols are built upon the same technologies. Zmodem has better performance in calls with high error rates. Kermit has been ported to more host platforms.

Appendix C. Upgrading Red Hat Linux from a serial console

Upgrades to Linux distributions are frequently released. A machine is not remotely manageable unless these upgrades can be installed without needing to physically touch the machine.

This section examines the remote installation and remote upgrade of Red Hat Linux.

Red Hat Linux can be installed over the network from a HTTP server using an install diskette. We modify this diskette to use the serial console. If we can control whether to boot from this diskette or from the hard disk then we can remotely upgrade the Red Hat Linux distribution from the serial port. If a blank diskette is placed in the drive when the machine is deployed then no on−site intervention is needed to upgrade the operating system.

If you have upgrade procedures for other Linux distributions please contribute them to the *HOWTO* maintainer.

C.1. Select boot disk

The key to a remote upgrade is to be able to boot from floppy disk to perform the upgrade, and then to reboot from the hard disk. The possibilities are:

1. Most BIOSs allow the boot disk order to be controlled through the BIOS' configuration. If the BIOS supports a serial console then the machine can be upgraded whilst leaving the floppy disk in the drive. No one need attend the site to upgrade the operating system
2. Someone can insert a floppy disk before the upgrade and remove it afterwards. Most co−location sites will provide this level of board−swap technical support.
3. Two records of the CMOS memory which stores the BIOS configuration can be made: one for booting from floppy and another for booting from hard disk. Unfortunately the nvram device driver does not yet work on a wide enough variety of machines for this HOWTO to pursue this option further.

C.2. Configure the BIOS to use the serial port

Many servers allow the BIOS to be configured from the serial port, especially on systems designed for rack mounting. At the moment few machines designed to be used as desktop systems allow the BIOS to be accessed from the serial port.

Refer to your vendor's documentation to set the BIOS to use the serial port. Some vendors call this feature console redirection . Unfortunately, the meaning of this term varies by vendor. Some vendors use it to mean the redirection of the VGA output and keyboard to a remote PC using a proprietary serial protocol. This feature can only be used in conjunction with the Linux serial console if the BIOS can be instructed to disable the serial redirection after booting.

As an example of the confusion, Dell uses console redirection when describing the Dell 2400 and the Dell 2450. The Dell 2450 BIOS can be configured from the serial port. The Dell 2400's console redirection is additional hardware that remotely replicates the computer's VGA monitor and keyboard.

An example of a BIOS configuration is given in Figure C−1.

Figure C−1. Configuring BIOS to use serial link

```
BIOS setup console redirection

Enter BIOS setup during boot when
  Keyboard:     [Ctrl+Alt+Esc pressed]
  Serial port:  ["HAL" is typed]

Serial port
  Port:         [COM1]
  Speed         [9600] bps
  Data:         [8] bits
  Parity:       [None]
  Stop:         [1] bits
  Handshaking:  [Full CTS/RTS handshaking]
  Terminal:     [Dumb]
```

Many BIOSs will enter their configuration dialogs if a particular terminal key is pressed during the BIOS boot. This can be a problem if the modem link is noisy.

For normal operation, set the boot order to attempt to boot from the hard disk first.

Figure C−2. Configuring BIOS to boot from hard disk

```
BIOS setup boot order

First:  [Hard disk]
Second: [CD−ROM]
Third:  [Floppy disk]
```

C.3. Configure modem to ignore DTR and assert DCD

We will be doing a fair about of rebooting and having this hang up the modem is annoying, so modify the modem's configuration to ignore Data Terminal Ready. Use the **AT&D0** to ignore the status of DTR.

We may also wish to disconnect during the install to reduce transmission charges. Configuring the modem to hold Data Carrier Detect on will prevent any disconnection and reconnection from being apparent to the installer. Use the command **AT&C0** to always hold DCD high.

Apply these changes using the procedure in Section 12.3, retaining all of the other `AT` commands.

C.4. Prepare a network install floppy diskette

The Red Hat Linux web site has a floppy diskette image for a network installation. For Red Hat Linux 7.1 the image is
`ftp://ftp.redhat.com/pub/redhat/linux/7.1/en/os/i386/images/bootnet.img`.

Install this image on a floppy disk.

```
bash# mkfs −t msdos −c /dev/fd0
```

```
mkfs.msdos 2.2 (06 Jul 1999)
bash# dd if=bootnet.img of=/dev/fd0 bs=1440k
1+0 records in
1+0 records out
bash# sync
```

Now mount the diskette and check that the installer files are present.

```
bash# mount -t vfat /dev/fd0 /mnt/floppy
bash# ls /mnt/floppy
boot.msg    general.msg  ldlinux.sys  rescue.msg    vmlinuz
expert.msg  initrd.img   param.msg    syslinux.cfg
```

This floppy disk uses the SYSLINUX boot loader which was discussed in Section 4.3 and in Section 5.3. Firstly, we alter the boot loader configuration file `/mnt/floppy/syslinux.cfg` to use the serial port. If you are going to use the vi editor to alter this file, use the `-n` option to avoid writing a swap file to the floppy disk.

```
bash# vi -n /mnt/floppy/syslinux.cfg
```

```
serial 0 9600
```

Secondly we add a new boot option. This is modeled upon the other boot options in the file. Our variant passes the serial console parameters to the kernel, the same parameters that we pass during normal operation when using serial console. "serial" seems an appropriate name for the boot option.

```
label serial
 kernel vmlinuz
 append initrd=initrd.img lang= text serial expert devfs=nomount console=ttyS0,9600n81
```

`text`, `serial` and `expert` are parameters to the Red Hat anaconda installer. Specifying `text` ensures that the graphical installer does not start. Specifying `serial` prevents scans for possibly non−existent video hardware. You will need to run Xconfigurator manually if you do have a video card. Specifying `expert` allows all the configuration options to be seen, giving one floppy image that can be used for all purposes.

Thirdly, we make this new configuration start automatically. As there is no−one at the site, there's no need to issue a `boot:` prompt.

```
default serial
prompt 0
```

Fourthy, we write the new configuration to diskette.

```
bash# umount /mnt/floppy
```

Check that the diskette boots. If it does not then write a new boot sector by downloading and running the most recent SYSLINUX.

```
bash# syslinux /dev/fd0
```

Finally, create a new boot image for copying to the computers to be upgraded.

```
bash# dd if=/dev/fd0 of=bootserialnet.img bs=1440k
1+0 records in
1+0 records out
```

If you test the new boot floppy on a machine with a serial console you should briefly see SYSLINUX booting

```
SYSLINUX 1.52 2001-02-07  Copyright (C) 1994-2001 H. Peter Anvin
```

and then presenting the `boot.msg` file and then the Linux kernel should be loaded

```
Loading initrd.img..............
Loading vmlinuz............. ready.
```

and run.

```
Linux version 2.4.2-2BOOT (root@porky.devel.redhat.com) (gcc version 2.96 200001
```

Next the init system flashes by

```
Greetings.
Red Hat install init version 7.0 starting
mounting /proc filesystem... done
mounting /dev/pts (unix98 pty) filesystem... done
Red Hat install init version 7.0 using a serial console
remember, cereal is an important part of a nutritionally balanced breakfast.
checking for NFS root filesystem...no
trying to remount root filesystem read write... done
checking for writeable /tmp... yes
running install...
running /sbin/loader
```

before the installation application, called anaconda, is started

```
Welcome to Red Hat Linux
+----------+ Devices +-----------+
|                                |
| Do you have a driver disk?     |
|                                |
|    +-----+          +----+     |
|    | Yes |          | No |     |
|    +-----+          +----+     |
|                                |
|                                |
+--------------------------------+
```

```
<Tab>/<Alt-Tab> between elements   | <Space> selects | <F12> next screen
```

There does not seem to be a way to access the function keys, fortunately the user interface does not require their use.

Now that the floppy has been tested, eject the disk and reboot the machine into normal operation.

C.5. Prepare HTTP server

It is best if the web server runs the version of Red Hat Linux as is being upgraded to. If it runs an earlier version, then do not rebuild the operating system on this machine and install anaconda−runtime from the later operating system.

Copy the Linux distribution to a local web server using a mirroring utility like **wget**. Alternatively the files can be copied from the distribution CDs to the web server.

```
bash$ mkdir --mode=664 --parents /var/www/html/redhat/linux/7.1/en/os/i386
bash$ umask 002
bash$ wget -nh -nH -r -N -nr -l0 -k -np -X SRPMS ftp://ftp.redhat.com/pub/redhat/linux/7.1/en/os/i386/ -P /var/www/html/redhat/linux/7.1/en/os/i386
```

It's best to use a mirror site in place of Red Hat's FTP site used in the example above.

It is very important not to gain files along the way. Delete any files generated by FTP servers, web servers and CD−ROMs.

```
bash$ cd /var/www/html/redhat
bash$ # Files added by FTP server
bash$ find . -name '.listing' -print -exec rm {} \;
bash$ find . -name 'ls-*' -print -exec rm {} \;
bash$ # Files added by a wget from a HTTP server
bash$ find . -name '\?*' -print -exec rm {} \;
bash$ # Files added by a CD-ROM
bash$ find . -name 'TRANS.TBL' -print -exec rm {} \;
```

We now need to add the latest updates to the distributed software. This is done to avoid the machine being compromised immediately following the upgrade.

Adding the updates is essential for Red Hat Linux 7.1, see Section A.1.

Collect together the updates RPMs from `ftp://ftp.redhat.com/pub/updates/7.1/en/os/` in the subdirectories `i386`, `i486`, `i586` `i686`, `images` and `noarch`.

Merge these updates into the copy of the distribution. For each updated RPM file, remove the original RPM file then replace it with the updated RPM file. For example:

```
bash$ cd /var/www/html/redhat/linux/7.1/en/os/i386/RedHat/RPMS
bash$ ls /var/www/html/redhat/updates/7.1/en/os/i386
```

```
SysVinit-2.78-17.i386.rpm
bash$ ls SysVinit-*.rpm
SysVinit-2.78-15.i386.rpm
bash$ rm SysVinit-2.78-15.i386.rpm
bash$ cp /var/www/html/redhat/updates/7.1/en/os/i386/SysVinit-2.78-17.i386.rpm .
bash$ chmod u=rw,g=r,o=r SysVinit-2.78-17.i386.rpm
```

Merge the RPMs from the `updates` subdirectories `i386`, `i686` and `noarch` into `/var/www/html/redhat/linux/7.1/en/os/i386/RedHat/RPMS`. Merge the files from the directory `/var/www/html/redhat/updates/7.1/en/os/images` into the directory `/var/www/html/redhat/linux/7.1/en/os/i386/images`.

The file `/var/www/html/redhat/linux/7.1/en/os/i386/RedHat/base/hdlist` and `hdlist2` contain the list of the RPMs to install. This needs to be modified to contain the names of the updated RPMs.

Install the `anaconda-runtime` RPM on the HTTP server. This RPM should be the same version as the Red Hat Linux being upgraded to.

Now create a new `hdlist` with the commands:

```
bash$ cd /usr/lib/anaconda-runtime
bash$ rm /var/www/html/redhat/linux/7.1/en/os/i386/RedHat/base/hdlist*
bash$ umask 002
bash$ ./genhdlist --withnumbers --hdlist /var/www/html/redhat/linux/7.1/en/os/i386/RedHat/base/hdlist /var/www/html/redhat/linux/7.1/en/os/i386
```

The distribution plus the updates can now be used for a network install. They cannot be used for a CD install, but that doesn't concern us.

As the distribution plus the updates is different from the original distribution, we should not use the version number of the original distribution. Appending the date to which the updates have been applied seems best.

```
bash$ cd /var/www/html/redhat/linux/
bash$ mv 7.1 7.1-20020202
```

C.6. Record network configuration

If the machine does not use the Dynamic Host Configuration Protocol then record the current network configuration. This is used to complete the installer's Configure TCP/IP screen.

Example C−1. Displaying the Internet Protocol configuration

```
bash$ ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 00:11:22:33:44:55
          inet addr:10.1.2.3  Bcast:10.1.2.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:344233 errors:0 dropped:0 overruns:0 frame:0
          TX packets:285750 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100
```

```
         Interrupt:10 Base address:0x9000
bash$ netstat -r -n                                                            ❶
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
10.1.2.0        0.0.0.0         255.255.255.0   U        40 0          0 eth0
127.0.0.0       0.0.0.0         255.0.0.0       U        40 0          0 lo
0.0.0.0         10.1.2.254      0.0.0.0         UG       40 0          0 eth0
bash$ cat /etc/resolv.conf
domain example.edu.au
nameserver 10.255.1.1
nameserver 10.255.2.1
nameserver 172.16.1.1
```

❶ The value of `inet addr` is the IP address. Our example shows `10.1.2.3`. The value of `Mask` is the Netmask. Our example shows `255.255.255.0`.

❷ The value in the Gateway column for Destination `0.0.0.0` is the Default gateway. Our example shows `10.1.2.254`.

❸ The value of the first listed `nameserver` is the Primary nameserver. Our example shows `10.255.1.1`.

C.7. Record LILO configuration

Record the current value of `append=`, `boot=` and `linear` in `/etc/lilo.conf`.

Example C−2. Displaying the LILO configuration

```
bash# fgrep append= /etc/lilo.conf
append="console=tty0 console=ttyS0,9600n81"
bash# fgrep boot= /etc/lilo.conf
boot=/dev/hda
bash# fgrep linear /etc/lilo.conf
bash#
```

If the `boot=` parameter points to a hard disk then LILO is installed in the master boot record, or MBR. It can also point to a partition.

If the `linear` parameter is present then the hard disk that is booted from uses linear block addressing, or LBA.

C.8. Upgrade Red Hat distribution

In this section it all comes together. We will walk through an entire serial console upgrade, not that it differs much from a standard text mode upgrade.

Configure BIOS to boot from floppy or insert the floppy disk. Now reboot the machine.

```
bash# shutdown −r now
```

```
SYSLINUX 1.64 1.64-pre2   Copyright (C) 1994-2001 H. Peter Anvin
                Welcome to Red Hat Linux 7.1!
 -  To install or upgrade Red Hat Linux in graphical mode,
    press the <ENTER> key.
 -  To install or upgrade Red Hat Linux in text mode, type: text <ENTER>.
 -  To enable low resolution mode, type: lowres <ENTER>.
    Press <F2> for more information about low resolution mode.
 -  To disable framebuffer mode, type: nofb <ENTER>.
    Press <F2> for more information about disabling framebuffer mode.
 -  To enable expert mode, type: expert <ENTER>.
    Press <F3> for more information about expert mode.
 -  To enable rescue mode, type: linux rescue <ENTER>.
    Press <F5> for more information about rescue mode.
 -  If you have a driver disk, type: linux dd <ENTER>.
 -  Use the function keys listed below for more information.
[F1-Main] [F2-General] [F3-Expert] [F4-Kernel] [F5-Rescue]
boot:
Loading initrd.img..............
Loading vmlinuz............. ready.
Linux version 2.4.2-2BOOT (root@porky.devel.redhat.com) (gcc version 2.96 20000731 (Red Hat Linux 7.1 2.96-79)) #1 Sun Apr 8 18:24:33 EDT 2001
```

Because we have booted into expert mode, the menus differ slightly from the standard upgrade. For example, you probably don't have a driver disk.

```
 Welcome to Red Hat Linux
+----------+ Devices +-----------+
|                                |
| Do you have a driver disk?     |
|                                |
|    +-----+          +----+     |
|    | Yes |          |[No]|     |
|    +-----+          +----+     |
|                                |
+--------------------------------+
```

The upgrade then continues in the usual fashion.

```
+--------+ Choose a Language +---------+
|                                      |
| What language should be used during  |
| the installation process?            |
|                                      |
|            Czech       :             |
|          [ English     : ]           |
|            Danish      :             |
|            French      :             |
|            German      :             |
|            Hungarian   :             |
|            Icelandic   :             |
|            Italian     :             |
|                                      |
|               +----+                 |
|               |[OK]|                 |
|               +----+                 |
|                                      |
+--------------------------------------+
```

Select HTTP to upgrade from the web server we prepared previously.

```
+-----+ Installation Method +------+
|                                  |
| What type of media contains the  |
| packages to be installed?        |
|                                  |
|            NFS image             |
|            FTP                   |
|          [ HTTP ]                |
|                                  |
|     +----+         +------+      |
|     |[OK]|         | Back |      |
|     +----+         +------+      |
|                                  |
+----------------------------------+
```

Here we supply the network details recorded in Example C−1. If your network supports Dynamic Host Configuration Protocol or the Bootstrap Protocol then these work fine too.

```
+--------------------+ Configure TCP/IP +--------------------+
|                                                            |
| Please enter the IP configuration for this machine. Each   |
| item should be entered as an IP address in dotted-decimal  |
| notation (for example, 1.2.3.4).                           |
|                                                            |
|       [ ] Use dynamic IP configuration (BOOTP/DHCP)        |
|                                                            |
|           IP address:           10.1.2.3________           |
|           Netmask:              255.255.255.0___           |
|           Default gateway (IP): 10.1.2.254______           |
|           Primary nameserver:   10.255.1.1______           |
|                                                            |
|            +----+                      +------+            |
|            |[OK]|                      | Back |            |
|            +----+                      +------+            |
|                                                            |
+------------------------------------------------------------+
```

Provide the name of the pre−prepared web server. Note that the response to Red Hat directory must start with a `/`.

```
+-----------------+ HTTP Setup +----------------------------------+
|                                                                 |
| Please enter the following information:                         |
|                                                                 |
|     o the name or IP number of your web server                  |
|     o the directory on that server containing                   |
|       Red Hat Linux for your architecure                        |
|                                                                 |
|   Web site name:     www.example.edu.au_______________________  |
|   Red Hat directory: /redhat/linux/7.1-20020202/en/os/i386____  |
|                                                                 |
|         +----+                 +------+                         |
|         |[OK]|                 | Back |                         |
```

```
|         +----+                 +------+                           |
|                                                                   |
+-------------------------------------------------------------------+
```

The following status messages then fly by before the welcome screen appears.

```
Retrieving base/netstg1.img...
Loading /mnt/runtime ramdisk...
Retrieving base/netstg2.img...
Loading /mnt/runtime/usr ramdisk...
Running anaconda - please wait...
Graphical installation not available for http installs.  Starting text mode.
```

```
+----------------+ Red Hat Linux +-----------------+
|                                                  |
| Welcome to Red Hat Linux!                        |
|                                                  |
| This installation process is outlined in detail  |
| in the Official Red Hat Linux Installation       |
| Guide available from Red Hat Software. If you    |
| have access to this manual, you should read the  |
| installation section before continuing.          |
|                                                  |
| If you have purchased Official Red Hat Linux,    |
| be sure to register your purchase through our    |
| web site, http://www.redhat.com/.                |
|                                                  |
|         +----+                 +------+          |
|         |[OK]|                 | Back |          |
|         +----+                 +------+          |
|                                                  |
+--------------------------------------------------+
```

Select Upgrade Existing Installation, although this procedure works fine for installations as well.

```
+--------------+ Installation Type +--------------+
|                                                 |
| What type of system would you like to install?  |
|                                                 |
|          Workstation                            |
|          Server System                          |
|          Laptop                                 |
|          Custom System                          |
|        [ Upgrade Existing Installation ]        |
|                                                 |
|         +----+                +------+          |
|         | OK |                | Back |          |
|         +----+                +------+          |
|                                                 |
+-------------------------------------------------+
```

The upgrade continues. When the LILO Configuration screen appears insert the kernel parameters recorded from Example C–2. These parameters should include `console=ttyS&`.

```
+---------------------+ LILO Configuration +---------------------+
```

```
|                                                                 |
| A few systems will need to pass special options to the kernel   |
| at boot time for the system to function properly. If you need   |
| to pass boot options to the kernel, enter them now. If you      |
| don't need any or aren't sure, leave this blank.                |
|                                                                 |
|       [ ] Use linear mode (needed for some SCSI drives)         |
|                                                                 |
|        console=tty0 console=ttyS0,9600n81______________         |
|                                                                 |
|       +----+              +------+             +------+         |
|       | OK |              | Skip |             | Back |         |
|       +----+              +------+             +------+         |
|                                                                 |
+-----------------------------------------------------------------+
```

```
+-------------+ LILO Configuration +--------------+
|                                                  |
|  Where do you want to install the bootloader?    |
|                                                  |
|[/dev/hda         Master Boot Record (MBR)      ]|
| /dev/hda1        First sector of boot partition |
|                                                  |
|         +----+               +------+            |
|         | OK |               | Back |            |
|         +----+               +------+            |
|                                                  |
+--------------------------------------------------+
```

```
+----------------------+ LILO Configuration +-----------------------+
|                                                                    |
| The boot manager Red Hat uses can boot other operating systems     |
| as well. You need to tell me what partitions you would like to     |
| be able to boot and what label you want to use for each of them.   |
|                                                                    |
| Device      Partition type           Default Boot label            |
|[/dev/hda6   Linux Native              *       linux     ] :        |
|                                                           :        |
|                                                           :        |
|                                                           :        |
|                                                           :        |
|                                                                    |
|        +----+              +------+              +------+          |
|        | Ok |              | Edit |              | Back |          |
|        +----+              +------+              +------+          |
|                                                                    |
|                                                                    |
+--------------------------------------------------------------------+
```

The upgrade continues. As installing the packages may take a few hours, you can disconnect.

```
+-------------+ Package Installation +--------------+
|                                                    |
| Name   :                                           |
| Size   :                                           |
| Summary:                                           |
```

```
|                                                    |
|                  Packages       Bytes        Time |
| Total    :              0         0M             |
| Completed:              0         0M             |
| Remaining:              0         0M             |
|                                                    |
|                                                    |
+----------------------------------------------------+
```

If you disconnected, then when reconnecting it is best to press **Tab** rather than pressing **Return**.

Pressing **Return** on the Bootdisk screen writes a boot disk. This will overwrite the upgrade disk.

You may wish to deliberately create a boot disk if you cannot alter the BIOS parameters to boot from the hard disk, or if you cannot wait for someone to eject the floppy disk before rebooting.

```
+------------------+ Bootdisk +-------------------+
|                                                  |
| A custom boot disk provides a way of booting     |
| into your Linux system without depending on      |
| the normal bootloader. This is useful if you     |
| don't want to install lilo on your system,       |
| another operating system removes lilo, or lilo   |
| doesn't work with your hardware configuration.   |
| A custom boot disk can also be used with the     |
| Red Hat rescue image, making it much easier to   |
| recover from severe system failures.             |
|                                                  |
| Would you like to create a boot disk for your    |
| system?                                          |
|                                                  |
|        +-----+                 +----+            |
|        |[Yes]|                 | No |            |
|        +-----+                 +----+            |
|                                                  |
+--------------------------------------------------+
```

When the Complete screen appears prepare to reboot into Linux. If you have a serial BIOS be prepared to alter the BIOS parameters to boot from the hard disk first. If you do not have a serial BIOS ask someone to eject the floppy disk.

```
+-----------------+ Complete +------------------+
|                                                |
| Congratulations, installation is complete.  # |
|                                             : |
| Press return to reboot, and be sure to      : |
| remove your boot medium after the system    : |
| reboots, or your system will rerun the      : |
| install. For information on fixes which     : |
| are available for this release of Red Hat   : |
| Linux, consult the Errata available from    : |
| http://www.redhat.com/errata.               : |
|                                             : |
| Information on configuring and using your   : |
| Red Hat Linux system is contained in the    : |
|                                                |
```

```
|                    +----+                   |
|                    |[OK]|                   |
|                    +----+                   |
|                                             |
+---------------------------------------------+
```

```
sending termination signals...done
sending kill signals...done
disabling swap...
        /tmp/swap/hda5
unmounting filesystems...
        /mnt/sysimage/var/www/html
        /mnt/sysimage/boot
        /mnt/sysimage/proc
        /mnt/runtime/usr
        /mnt/sysimage
        /proc/bus/usb
        /mnt/runtime
        /dev/pts
        /proc
rebooting system
Restarting system.

LILO
Loading linux.....................
Linux version 2.4.3-12 (root@porky.devel.redhat.com) (gcc version 2.96 20000731 (Red Hat Linux 7.1 2.96-85)) #1 Fri Jun 8 15:05:56 EDT 2001
```

C.9. Create boot disk for serial console

Once the upgrade has been sucessfully done create a boot floppy which has serial console support. This is most simply done by creating a boot disk, as done by the anaconda installer or as described in Section 3.1; modifying the configuration file `\SYSLINUX.CFG` to configure the boot loader to use the serial console, as described in Section 4.3; and finally configuring the kernel to use the serial console, as described in Section 5.3.

An alternative is to create your own mkbootdisk RPM package containing a modified copy of the shell script `/sbin/mkbootdisk`.

The `\SYSLINUX.CFG` file on the boot floppy is written by **mkbootdisk** using the code in Figure C−3. We alter this code to use the serial console; the result is shown in Figure C−4.

Figure C−3. Extract from Red Hat Linux 7.2 `mkbootdisk` which creates `SYSLINUX.CFG`

```
cat > $MOUNTDIR/syslinux.cfg <<EOF
default linux
prompt 1
display boot.msg
timeout 100
label linux
  kernel vmlinuz
  append $INITRDARG root=$rootdev
EOF
```

Figure C−4. Altered extract from `mkbootdisk`, which creates a `SYSLINUX.CFG` that uses a serial

console

```
cat > $MOUNTDIR/syslinux.cfg <<EOF
serial 0 9600
default linux
prompt 1
display boot.msg
timeout 100
label linux
  kernel vmlinuz
  append $INITRDARG root=$rootdev console=tty0 console=ttyS0,9600n81
EOF
```

Created boot floppies will now use the serial console.

By far the best alternative would be the addition of parameters to **mkbootdisk** to allow the kernel parameters and serial port, speed and flow control to be given when the boot floppy is created. For this enhancement request see Red Hat Bugzilla entry 59351.

C.10. Further references

Sometimes the kernel on the installation CD won't boot on the machine to be upgraded, or the filesystem requires modules that are not present. In this case you will need to build a new kernel and rebuild the installation disk to use the new kernel. This is documented in the *RedHat7 CDs mini−HowTo*. This is an informal HOWTO not available through the Linux Documentation Project.

An older document that more fully describes an older Red Hat distribution build process is *Burning a RedHat CD HOWTO*.

Appendix D. Terminal server configuration

Terminal servers were originally designed for connecting terminals to minicomputers. Each terminal would have an RS−232 port. The connection to the minicomputer usually used an ethernet port. Connecting terminals would be connected to a command line interface where they could select from a list of predefined machines. A Telnet session would then be started to that machine.

Over time terminal servers gained more features. For example, modems could be connected. These initially allowed people to dial in to the minicomputer but grew in features until most terminal servers became routers with a great number of serial ports.

As well as allowing the connection of many console to a single terminal, the terminal server can be configured with user accounts and passwords, preventing unauthenticated access to the console whilst still allowing the console to be reached from any modem.

Internet Service Providers have been large users of terminal servers in the past. Each modem would be connected to a terminal server port and incoming users would be permitted to send IP packets anywhere, not just to some predefined minicomputer. Manufacturers renamed the equipment to access server or modem server to reflect this new use.

These devices have been superseded by a new generation of access server that allows telephone trunks to be plugged directly into the ISP's router. There are no discreet modems; the modem tones are decoded by digital signal processing chips within the router.

As a result terminal servers are currently readily available on the second−hand market.

Most old terminal servers will not support Secure Shell. In this is the case accessing the terminal server by its ethernet port is a poor idea: when you login to the console you password will travel across the Internet in clear text. Either dial in to the terminal server or use a one−time password system such as the RADIUS protocol with S/Key authentication.

An alternative to using a terminal server is to use a multiport serial card in another Linux system.

This remainder of this section lists the cabling pinouts and basic software configuration needed for differing types of terminal servers.

Further contributions are welcome and should be e−mailed to the maintainer of this *HOWTO*.

D.1. Cisco 2511

Before purchasing a Cisco router on the second−hand market ensure that the seller has the right to sell you a license to Cisco's propietary software, named IOS . If you need to purchase a software license from Cisco then this may cost more than the used 2511 hardware.

Another hidden cost is the price of a maintenance contract with Cisco. This entitles you to hardware repair or replacement and software upgrades. The software upgrades include the IOS operating software and the boot ROMs. Third parties will also offer maintenance contracts on Cisco equipment, these contracts may be significantly cheaper.

The IOS software is stored in flash memory. Later versions of IOS are larger than earlier versions, so you may need a flash memory upgrade and a dynamic memory upgrade to run a later version of IOS. If you plan to upgrade the flash memory then be aware that the boot ROM needs to be aware of the flash memory's characteristics.[5] An old boot ROM may not load IOS from a newly−purchased flash memory DIMM. It is best to order a new boot ROM when upgrading the flash memory.

Purchasing flash memory and dynamic memory from Cisco may not be as economic as purchasing Cisco−approved memory from a third party supplier such as Kingston or MemoryX.

Figure D−1. Basic configuration for Cisco 2511 terminal server to Linux PC

```
interface Async1
 description To Linux PC
 ip unnumbered Loopback0
 async mode interactive
 no peer default ip address

line 1
 location To Linux PC
 session−timeout 30
 no exec
```

```
 login
 modem InOut
 terminal-type vt100
 special-character-bits 8
 transport preferred none
 transport input telnet
 telnet break-on-ip
 telnet ip-on-break
 stopbits 1
 flowcontrol hardware

line vty 0 4
 location Network
 password PASSWORD
 login local
 terminal-type vt100
 transport preferred none
 transport output telnet
```

There is a port of Linux to the Cisco 2500 series of routers. At the time of writing it did did not support the asycnhronous ports on the Cisco 2511. The attractiveness of running Linux instead of running Cisco's IOS is that Linux can support SSH. At the time of writing Cisco were yet to release SSH on the Cisco 2500 series of routers, although a unofficial beta version has been seen.

Appendix E. Gratuitous advice for developers

E.1. Advice for boot loader authors

Serial console support in a boot loader is very useful. Thank you for supporting it.

The boot loader should support the 8250A UART and its programming–compatible 82510, 16450, 16550 and 16750 descendants. The serial chip used in the IBM PC/XT, the 8250 (no A), and its 8250B descendant need not be supported. The 8250A data sheet is *82C50A CMOS Asynchronous Communications Element* and is updated by Intel's errata *82510 PC Software Compatibility*. The 16550 data sheet is *PC16550D Universal Asynchronous Receiver/Transmitter with FIFOs*.

To set the serial port and serial parameters, most Linux boot loaders use a syntax modeled upon the kernel's `console` parameter. It would be nice to retain this consistency, since the user needs to learn the kernel syntax in any case.

The default value should be `0,9600n81`. This gives the maximum interoperability with the other programs that use the serial console.

Please do not ignore the lower speeds, as remote serial console is at its most valuable when the computer is located three days walk up a mountain in the New Guinea highlands. It is difficult to get more than 75bps from HF radio under adverse sky conditions.

Be conservative in your use of the modem status lines. Even if you are ignoring incoming status (DSR, DCD) and handshaking lines (RTS) at least assert the outgoing status (DTR) and handshaking (CTS) lines. Correctly configured modems will not receive calls with DTR low, and dropping DTR will cause the modem to hang up.

Consider that the BIOS may have already initialised the UART and provide a configuration option to allow the boot loader to be informed of that. When the boot loader initialises the UART, DTR will fall and the line will hang up. In some scenarios each hang up requires the satelite circuit to be re–booked before another call can be placed.

Cater for line noise. Imagine the boot loader starting and then being sent nonsensical characters every few seconds. Although this is certainly wrong, a fault in a modem is difficult to remotely diagnose and correct if the machine is left stranded at the boot loader prompt. A solution is to boot the default image upon the expiry of a timer; the boot occurring even if the user (or line noise) has started to type. For example the boot loader configuration could say:

```
# Start the machine regardless after 30 minutes
#   30 * 60 seconds per minute * units of tenths of seconds
lifetime 18000
```

The default should be no life timer. The timer is also useful in high availability applications: when a machine is used in environments with an planned availability of 99.999% the lifetime value should be configured to three minutes or less.

E.2. Advice for BIOS authors

Thank you for adding support for remote operations to your BIOS. A few points will maximize the benefits of that support, most of them are listed in Section E.1.

- Keep the user interface simple. There is no need for fancy cursor–addressed terminal support. Fancy features simply limit the number of client terminal emulators that can be used. A surprising number of these have very buggy DEC VT100 implementations.

  In addition to supporting lower speeds, also test your user interface at low data rates.
- Don't do too much. In Linux the boot loader and operating system both have explicit support for a serial console. So all the BIOS need do is to support the a serial interface for itself. Linux has no need for a generic serial redirection facility. If you do provide such a facility for other operating systems, please allow it to be disabled after system boot.
- Don't allow line noise to prevent the computer from booting. Don't require just one key to enter the BIOS configuration, make your users and your marketing people happy by using a phrase like `dell`, `hp` or `ibm`. Copy the `lifetime` idea from Section E.1.
- Present a consistent prompt. Imagine a user with a supercomputer array of five hundred PCs. You want to change a BIOS parameter. Make it easy for Expect to set those parameters.
- Make sure the Linux utilities work. Check that the Linux `nvram` device driver returns the full contents of CMOS. This makes it simple to set the same CMOS settings on a large number of machines. The commands in Figure E–2 and Figure E–3 should work to copy the BIOS settings from one machine to another, where the make, model and BIOS versions of the machines are the same.

  Figure E–1. Configuring /dev/nvram to access the CMOS configuration

  ```
  bash# /dev/MAKEDEV nvram
  bash# vi /etc/modules.conf
  ```

```
alias char-major-10-144 nvram
```

```
bash# depmod -a
```

Figure E−2. Getting the CMOS configuration

```
bash# cat /dev/nvram > /etc/nvram.bin
```

Figure E−3. Setting the CMOS configuration

```
bash# cat /etc/nvram.bin > /dev/nvram
```

Have a flash BIOS upgrade program that works from Linux. Make the source code to this available. Or publish the specifications so that one can be written.

- Be clear about what you are providing. Some BIOSs with a serial redirection feature don't allow the BIOS to be redirected to a plain text terminal, but instead use a proprietary protocol. This isn't of much use to Linux serial console users.

Colophon

Written in DocBook 4.1 SGML. XEmacs and the PSGML package were used to create the SGML source file. The HTML and PDF output was generated using Jade Wrapper and the Norman Walsh DSSSL stylesheets, which were tweaked by the Linux Documentation Project.

Jade Wrapper is a front end to OpenJade. OpenJade in turn uses TeX for its page layout.

Notes

[1]
: The Linux 2.4 kernel also supports the output of console messages to Centronics or *IEEE 1284−2000* parallel printer interfaces.

[2]
: A *bit−time* is the time taken to transmit one bit. The distinction between *bit−times* of signal and *bits* of data is apparent when you consider that 1.5 bit–times of signal is possible but that 1.5 bits of data is impossible.

[3]
: This is not as inefficient as it may appear. The last 5% of a disk formatted with a general purpose filesystem always performs poorly and is best left empty.

[4]
: But don't submit your proposed password to a search engine! Sending passwords in plain text across the Internet isn't good, nor the possibility of having them appear in the logs of a search engine.

[5]
: This is a fault of the design of flash memory. It identifies itself with a model designator rather than with the timings required to read and write the memory. So to load software from flash memory the boot ROM must have a table of flash memory models and timings.