

QNX4.0

Administrator's Guide



QNX® 4.0 Operating System

Administrator's Guide

Quantum Software Systems Ltd.

175 Terrence Matthews Crescent
Kanata, Ontario K2M 1W8
Canada
(613) 591-0931

© Quantum Software Systems Ltd. 1991
All rights reserved.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise without the prior written permission of Quantum Software Systems Ltd.

Although every precaution has been taken in the preparation of this book, we assume no responsibility for any errors or omissions, nor do we assume liability for damages resulting from the use of the information contained in this book.

This edition: April, 1991

QNX is a registered trademark of Quantum Software Systems Ltd.
UNIX is a registered trademark of AT&T

UNIX Software Acknowledgments and Copyrights

The following utilities have been placed in the public domain by an individual or organization. We have enhanced these utilities and hope you will find them helpful. The full source code for these utilities along with additional notes and credits are available from our on-line update and conferencing system QUICS:

cpio, **pax**, and **tar**—developed by Mark H. Colburn and sponsored by The USENIX Association.

freeze and **melt**—Leonid A. Broukhis

Cover art by Robert Tinney.

Printed in Canada.

Acknowledgments

Several members of Quantum Software Systems Ltd. have contributed to the design and implementation of QNX 4.0. This product embodies more than a decade of research, development, and field-tested experience. Veteran QNX users will recognize the performance advantages and comfortable feel of their familiar OS, while new users can expect a “standard” operating system with a difference. We are proud to see the QNX operating system enter the standards arena without sacrificing the fundamental qualities of its original vision.

The following people deserve a special word of thanks and acknowledgment for their substantial contribution to QNX 4.0:

Software Development:

Luc N. Bazinet
Gordon Bell
Andrew C. Boyd
Brian Campbell
Grant D. Courville
Dan Dodge
Andrew Edgar
William A. Flowers
Dan Frank
Jeffrey G. George
Dan Hildebrand
Eric Johnson
Debbie Kennedy
Terry Laughlin
Steve McPolin
Rob Oakley
Martin Walter

Documentation:

Kevin Didsbury
Paul N. Leroux
Paul McManus
John M. Ostrander



About this Guide

What will I find in this guide?

The QNX operating system *Administrator's Guide* accompanies the QNX operating system (release 4.0) and is intended to show system administrators how to install and configure a QNX system.

The *Administrator's Guide* contains two main parts:

- **Installation & Configuration**—a detailed look at installing, configuring, and maintaining your system
- **Managers, Drivers, & Utilities**—a “man page” for each, arranged alphabetically



Throughout this guide, we assume that as the system administrator, you have *superuser* privileges, which you acquire by logging in using the **root** login account.

Conventions for technical terms

Throughout this manual, we use certain typographical conventions to distinguish technical terms. In general, the conventions we use conform to those found in IEEE POSIX publications. The following table summarizes our conventions.

Reference	Example
command names	ls
data types	<i>long</i>
environment variables	PATH
error numbers	[EINTR]
filenames	/dev/tty
functions	<i>system()</i>
function arguments	<i>arg1</i>
global externals	<i>errno</i>
headers	<sys/stat.h>
options	-c
option arguments	-w <i>width</i>
operands	<i>file_name</i>
parameters	<directory pathname>
symbolic constants	{_POSIX_VDISABLE}

Contents

Part I Installation & Configuration

Chapter 1 Installation 3

- Installing QNX on a hard disk 4
- Booting QNX over a network 5
- The system initialization file 8
 - Initial boot 8
 - Base-level services 10
 - Optional services 12
- Setting your time zone 13
 - Establishing the time zone 13
 - Getting the date and time from the realtime clock 14
- Licensing 15
 - Operating system licensing 15
 - Application licensing 16
 - Expanding your license 17

Chapter 2 Setting up User Accounts 19

- Security 20
 - Access control utilities 20
 - User and group IDs 21
 - The password database 23
 - Default password files 25
- Accounting file 25

Chapter 3	Connecting Terminals, Printers, & Modems	29
	Starting device drivers	30
	Parallel devices	30
	Single parallel port	30
	Multiple parallel ports	30
	Output buffers	31
	Serial devices	31
	Hardware adapters	31
	The RS-232 serial protocol	33
	Configuring serial ports	35
	Connecting serial devices	36
	Configuring terminals for users	39
	Troubleshooting serial device problems	42
Chapter 4	Print Spooling	45
	Sharing devices in a network	46
	Print spooling	46
	Submitting print jobs	47
	Querying print jobs	47
	Cancelling print jobs	47
	Controlling the print queues	48
	Configuring the spooler	48
	Spool configuration file	50
Chapter 5	Setting up a Second Filesystem	57
	Introduction	58
	Partitioning the pathname space	58
	A hard disk and a floppy	58
	Two hard disks (same node)	59
	Two or more QNX partitions	59
	Local and remote hard disks	59
	Setting up a DOS filesystem	61
	Invocation modes	61
	Starting Dosfsys	62

Dosfsys name adoption	62
DOS devices	62
DOS version support	63
DOS partition types	64
DOS text files	64
DOS binary files	64
QNX to DOS character and name mapping	65
DOS volume labels	65
DOS/QNX permission mapping	65
File ownership	66
Error codes returned by Dosfsys	66

Chapter 6 **Making Backups** **69**

Introduction	70
When to back up	70
Backup formats	70
Archive backups	71
Filesystem backups	72
Backup media	72
Floppy	72
Tape	73
Removable disk	74
Fixed disk	74
Compression	74
Archive examples	75
Compressed floppy archive	75
UNIX-compatible floppy archive	75
Tape archive	75
Cartridge/optical	76

Chapter 7 **Building a Custom Operating System Image** **77**

Introduction	78
Constructing a build file	78
Setting the heap size	79

Building a Custom Operating System Image (cont'd)

- Selecting processes for an image 79
- Disk images 80
- The Makefile 80
- Copying an image to `./boot` 81
- Network images 82
- Embedded images 83

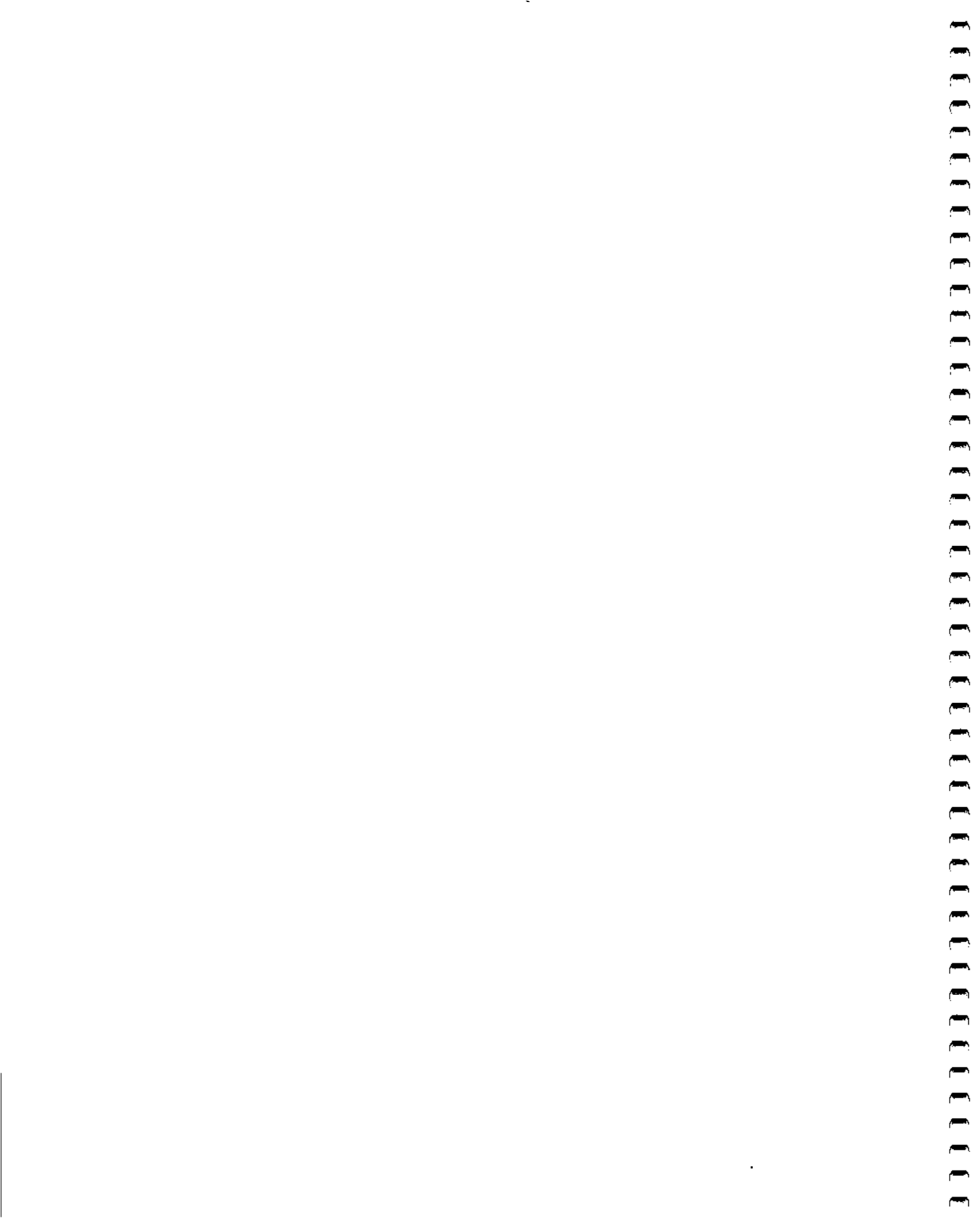
Chapter 8 Disk & File Recovery 85

- Introduction 86
- Overview of QNX 4 disk structure 86
 - Partition components 87
 - Directories 89
 - Links 90
 - Extent blocks 91
 - Files 92
- File maintenance utilities 93
 - fdisk** 93
 - dinit** 93
 - chkfsys** 93
 - zap** 94
 - spatch** 94
- Disk recovery procedures 94
 - Using **chkfsys** 94
 - Recovering deleted files 96
 - Recovering from a bad block in the middle of a file 97
- What to do if your system will no longer boot 97
 - If the disk is unrecoverable 101
 - If the filesystem is intact 101
- Recovering lost files and directories 101
 - The **lost+found** directory 102
 - Using **spatch** 102

Chapter 9 The Quantum Update System 105

Introduction 106
Reporting bugs 106
Free software 106
Electronic mail 106
Conferencing system 107
About our hardware 107
How to phone us 108
Logging in 109
Using QUICS 110

Part II Managers, Drivers, & Utilities**Appendix A Time Zone Rules 201****Index 207**



Part I

Installation & Configuration

In this Part...

Installation

Setting up User Accounts

Connecting Terminals, Printers, & Modems

Print Spooling

Setting up a Second Filesystem

Making Backups

Building a Custom Operating System Image

Disk & File Recovery

The Quantum Update System



Chapter 1

Installation

In this Chapter...

Installing QNX on a hard disk

Booting QNX over a network

The system initialization file

Setting your time zone

Licensing

Installing QNX on a hard disk

You use an automated **install** program to install QNX 4.0 onto a hard disk.

To install QNX 4.0, follow these steps:

Step 1 **Boot from floppy**

Place the QNX 4.0 *boot diskette* into the floppy drive and reboot the computer. You should see a spinning arrow in the top left corner of your screen, followed by the QNX logo, a welcome message, and a shell prompt.

Step 2 **Run the `install` program**

At the shell prompt (**#**), type:

```
install
```

Simply follow the instructions on your screen to set up your hard disk so it will boot QNX 4.0.

Step 3 **Reboot from hard disk**

Once installation is complete, you should remove any floppy diskettes and reboot your computer from hard disk. QNX 4.0 should now be up and running.

Step 4 **Read the *User's Guide***

At this point, it's a good idea to familiarize yourself with the basic philosophy and operation of QNX as described in the *User's Guide*. For the remainder of this *Administrator's Guide*, we'll assume that you have a basic level of familiarity with QNX.

Booting QNX over a network

Once you've installed QNX onto a hard disk that will be used as a network file server, follow these steps to boot a computer over the network. Note that the booting computer may be a diskless workstation or a file server.



Before you boot over the network, you must first install QNX on at least one machine that boots from hard disk (as described in the previous section). If you haven't done this, please do so now, then return to this section.

Step 1 Check your network license

QNX must be licensed to run on each computer in a network. Before you proceed, make sure you are licensed to run the machine you wish to boot over the network. This is described in the section "Licensing" in this chapter.

Step 2 Select a boot server

You must select which node in the network you wish to boot from.

For small networks we recommend you pick a single node, typically node 1. This node should have a local QNX filesystem.

For larger networks, you may want to select more than one boot server node. For example, let's assume you have three departments—R&D, Marketing, and Operations—each with its own machine that acts as a major file server. You may elect to have each department boot from its own major server. This provides fault tolerance, since a failure in one department won't prevent another department from booting. For large numbers of nodes, this practice should also reduce the potential for a boot bottleneck when people in the office start work first thing in the morning.

Step 3 Start the **netboot** utility

When a node boots, it sends a boot request to its boot server. When the network manager receives a boot request message, it forwards this request to the **netboot** utility, which must be running on the boot server if the request is to be honored.

When you boot over the network, the operating system image loaded will inherit the following from the boot server:

- the time
- its list of name locators (see **nameloc** utility in *Utilities Reference*)
- its licensing capabilities

The **netboot** utility is typically started in the system initialization file of the boot server as follows:

```
netboot &
```

Step 4 Edit the **netboot** configuration file

When a boot request is received, the **netboot** utility opens the **/etc/config/netboot** file on the boot server. This file contains entries that describe which operating system image to boot for each node in the network.

Each line in the **/etc/config/netboot** file starts with a logical node number followed by a file specification identifying the operating system image to boot. For example, the following file might be used to boot nodes 2 through 7 (node 1 is the boot server, which boots from disk, so it doesn't need to be in the file):

```
2 f=build/ws
3 f=build/ws
4 f=build/ws2
5 f=build/ws
6 f=build/ws
7 F=images/special
```

The first line, **f=build/ws**, indicates that the **buildqnx** utility should be invoked to construct a workstation image (**ws**) on the fly. On the standard distribution, we provide a file by this name, which is suitable for booting a machine as a network workstation.

The **F=images/special** line will ship the exact image in **/boot/images/special**. For more information, see Chapter 7, Building an Operating System Image.

The above file will allow *only* nodes 2 through 7 to boot. You could simplify the file as follows:

```
4  f=build/ws2
7  F=images/special
*  f=build/ws
```

where ***** matches any node. Note that the ***** must be the last entry in the file.

In this case, all nodes except 4 and 7 will use the **ws** build file to construct a boot image.

Step 5 Configure the network card

See the information given in the documentation that was shipped with your network card or network ROM from Quantum.

Step 6 Connect the card to the network

You should now physically connect the new machine into the network.

Step 7 Create a system initialization file

Now create a custom system initialization file. Creating this file is described in the section “The system initialization file,” below.

Note that you can still boot without creating a custom system initialization file. If you boot without creating your own file, the default file **/etc/config/sysinit** will be used. That file should contain generic commands suitable to start any machine.

The system initialization file

Initial boot

When you boot QNX, you are booting an image composed of several processes. The first process in the image is always the Process Manager (which contains the Microkernel). The Process Manager performs processor initialization, and then schedules each process included within the image for execution.

The last process is the **sinit** utility, which starts the second phase of your system initialization. This is done by starting a shell that executes commands from a system initialization file. This is a standard shell file that runs in exactly the same manner as any other shell file except that breaks are disabled.

Starting system services after boot is one of the benefits of QNX's modular architecture. The image that is booted typically contains only those few essential services needed to start all other desired services.

In order to run, **sinit** needs access to a filesystem that contains a shell that can process the system initialization file. This filesystem can be local (a hard disk on that machine) or remote (a hard disk on another machine). If the filesystem is local, the image must contain the Filesystem Manager and a disk driver to access the local disk. If the filesystem is remote, the image doesn't need the Filesystem Manager, but will require the Network Manager to access the remote filesystem over the network.

When **sinit** runs, it firsts determines if the image it's part of was booted from disk or over the network. If the image was booted from disk, **sinit** checks to see if a normal boot or an alternate boot occurred. Based upon this information, it attempts to open one of two files (see below). If the open succeeds, then **sinit** will *exec* into the Shell (**/bin/sh**) with the file it opened as an operand. The Shell replaces **sinit**, whose job is now done.

If (and only if) **sinit** can't open this first file, it will attempt to open the **/etc/config/sysinit** file. If it opens this file, it will *exec* into the Shell with this file as an operand. If it can open neither file, then **sinit** will terminate and no system initialization file will be processed. This is summarized in the following table:

If you choose:	sinit tries to find and open:
a normal boot	<code>/etc/config/sysinit.node</code> If this fails, it tries: <code>/etc/config/sysinit</code>
an alternate boot	<code>/etc/config/altsysinit</code> If this fails, it tries: <code>/etc/config/sysinit</code>

For any given node in the network, a total of three potential files may be executed by **sinit**:

- **sysinit.node**
- **altsysinit**
- **sysinit**

Each of these files serves a particular purpose.

sysinit.node

This file contains the custom commands needed to set up the environment and services for your machine, allowing every node in the network to have its own custom setup. The *node* after the dot (.) is replaced at runtime by **sinit** with the logical node number of your machine. This is a value between 1 and the number of nodes in your network. If your machine doesn't contain a network card, *node* will be 1.

altsysinit

This file is executed only if you boot from the local disk and specify an alternate boot by pressing <Esc> when you are prompted to do so just before the QNX loader loads the operating system image.

The **altsysinit** file should contain a backup of the last working copy of the **sysinit.node** file for this machine. It's there as a safety net in case you make a modification to your **sysinit.node** file that leaves the system in a state where you can't log in.

Since many services are started in the system initialization file, we recommend that you update this file after all successful changes to your *node*-specific file.

Note that the **altsysinit** file is never used by a node that boots over the network. In this case, you can correct any problems in a modified **sysinit.node** on the up-and-running boot server node.

sysinit

This file is executed if a custom node-specific system initialization file isn't present. It should contain services that will work on any machine. Quantum ships such a generic file; we recommend that you make few or no modifications to this file.

For testing purposes, you can boot a new machine over the network without creating a custom **sysinit.node** file. When the node boots, it will use the default **sysinit** file. Once you're satisfied that the machine is functional, you can then create a node-specific initialization file.



If you remove a network card, or change the logical node number of a machine, the machine won't find its old system initialization file, which is keyed off its old node number. In this case, the machine will look for a new node-specific file (node 1 if the card was removed or if a new node number is to be used). If the file is missing, the machine will use the default **sysinit** file.

Base-level services

The contents of each machine's system initialization file reflects the hardware on that machine and the services you wish to provide. The following describes a base set of services that will be in most initialization files.

The following lines define the time zone (EST in this case) and get the time from the realtime clock. These two lines should be the first in the file for machines that boot from hard disk. They are optional in a machine that boots over the network, since the machine inherits this information from the node it boots from.

```
export TZ=EST5DST4,M4.1.0/3,M10.5.0/3
rtc hw
```

The following lines start the Device Manager and the console driver with eight virtual consoles, then instruct the Shell to reopen its standard I/O through the new console device.

```
Dev &  
Dev.con -n 8 &  
reopen /dev/con1
```

The following lines start up any other device drivers you may need. They start both a serial driver, which will look for COM1 and COM2, and a parallel driver. These drivers will terminate if they can't find the needed hardware.

```
Dev.ser &  
Dev.par &
```

If you started **Dev.ser**, you might need to use the **stty** utility to change the default line configuration. For example, the following lines change the baud rate to 19200:

```
stty baud=19200 </dev/ser1  
stty baud=19200 </dev/ser2  
etc...
```

If your programs use floating point and you don't have an 80x87 chip, you need to start the floating point emulator:

```
emu87 &
```

The following line starts a login on the first console and arms all other consoles. This should be the *last* command executed in the file. This particular example shows a reasonable default suitable for the console:

```
tinit -T /dev/con* -t /dev/con1 &
```



You may wish to place common sets of commands in a separate Shell file that you execute via the **.** (dot) Shell built-in. This allows you to edit a single file to affect a group of machines. For example, the following system initialization file could be used for all machines in the Technical Department:

```
./etc/config/techies
```

Note that node-specific commands would be added after this “dot” line, followed by the **tinit** utility.

Optional services

You can add a host of other services to your system initialization file. You should add these services just before the line containing the **tinit** command. The following examples show commonly used optional services. Note that these utilities typically support command-line options to modify their behavior—these options are explained in the documentation for each utility.

Define any environment variables you wish inherited by all processes started in the **sysinit** file:

```
export var=value
```

Start a local floppy driver (if the QNX filesystem is running locally, *don't* include the first line):

```
Fsys &  
Fsys.floppy
```

If a floppy driver is running, you can use the **vol** and **pax** utilities to access the driver's block special file. If you want to access it as a QNX filesystem, you must mount it as such (we recommend that you transport files using **pax** rather than as files in a QNX filesystem):

```
mount /dev/fd0 /fd0
```

Start a DOS filesystem to access DOS floppies and partitions. This requires that the QNX filesystem be running with a driver for the floppy and/or hard disk:

```
Dosfsys &
```

If this is a boot file server, start the **netboot** utility:

```
netboot &
```

If this is a machine that you hope will be available at all times, you may select it as a global name server:

```
nameloc &
```

If this is a machine that you hope will be available at all times, you may also select it as a **cron** server:

```
cron &
```

Start a clock that will continuously display the time in the upper right corner of the screen:

```
clock -f green "+%X" &
```

Setting your time zone

It's important that the correct date, time, and time zone information be established early during initialization. This should be the first step in your system initialization file.

Internally, QNX uses Coordinated Universal Time (UTC), which is often referred to as Greenwich Mean Time. Applications and utilities convert to local time by using information about your time zone.



If you don't set your time zone, QNX will assume that local time is the same as UTC time. This would present a problem only if you wish to transfer files to another system in a different time zone. In this case, the dates on the file will appear to be shifted by the difference between the two time zones. If you plan to use Quantum's update system to download new software, you'll want to set your time zone correctly.

Establishing the time zone

You should establish time zone information before setting the current date and time. If the realtime clock in your computer has been set to local time, QNX needs the time zone information in order to establish UTC.

In the following example, the time zone as well as the time change rules are set for Eastern Standard Time in North America:

```
export TZ=EST5DST4,M4.1.0/3,M10.5.0/3
```

where:

export	Shell command to set an environment variable
TZ	Name of variable
EST	Eastern Standard Time
5	5 hours before Greenwich Mean Time
DST	Daylight savings time
4	4 hours before Greenwich Mean Time
M4.1.0/3	First Sunday of April at 3am
M10.5.0/3	Last Sunday of October at 3am



For more information on the time zone rules, see Appendix A.

Getting the date and time from the realtime clock

If you're booting from disk, you should follow the first line in your system initialization file with the **rtc** utility to establish the current date and time from the realtime clock. The following two lines would accomplish this:

```
export TZ=EST5DST4,M4.1.0/3,M10.5.0/3  
rtc hw
```

This will set the date and time on an IBM AT-compatible computer.

Note that there are two possible approaches to take when setting the realtime clock in your machine:

- setting the realtime clock to *universal* time
- setting the realtime clock to *local* time

We recommend that you set the time in the realtime clock to UTC. But if you're also running operating systems that assume the realtime clock is set to local time (e.g. DOS), you'll want to use the **rtc** utility with the **-l** option:

```
rtc -l hw
```

This invocation of **rtc** assumes that the realtime clock was set to local time. Note that when you use local time in the realtime clock, you'll have to manually change the value in the realtime clock when you switch to and from daylight savings time in locales where it's used.



If the time in your hardware clock is incorrect (perhaps the battery has been replaced), you should set the system time using the **date** utility, then set the realtime clock using the **rtc** utility with the **-s** option. For details, see the documentation for these utilities in the *Utilities Reference*.

If you're booting over the network...

If you boot over the network, the booting machine will inherit the UTC time and time zone environment variable from the node it booted from. Therefore, you don't need to put this information in your system initialization file.

Licensing

Operating system licensing

QNX must be licensed for each computer it runs on. This is true whether you're using 10 standalone machines or 10 networked machines.

In a network, each machine is referred to as a "node" and each node is assigned a "logical" node number. Logical node numbers range from 1 to the number of nodes in the network. For example, a network licensed for three nodes would support logical node numbers 1, 2, and 3. You set the size of the network by running the **license** utility (see the section "Expanding your license").

You can use the **sin** utility to query the number of nodes your network has been licensed for.

Typing:

```
sin info
```

will produce output similar to the following:

```
Node   CPU      Machine Speed      Memory    Ticksize  Monitor      Flags
  2  386/0          AT   563    840k/ 2048k 10ms    VGA Color  -----P

Heap  Handlers Names Sessions Procs  Timers Nodes Boot
13668      32    30         32   150     40    10 Hard

Locators: 1 4
```

The number under the **Nodes** heading (in this case 10) indicates the number of nodes licensed within this network.

Application licensing

Per node licensing

Applications are usually licensed per node *or* per concurrent use.

Some applications are licensed per node in a QNX network in the same way as the operating system. If you have an application licensed for, say, three nodes, then you would be restricted to running it only on three nodes of your network. Note that your network license size may exceed the license size for an application.

You may run multiple invocations of the application on each licensed node. When you run an application licensed per node, the invoking node will consume one of the licenses. The license will remain with that node until it is rebooted.

By convention, serial numbers that end in “**nddd**” are licensed per node.

Concurrent use licensing

Applications may also be licensed for the number of copies in concurrent use. For example, if an application is licensed for three concurrent copies, then up to three copies may be run anywhere in the network at one time. Note that on a non-networked single node machine, you may still run only three concurrent invocations.

By convention, serial numbers that end in “**uddd**” are licensed per use.

Expanding your license

You expand your license size for the operating system or an application using the **license** utility. This utility installs license serial numbers from floppies to your hard disk. Each QNX boot diskette and application diskette contains a serial number license for one node (or for one use).

The following example transfers the license from floppy drive 0 ("drive A:" in DOS terminology) to your hard disk. Note that the floppy driver (**Fsys.floppy**) must be running.

```
license /dev/fd0
```

You can list the serial numbers you own by doing an **ls** of the **/etc/licenses** directory. After adding new serial numbers to your set of licenses, you must reboot before they will take effect.

Any machines that boot over the network will inherit the serial numbers from the machine they boot from. If you wish to boot networked machines from their local hard disk, you must use the **license** utility to add serial numbers to *each* hard disk you wish to boot from. Once you have done this on a single hard disk, you can use **license** to install the serial numbers across the network from one hard disk to another and thus avoid having to re-insert floppies.

For more information on the **license** utility, see Part II, *Managers, Drivers, & Utilities*.



Chapter 2

Setting up User Accounts

In this Chapter...

Security

Accounting file

Security

QNX provides mechanisms to control access to resources and critical system functions. These mechanisms are based on the ability of the system to identify a particular user.

Access control utilities

The QNX access control utilities are:

login	sign on to the system
su	temporarily become another user
passwd	maintain the user accounts/change password
newgrp	switch to a different group

The **login** utility The **login** utility should be started by **tinit** on all tty devices. The **login** utility demands a username and password which it verifies in the password database. If the user fails to enter a correct username/password combination within three attempts, the utility will exit, updating appropriate accounting information. If the user enters a correct combination, **login** will start a login shell and will load the user's environment.

The **su** utility The **su** utility lets a user temporarily have the privileges of another user. If the user is not the superuser (**root**), **su** will demand the appropriate password. If a valid combination is entered, **su** will do an *exec()* to create a shell for the user with the effective group ID and user ID of the desired user.

Exiting from the shell will return you to where you were before running **su**.

The **passwd** utility The **passwd** utility may be used to change a password or to add a new user account to the system. Only the superuser (**root**) may add a new account. When changing or setting a password, the **passwd** utility will insist that the password consist of at least six characters from at least two different character classes. Examples of character classes are alphabetic, numeric, and punctuation. This is to lessen the chance of accidental discovery of another user's password and to discourage unsafe practices.

The **passwd** utility asks the user for the old password, the new password, then the new password again (to verify it was entered correctly). If the user is the superuser, the old password isn't requested.

When the superuser adds a new account, **passwd** prompts for the following information:

- user name
- user ID
- initial password
- home directory
- shell
- group name

If the initial password isn't entered, the user account will be created without a password. If the home directory doesn't exist, it will be created under **/usr** with the permission mask "**rwxr-xr-x**." If no shell is given, the system default shell (**/bin/sh**) will be used. If the group doesn't exist, a new group will be created with one member, namely this user. If the group *does* exist, this user will be added to it.

The **passwd** utility doesn't provide any mechanism for deleting accounts. This function may be performed using a text editor on the **/etc/passwd**, **/etc/group**, and **/etc/shadow** files.

The **newgrp** utility

The **newgrp** utility changes the current process's real and effective group ID. If invoked without arguments, **newgrp** changes the group identification to the group identified in the password database for the current user. The groups that a particular user may change to are controlled by the password database.

User and group IDs

QNX uses two pieces of information to control access to resources:

- user ID
- group ID

The user ID should be unique—that is, no two users should share the same user ID. This rule is enforced by the **passwd** utility, but the superuser can override this rule by editing the password file directly.

The group ID refers to a database that allows several users to be associated to a group. This group mechanism allows a team of users to share resources without making them available to the rest of the world.

File permissions

All files and directories in the QNX filesystem have three sets of access fields corresponding to:

- owner
- members of the owner's group
- everyone else

All files and directories also have fields that identify the *owner* and *group* of the file.

setuid and setgid

Executable files have further permissions that cause them to execute with the privileges of the file's owner or group, regardless of who executes them. Colloquially, this mechanism is referred to as "setuid" and "setgid," respectively.

This simple mechanism allows non-privileged users to perform certain privileged functions. For example, **passwd** allows anyone to modify their own entry in the password database, even though they would normally be denied access to the file.

The **passwd**, **login**, **su**, and **newgrp** utilities are all "setuid" to **root**; these programs therefore run with the permissions of the superuser.



By convention, **root** is the only user with user ID zero, which yields superuser status. With respect to access control, you must ensure that only programs that can be trusted—and absolutely need to be trusted—are “setuid” or “setgid” to **root**.

Since these “setuid” and “setgid” programs will inherit superuser capabilities, you should make sure they do *not* have general write permissions so that non-superusers won’t be able to modify the programs.

The password database

Three files collectively form the *password database*:

- **/etc/passwd**
- **/etc/shadow**
- **/etc/group**

The access to these files should be as follows:

File	Owner	Group	Permissions
/etc/passwd	root	root	rw- r-- r--
/etc/group	root	root	rw- r-- r--
/etc/shadow	root	root	rw- --- ---

/etc/passwd

The **/etc/passwd** file contains a set of lines in the following format:

```
username:haspw:userid:group:comment:homedir:shell
```

where:

<i>username</i>	Login name of user
<i>haspw</i>	If empty, user has no password, otherwise a password exists in the /etc/shadow file
<i>userid</i>	Numeric user ID
<i>group</i>	Numeric group ID
<i>comment</i>	Free-form comment field, must not contain “:”
<i>homedir</i>	Home directory of this user (default is /)
<i>shell</i>	Initial shell (and arguments) to start from login (default is /bin/sh)

/etc/group The **/etc/group** file contains lines in the following format:

*groupname:reserved:group:member[,member]**

where:

<i>groupname</i>	The name of the group
<i>reserved</i>	Reserved for later use
<i>group</i>	Numeric ID of the group
<i>member</i>	A list of usernames that belong to this group

/etc/shadow The **/etc/shadow** file contains lines in the following format:

userid:password:reserved:reserved:reserved

where:

<i>userid</i>	The name of this user
<i>password</i>	The encrypted password of this user
<i>reserved</i>	The first three numeric fields are reserved for “password aging”; for now, they are always 0.

You may notice from the above permission list that **/etc/passwd** is readable by anyone. This is to provide standard utilities with a simple mechanism to find information about users. Since this file is readable, the encrypted password isn’t stored within it. The encrypted password is stored in the **/etc/shadow** file, which is readable only by the superuser. This is to inhibit unauthorized attempts to *decrypt* the passwords by malevolent users. As a system administrator, you should ensure that these permissions are maintained.

QNX is shipped with a default password database that includes **/etc/passwd** and **/etc/group**. The **/etc/shadow** file isn’t shipped, because the accounts initially don’t have passwords associated with them.

The first job, after the system is installed, is to set up passwords for all the *system accounts* and to define any *user accounts*. You perform both of these functions with the **passwd** utility. By convention, user accounts have a user ID greater than or equal to 100.

Default password files

The default `/etc/passwd` file that was shipped with your QNX system contains the following:

```
root::0:0:::/bin/sh
```

The default `/etc/group` file contains the following:

```
root::0:root
```

Accounting file

Accounting information is logged to the `/etc/acclog` file. If this file doesn't exist, all accounting information will be discarded. This is the normal mode of operation after QNX has been installed.

For most realtime systems, this default of discarding accounting information is recommended. If you have a dial-up line to a computer or if you run QNX in a network of many users, you may wish to change this default by creating an empty `/etc/acclog` file.

Enabling accounting

To enable accounting, you create an empty `/etc/acclog` file. You can do this using the `touch` utility:

```
touch /etc/acclog
chmod g,o= /etc/acclog
```

Once this file is created, accounting information will be logged here.

Note that only the superuser (user ID `root`) should create and modify this file.

Record format

Each record in this text file is of the form:

```
tttttttt cc data...
```

where `tttttttt` is the time in seconds (in decimal). This is always followed by a single space. The time is followed by a two-character code `cc`. This code is then followed by a space and `data` specific to each code and terminated by a `<newline>`. The following utilities write these accounting records:

Utility	Purpose	Record
login	user logged in	<i>TTTTTTTT LO device uid gid uname</i>
login	login failed	<i>TTTTTTTT LF device uname</i>
modem	modem connect	<i>TTTTTTTT MO device baud</i>
su	switch user	<i>TTTTTTTT SU device uid gid uname</i>
tinit	start a command	<i>TTTTTTTT TS device command</i>
tinit	arm a device	<i>TTTTTTTT TA device</i>

A typical logfile might look like this:

```
670464500 TS //1/dev/ser1 modem -b 19200 -L
670464545 MO //1/dev/ser1 2400
670464550 LO //1/dev/ser1 100 101 steve
670465824 TS //1/dev/ser1 modem -b 19200 -L
```

This record shows that **tinit** started a **modem** program to wait for calls. A call was received and answered at 2400 baud, and user ID **steve** logged in. Note that the log doesn't show a logout. The logout is inferred, because in the final entry **tinit** starts another **modem** program.

The total connect time for the user (from successful login) can be calculated thus:

$$670465824 - 670464550 = 1274 \text{ seconds}$$

On a busy system, records from many devices will be interspersed throughout the logfile. In order to match events keyed to each device, you'll find a node number associated with each device, letting you track accounting records for all devices throughout a network in a single logfile.

Here are several common event sequences:

TS → LO → TS	A login and logout on a dedicated line.
TS → MO → LO → TS	A login and logout on a dial-up line.
TA → TS → LO → TA	A login and logout on a dedicated line armed by a keystroke.
TS → TS	An unsuccessful login on a dedicated line.
TS → MO → TS	An unsuccessful login on a dial-up line.

Clearing the logfile

Once you create `/etc/acclog`, the file will start to grow as records are appended to it. If left unmanaged, this file may grow to consume considerable disk space, so you should print or archive the information in this file on a regular basis. You may even want to automate this housekeeping job using the `cron` utility (see the *Utilities Reference*).

In the following example, the logfile is moved to a file named by year and month, and a new empty log is created:

```
mv /etc/acclog /etc/acclogs/9106
touch /etc/acclog
```

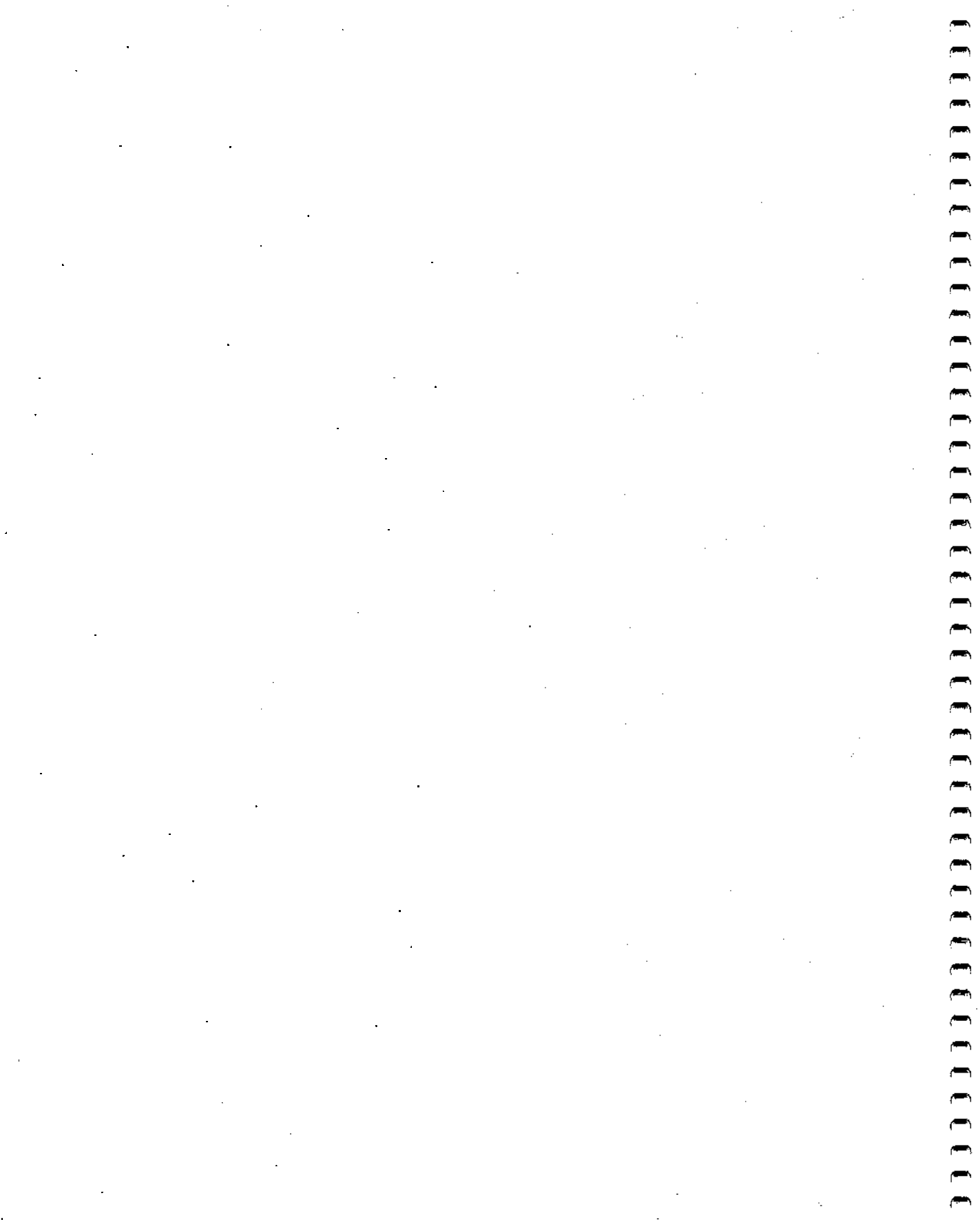


Since the data in this file is very regular, you may use the `freeze` compression utility, which will achieve very high rates of compression on the file. This can significantly reduce disk space requirements if you keep the saved logs online or save them to a floppy. Remember to move the file *before* compressing it. Never compress `/etc/acclog` directly.

Here's an example of the recommended compression procedure:

```
mv /etc/acclog /etc/logs/9106
touch /etc/acclog
freeze /etc/logs/9106
```

Note also that other utilities (possibly third-party) may add their own accounting records to the `/etc/acclog` file. Utilities may also log problems or unexpected events to the `/etc/syslog` text file.



Chapter 3

Connecting Terminals, Printers, & Modems

In this Chapter...

Starting device drivers

Parallel devices

Serial devices

Starting device drivers

A QNX system will usually contain one or more *terminal devices*. All such devices are managed by the **Dev** process. This process must be started first, before any device drivers are started.

/bin/Dev & Device Manager process

Once **Dev** has been started, one or more of the following device drivers may be started:

/bin/Dev.con & console device driver process

/bin/Dev.par & parallel printer device driver process

/bin/Dev.ser & serial device driver process

Each of these drivers is described in more detail in Part II, *Managers, Drivers, & Utilities*.

Parallel devices

Parallel ports are used to communicate with parallel printers. Apart from starting up the driver, there's little work for you to do other than connect the printer to the machine.

Single parallel port

If only one parallel port is available on a machine, then no parameters are required:

Dev.par &

Started this way, the parallel driver will create a device called **/dev/par**, which corresponds to the first parallel port found by the BIOS (LPT1).

Multiple parallel ports

If your machine has more than one parallel port, then you'll need to start an additional **Dev.par** for each extra port. You must provide a unique name for the extra devices. For example:

Dev.par &
Dev.par -b 2 -N laser &

Output buffers

These commands will create a device called **/dev/par** on LPT1, and a second device called **/dev/laser** on LPT2.

If you have the memory, you might find that specifying large output buffers significantly improves turnaround time when sending data to your printer. Here's an example of a parallel device created with a 30K output buffer:

```
Dev.par -O 30000 &
```

Serial devices

Hardware adapters

I/O addresses

The **Dev.ser** driver can support one or more serial ports. The hardware interface to the computer consists of a UART (*Universal Asynchronous Receiver/Transmitter*) for each serial port. The driver will support any or all of the 8250, 16450, or 16550 family of serial controllers.

Each UART exists in the I/O address space of the computer at eight consecutive addresses. The **Dev.ser** driver is informed of the I/O address range for each UART by command-line arguments when it is started.

Hardware interrupt

Just as important as the I/O address is the hardware interrupt generated by each UART. Most microcomputers provide several hardware interrupt signals on the bus, labelled IRQ2 through IRQ15 (except for interrupts 0, 1, 8, 9, and 13, which are used internally by the system motherboard).

These interrupt signals are active-high TTL logic signals on ISA buses, which means that you can connect only *one* adapter card to any one interrupt signal. Microchannel architectures (MCA bus) don't have this limitation.

Serial adapter cards come in various configurations. Adapter cards with only one serial port typically offer only a limited set of choices for I/O address and hardware interrupt. The following table shows

some commonly found combinations, but we recommend that you read the hardware documentation carefully to discover the choices available on a given manufacturer’s adapter card:

Name	Address	Interrupt
COM1	3F8	IRQ4
COM2	2F8	IRQ3
COM3	3E8	varies
COM4	2E8	varies

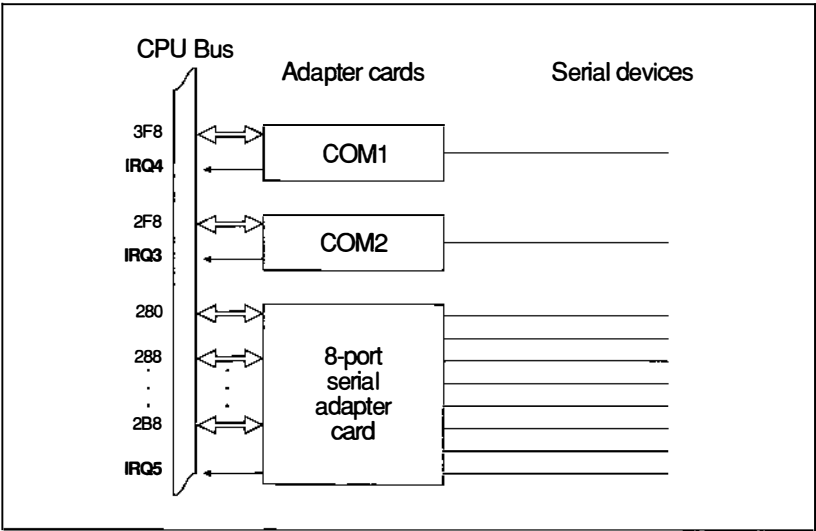
Multi-port serial adapters

You can usually configure multi-port serial adapters to respond to a wide range of I/O addresses. The adapters may also give you considerable flexibility in selecting hardware interrupts. A good choice for I/O addresses is often the 280 through 2BF range.

Because of the limited number of hardware interrupts available, these cards will often “OR” the interrupt lines from the individual UARTs into a single interrupt connected to the bus. QNX allows for many serial ports to share the same interrupt, since `Dev.ser` will check every UART that shares an interrupt.

Typical hardware installation

The following diagram shows a typical configuration of serial adapter cards in a QNX system.





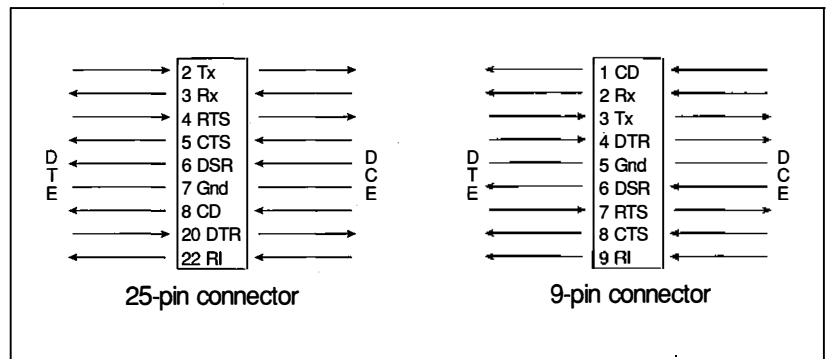
The RS-232 serial protocol

Electrical interface

For proper operation, each serial channel must have a unique I/O address, and each adapter card must use a unique hardware interrupt.

The RS-232C asynchronous communications protocol defines the electrical and physical interface between Data Terminal Equipment (DTE or *terminals*) and Data Communications Equipment (DCE or *modems*).

The following figure shows the cabling assignments of an RS-232 connection.



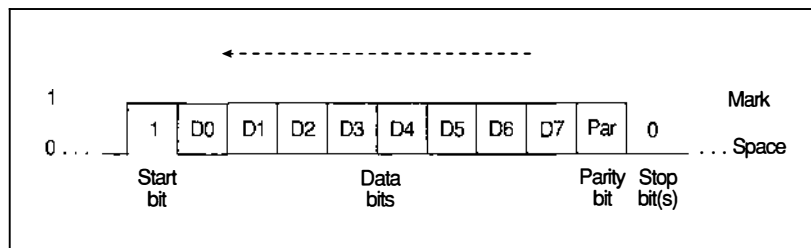
The host computer is usually configured as a DTE, acting as a terminal device. We assume that the computer will be connected to a modem device.

The RS-232 signals have the following names:

Tx	transmit data
Rx	receive data
RTS	request to send
CTS	clear to send
DSR	data set ready
DTR	data terminal ready
CD	carrier detect
RI	ring indicator

Serial protocol

Data is transmitted asynchronously using a bit protocol as shown below:



Normally, an RS-232 data line is in the SPACE (0) condition. A transmitted character consists of bits in the following order:

- 1) START bit (always 1)
- 2) 5 to 8 data bits (least-significant bit first)
- 3) parity bit (optional)
- 4) one or more STOP (0) bits.

The duration of each bit is defined by the *baud rate*, which indicates the number of bits per second that can be transmitted.

Parity is optional, but if used can be one of:

odd	sum of data bits plus parity bit is odd
even	sum of data bits plus parity bit is even
mark	always 1
space	always 0

Session control

RS-232 uses the DTR and DSR lines to control communication sessions. The terminal raises DTR when it is powered up and available. Similarly, the modem raises DSR when it is powered up and available (but not necessarily *connected* to a remote modem). No communication is expected to occur unless both DTR and DSR are raised.

A terminal indicates that it no longer wishes to communicate by dropping the DTR line, which causes most modems to hang up the telephone line, thus releasing the connection.

A modem indicates that it has established a connection with another modem by raising CD.

Some modems will also indicate that they have detected (but not yet answered) an incoming call by raising RI.

Flow control

The RTS and CTS lines control the flow of data between terminal and modem. The terminal raises RTS when it is capable of receiving data on the Rx line. Similarly, the modem raises CTS when it can accept data on the Tx line.

Configuring serial ports

You use the **stty** utility to set the four major parameters that define an RS-232 link.

Data bits

QNX supports four character sizes. You choose the size of data character with the **stty** utility:

```
stty bits = 5  
stty bits = 6  
stty bits = 7 or  
stty bits = 8 (default)
```

This parameter defines how many bits following the start bit will be used to form the least significant portion of a received character.

Stop bits

It's possible to transmit data that is followed by either one or two stop bits. Two stop bits are used only to slow down the overall transmission of data so that the remote end has a chance to keep up. Using **stty**, you specify one of these:

```
stty stopb=1 (default)  
  
stty stopb=2
```

Parity

To disable the transmission of parity bits, and suppress the checking (in hardware) of received parity bits, you specify:

```
stty par=none (default)
```

If parity is used, you specify one of the following values:

```
stty par=odd
stty par=even
stty par=mark
stty par=space
```

Baud rate You can specify the baud rate with the **baud=number** option of the **stty** utility:

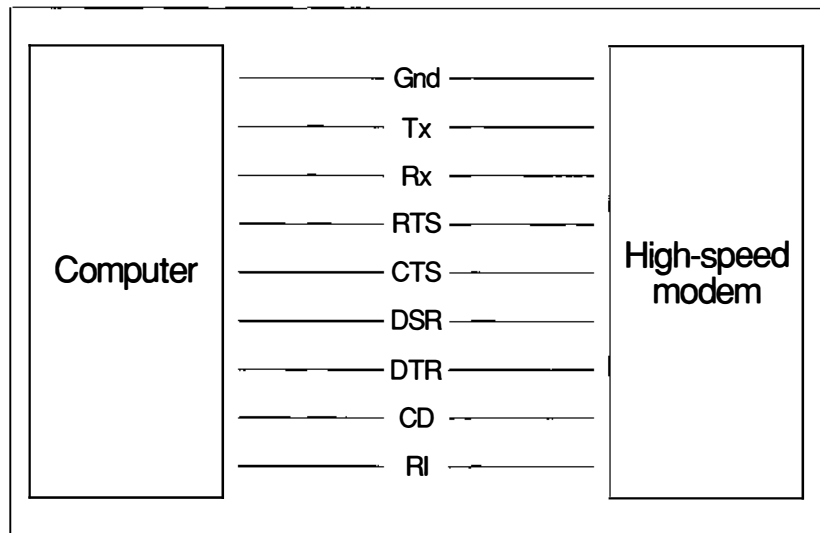
```
stty baud=number
```

QNX defaults to 9600 baud.

Connecting serial devices

High-speed ECC
modems

High-speed, error-correcting modems are becoming very sophisticated—they work best when all hardware handshaking signals are respected. These modems often communicate with the host computer at a fixed high-speed baud rate (e.g. 19200 baud) and use the RTS/CTS handshaking lines to regulate the actual flow of data over the communications link. QNX is ideally suited for communicating with such modems. A nine-pin, straight-through cable is used to connect the modem to the computer.



Cabling assignments for linking a computer to a high-speed modem.

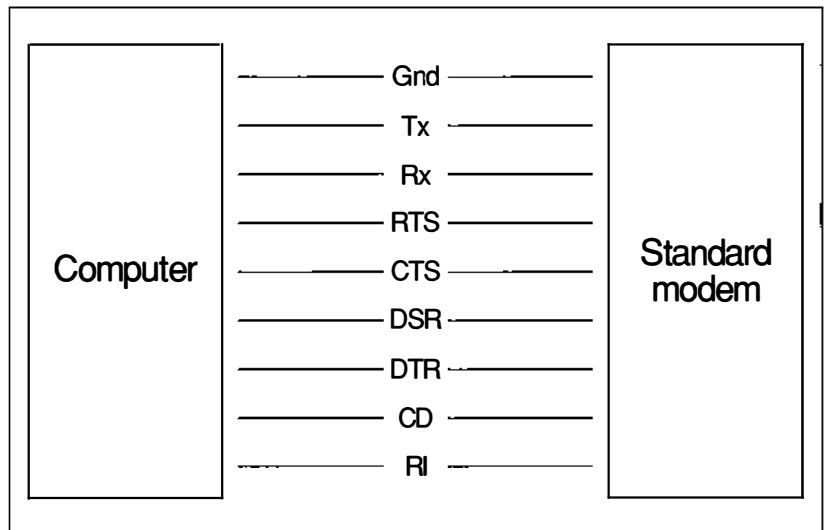
You should enable hardware handshaking in both directions using the **stty** utility:

```
stty +ihflow +ohflow </dev/ser1
```

Standard modems

Traditional modems are less sophisticated. Once a communications channel is established at one of the baud rates supported by that modem, all subsequent communication over the modem link takes place at exactly that baud rate. In other words, all transmitted data is simply passed through the modem link at the same rate that it's presented. Although these modems don't typically support hardware flow control, they often won't function properly unless all four flow control lines are connected.

A nine-pin straight-through cable is also used for these modems.



Cabling assignments for linking a computer to a standard modem.

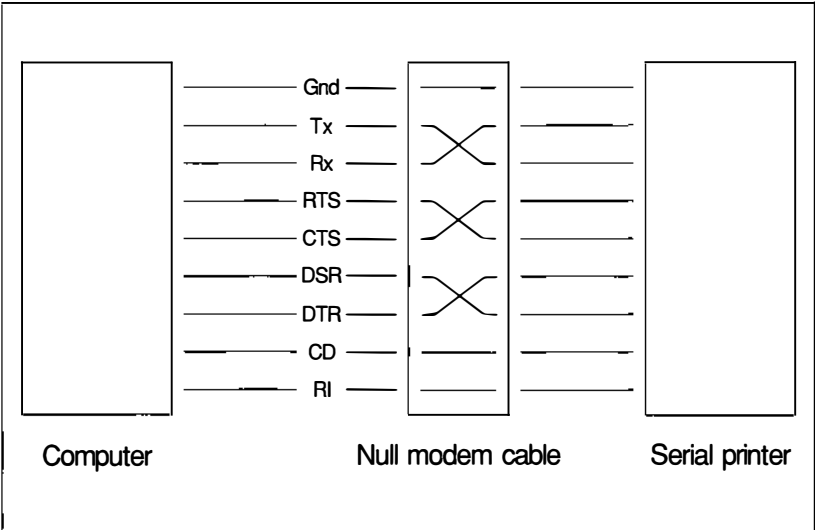
No flow control need be enabled with **stty** when using these modems.

Serial printers

Serial printers are usually *bidirectional* devices. Data flows from computer to printer as expected, but since printers can't keep up with the host computer, serial printers often use software flow control to

regulate the flow of data. In other words, they transmit XON and XOFF characters back to the computer. Some printers use the hardware handshaking lines for this purpose, some support both forms of flow control.

To be safe, you should connect all nine signals, although printers that support only software flow control may function just as well with a three-wire cable (Rx, Tx, and Gnd). Also, since printers are usually configured as Data Terminal Equipment (DTE)—just like the host computer—you may need to use a null-modem cable.



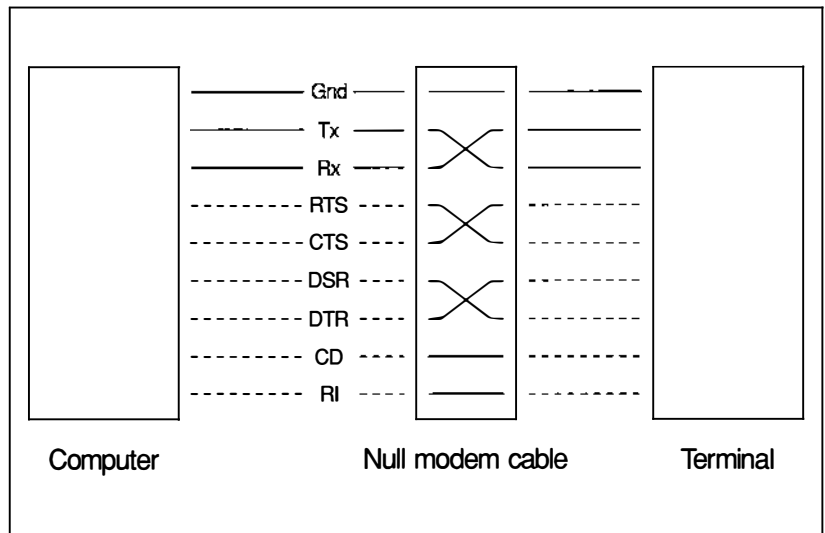
Cabling assignments for linking a computer to a serial printer.

If the printer uses:	you use:
software flow control	<code>stty +osflow </dev/ser1</code>
hardware flow control	<code>stty +ohflow </dev/ser1</code>
both software flow control and hardware flow control	both <code>stty</code> options

Terminals

Terminals operate with or without flow control and at a fixed baud rate. Unlike printers, terminals can usually keep up with the host computer at a supported baud rate. Thus, a simple three-wire cable is usually sufficient, although you can still use a nine-wire cable.

Like the host computer, terminals are normally configured as DTE devices, so a null-modem cable is usually required.



Cabling assignments for linking a computer to a terminal.

Configuring terminals for users

QNX can be used as a full-function time-sharing system. Many users can be connected to some or all of the computers in a QNX network via hardwired terminals or through modem links to terminals at remote sites.

Assuming that the terminal/modem is properly configured, you'll also need to provide a mechanism for users to "log in" to the system.

Simple example

Assume a terminal, properly connected and configured, on the serial port `/dev/ser1`. The simplest way of allowing a user sitting at that terminal to log in is to use this command:

```
ontty /dev/ser1 login
```

The user will be able to log in and execute commands. However, once logged off (i.e. once the user terminates that shell session), the user won't be able to log back in.

Automated login

To automate this login process, you can use the **tinit** utility. This utility will watch a specified device, waiting for any key to be pressed on that terminal. It will then start a **login** automatically.

When the user has logged in and then logged off again, **tinit** will once again wait for another key to be pressed. To start logins when a data key is pressed on two serial devices called **/dev/ser1** and **/dev/ser2**, you can use the following command:

```
tinit -T /dev/ser1 /dev/ser2 &
```

Suppose instead you wanted logins to be automatically started without first waiting for the user to press a key. You would use the **-t** option:

```
tinit -t /dev/ser1 /dev/ser2 &
```

Launching custom applications

You don't have to always start up **login**. You can specify any program or even specify a different program on each device. In some environments, a "canned" application is always expected on a given terminal.

For more information, see the documentation for **tinit** in the *Utilities Reference*.

Dial-up access

It's fairly easy to configure hardwired terminals, since all possible parameters are known. Modems, on the other hand, present special problems since they can have different configurations at different times. For example, most modems will answer calls at one of several baud rates (300, 1200, 2400, etc). You can't always know in advance the baud rate of an incoming call.

A special utility called **modem** is provided in QNX to respond to modems. Used in conjunction with **tinit**, **modem** can provide excellent dial-up capabilities. A typical dial-up system using QNX might have several serial ports (**/dev/ser1**, **/dev/ser2**, etc.) and might use the following command to permit dial-up access through Hayes-compatible modems:

```
tinit -c modem -t /dev/ser1 /dev/ser2 ... &
```

The **tinit** utility will automatically launch **modem** on each of the serial lines. When communication is established, **modem** will do the following:

- answer the phone
- determine and set the proper baud
- *exec* into **login**

When the user either logs off or hangs up, **tinit** will once again launch a new **modem**, which will wait for another call.

For more information, see the documentation for **modem** in the *Utilities Reference*.

The **terminfo** database

Full-screen programs in QNX have access to a database of terminal capabilities known as the **terminfo** database, which is implemented as a collection of files in the **/usr/lib/terminfo** directory.

A separate file exists for each terminal supported in QNX. These files use an industry-standard format, so can be shared between different UNIX systems. Your QNX system will have been shipped with several of these files, with names such as:

```
/usr/lib/terminfo/q/qnx      (the QNX console)
/usr/lib/terminfo/v/vt100    (a VT100 terminal)
```

A QNX program will access the appropriate file in the database, provided that the **TERM** environment variable has been set.

If you don't have a **terminfo** file for your type of terminal, there are several things you can try:

- access the Quantum Update Service (QUICS) and download a **terminfo** file (if it exists)
- access another UNIX system and look for the appropriate **terminfo** file
- use **infocomp** and **tic** to make your own **terminfo** file based on the reference manual for your terminal

Defining terminal type

Full-screen programs use the **TERM** environment variable to determine which type of terminal is currently used. They also automatically adjust their behavior to properly display data and interpret keys from that type of terminal.

This all works as expected, provided that the **TERM** environment variable is properly defined.

For hardwired terminals running canned applications, you can preset this environment variable when the application is launched. For example:

```
TERM=vt100 ontty /dev/ser1 custom_application
```

However, users logging in via **login** through **tinit** represent a special problem. If the type of terminal is known, then **tinit** can be told to define this environment variable before launching **login**:

```
tinit -c login TERM=vt100 -t /dev/ser1 &
```

Unless you restrict dial-up access to only a particular type of terminal, modem users will need to run a special program to query the user for his terminal type. In QNX, such a program is provided called **termdef**. This program will display all terminal types supported on this system and will ask the user to select one. Once a terminal type has been selected, **termdef** will set the **TERM** environment variable appropriately, then *exec* into **login**.

```
tinit -c "modem -c termdef" -t /dev/ser1 &
```

Troubleshooting serial device problems

The following table describes actions you can take if you encounter some of the common problems involved in connecting serial devices.

Problem	Probable Cause	Remedy
Data characters are unrecognizable	Wrong baud rate or parity	Use stty to set correct baud rate and/or parity
Some characters are fine, some aren't	Wrong parity	Try different parity using stty
No data can be sent or received	Wrong cable	Check cables; use null-modem if necessary
	Wrong I/O ports	Check hardware settings and verify correct parameters to Dev.ser
	Interrupt conflict	Change interrupt on adapter card
Data is received and transmitted only when another serial port is in use	Interrupt conflict	Check hardware interrupts and Dev.ser startup parameters; make sure that two serial adapters are <i>not</i> using the same IRQ
Data is lost occasionally	Flow control supported, but not enabled	Determine the type of flow control supported by the device, and enable with stty (ihflow , ohflow , isflow , and osflow)
	Flow control not supported	Reduce baud rates and/or increase stop bits; if only received data is lost, specify larger input buffer to Dev.ser (-I option)
	Cabling problems	Make sure cable is well grounded and not too long; also, verify that all RS-232 wires in the cable are connected



Chapter 4

Print Spooling

In this Chapter...

Sharing devices in a network

Print spooling

Configuring the spooler

Sharing devices in a network

QNX encourages the distribution of resources throughout a local area network with few artificial boundaries. Every device (disk, modem, or printer) connected to any computer is, by default, a *shared* resource. Any program running on any computer has equal access to all of these devices, whether they are connected to the same computer (local) or not (remote).

This transparent sharing of resources is a mixed blessing. Users of a QNX network will enjoy the ease of access to network resources. The system administrator, on the other hand, must take steps to prevent misuse and uncontrolled access to some resources.

Some devices, such as hard disks, can be readily accessed by many users concurrently. But other devices, such as printers and modems, can't be used by more than one user at a time. Access to such resources therefore requires some sort of *sequencing* or *spooling*.

Print spooling

QNX provides a set of spooling services to allow convenient access to shared resources such as printers. Spooling is a simple process where data is written (spooled) into a disk file rather than directly to a hardware device. This process can occur for many users simultaneously. You can set up a system spooler process that takes print files and sends them to the printer one at a time.

The QNX print spooling server is:

lpsrvr

The QNX print spooling utilities are:

lp	send files to a printer
lprm	remove jobs from the printer queue
lpc	control print spooler queue
lpq	display print queue status

The server (**lpsrvr**) is documented in Part II of this guide. For documentation on the other print utilities, see the *Utilities Reference*.

Submitting print jobs

Users submit jobs to the spooler using the **lp** utility. For example, the following command will cause the file **report** to be inserted into the default print queue and (ultimately) printed:

```
lp report
```

In systems where more than one print queue is available, you can specify the symbolic name of a print queue:

```
lp -n txt report
```

The spooler for that symbolic printer will queue the submitted jobs and will ship them to the actual printer one at a time.

Querying print jobs

A user may wish to examine the printer queue using the **lpq** utility. A sample output from the **lpq** utility might look like this:

```
1:      steve  [job #39]    1400 bytes  lalist.doc
2:      aboyd  [job #42]    2312 bytes  netdrv.c
```

This utility lets the user determine when any jobs submitted have actually been printed. It also provides the print job ID for use in other **lp** utilities.

Cancelling print jobs

The **lprm** utility lets the user remove jobs from a print queue. Jobs may be removed by explicitly specifying the job ID number(s). If the state of the system print queue was as shown above, **steve**'s job (job #39) could be cancelled with the command:

```
lprm 39
```

If job #39 was currently being printed, the printing would be abandoned. The success of abandoning current print jobs may vary with the type of printer you're using, since some printers have large internal buffers.

The superuser may also remove all jobs belonging to a particular user. For example, all of **steve**'s jobs can be cancelled with the command:

```
lprm steve
```

Controlling the print queues

The **lpc** utility is a system administration tool for managing the system spoolers. It lets you perform many control functions, such as shutting down and starting up a queue or a printer. The following basic functions are provided:

- suspend/resume enqueueing of jobs
- suspend/resume dequeueing of jobs
- suspend/resume the currently printing job
- delete the current job
- rearrange the jobs in a queue
- move jobs to a different queue
- display the status of queues

Note that **lpc**'s functionality overlaps that of **lprm** and **lpq**. This is for convenience, since **lpc** may be used in an interactive mode.

Configuring the spooler

Before any print spooling can occur in a QNX system, you must run **lpsrvr**, the print pool server:

```
lpsrvr &
```

When started, the **lpsrvr** utility reads the **/usr/spool/lp/.setup** file to determine what resources it has available and how it is expected to manage them. Note that you must create this file.

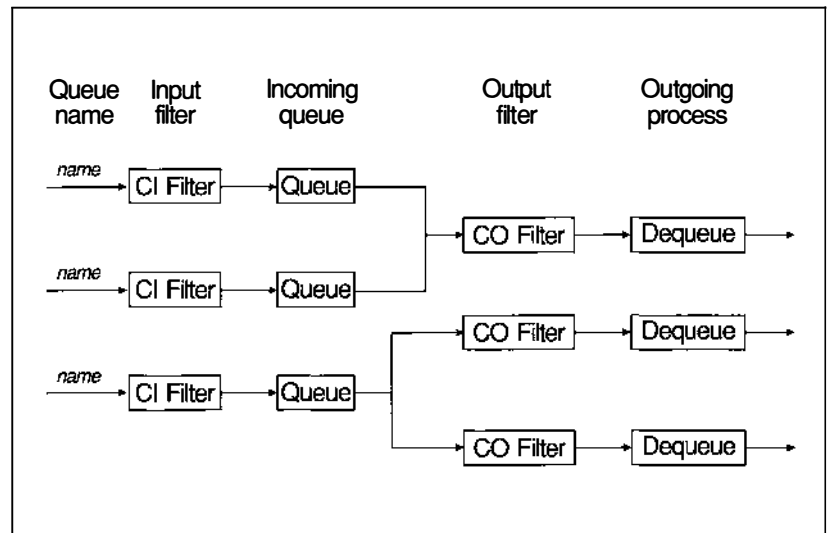
You can specify a different configuration file via **lpsrvr**'s **-c** command-line option:

```
lpsrvr -c /usr/spool/lp/custom &
```

The **lpsrvr** utility manages two primary objects: *queues* and *dequeues*. Queues are internal lists of data files that need to be printed. Dequeues are the processes that actually print the data.

You can specify a program to run that will serve as a filter to control how data is “copied in” to a queue, and another filter to control how data is to be “copied out” of a queue (and into a dequeue).

The following diagram illustrates the internal flow of data within the print spooler:



Jobs are submitted, via **lp**, into one of the queues based on a symbolic name. If the symbolic name isn’t specified when a job is submitted, the first entry in the configuration file will be used as the default queue.

A dequeue will process data from a queue based upon priority, then upon time of job submission.

Multiple queues may feed a single dequeue, in which case the dequeue will select the appropriate job from all the jobs in those queues.

Multiple dequeues may serve the same queue. In this case, the spooler will select a dequeue based upon availability.

Spool configuration file

When started, the spooler accesses a file to get its configuration information. If no file is specified on the command line, the spooler uses the `/usr/spool/lp/.setup` file. This configuration file defines queues, dequeues, and the relationships between them.

Queues and dequeues have symbolic names as well as a set of attributes. The format of each entry in this file is as follows:

name:attribute:attribute: ... :

If a single entry spans a line, the last character before the `<newline>` must be a backslash (`\`); all leading white space on the following line will be discarded. Comment lines start with a pound sign (`#`).

The *name* may contain only alphanumeric characters and slashes (`/`), and is limited to 48 characters. If the object being described is a *dequeue*, the name must start with a dash (`-`). The dash is for delineation only; it isn't considered part of the name.

Each *attribute* consists of a two-letter *key* in one of the following forms:

<i>key</i>	Boolean
<i>key#number</i>	Numeric
<i>key=string</i>	Character

All *numbers* are assumed to be decimal numbers, unless they start with a leading zero (meaning octal) or a leading `0x` (meaning hex).

All *strings* contain printable characters. The backslash (`\`) is a “special” character. It can be used to escape other characters. In particular, the colon (`:`) must be represented with:

`\:`

and a “real” backslash must be represented with:

`\\`

The following *keys* are defined:

ab=string	Executed when dequeue ab andons command for any reason
af=string	accounting file that the <i>queue</i> uses
ci=string	copy in command, to place the job on the <i>queue</i>
co=string	copy out command, used by the <i>dequeue</i>
de=string	despooler associated with this <i>queue</i> (or name of device)
dv=string	device that the <i>dequeue</i> uses
mn#numeric	minimum number of jobs for <i>queue</i> before flushing
mx#numeric	maximum number of jobs the <i>queue</i> will hold
na=string	name , a string that describes the <i>queue</i> or <i>dequeue</i>
ok=string	Command executed when dequeue completes normally
pr#numeric	priority to run this <i>queue</i> at (1-100); 100 is the highest
sp=string	spooler registered name for this queue
wa#numeric	wait this number of seconds before despooling each job
qn=string	After despooling, enqueue job on this queue

The following defaults are used if their corresponding attributes aren't specified:

ab	nil; no command is executed
af	accounting isn't performed (note that lpsrvr logging is unaffected)
ci	binary copy in of file; no transformation applied
co	binary copy out of file; no transformation applied
de	output is sent to standard output of lpsrvr
dv	output is sent to standard output of lpsrvr
mn	0; jobs are despooled as soon as possible
mx	no limit; queue limits are based upon memory and disk space
na	nil string
pr	50 is the default priority
ok	nil; no action is performed
sp	/qnx/SPOOLER ; registered only if <i>no</i> queue specifies a name
wa	0; jobs are despooled as soon as possible
qn	nil; delete job after despooling

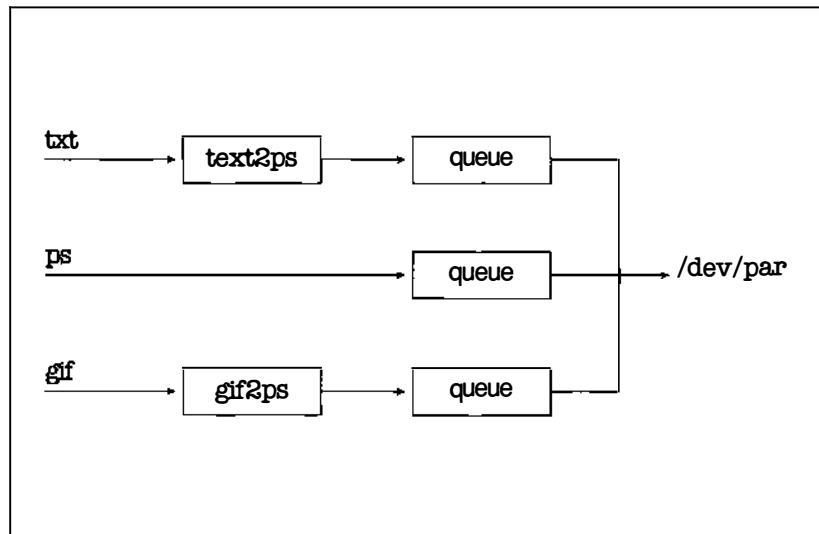
Since the *keys* are case-sensitive, Quantum reserves all keys formed by two lowercase letters. Any custom extensions may be safely implemented using uppercase or mixed-case keys. The spooler utilities will ignore any options they don't understand.

Example configuration
file

The following example shows a set of queues that share a common dequeue. The three queues are named:

- **txt** (text files)
- **ps** (PostScript files)
- **gif** (Graphic Interchange Format files)

The configuration is as follows:



Users send files with the **lp** utility to the appropriate *queue*, which converts the file appropriately, then prints the converted data on the printer (**/dev/par**). Since GIF files tend to be large, they are assigned a lower priority than the others.

The following example shows the flexibility of the spooler and demonstrates some important ideas.

```
# This is the spool setup file.
# First, the plain-text printer.
txt:de=/dev/par:af=/usr/spool/lp.log:\
    ci=text2ps $(fname) >$(spfile):pr#50:
# Second the postscript printer.
ps:de=/dev/par:af=/usr/spool/ps.log:pr#60:
# Third the gif.
gif:de=/dev/par:af=/usr/spool/gif.log:\
    ci=gif2ps $(fname) >$(spfile):pr#5:
# end of the spool setup file.
```

First, notice that in the `gif` queue, the `ci` key is used. This is important because the (imaginary) program `gif2ps` may require substantial time to complete. If the program were used as a “copy-out” filter, it would needlessly block the despooler while generating the image. With this configuration, other jobs may be despoiled while the image is being generated.

Second, notice that the `de` attribute specifies a device rather than a dequeue. Since no dequeue was specified, data is copied directly to the indicated device without further translation. A number of names were introduced in the `ci` commands. These are *metavariables* that the spooler will set as appropriate when it performs the action.

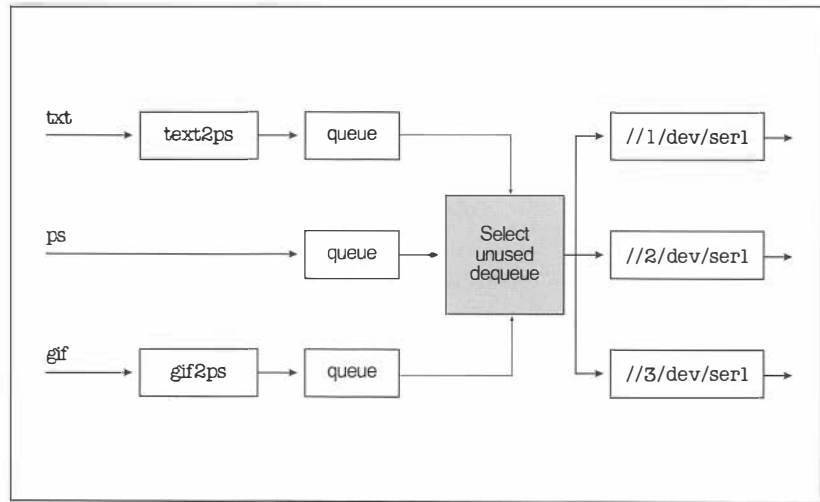
The following metavariables are available:

<code>\$(fname)</code>	the original file name submitted in the job
<code>\$(spname)</code>	the name of the spool data file
<code>\$(username)</code>	the login name of the user who submitted the job
<code>\$(userid)</code>	the numeric userid of the user who submitted the job
<code>\$(qname)</code>	the name of the queue the job currently belongs to
<code>\$(deqname)</code>	the name of the dequeue the job currently belongs to
<code>\$(device)</code>	the name of the device the job is scheduled on
<code>\$(jobid)</code>	the job ID number of this job

In addition, all the *keys* defined earlier can be referenced as metavariables. For example, `$(ci)` will expand to the name of the “copy in command” (`ci=string`).

A more complicated example

The following example is a further refinement of the above setup with some additional features required for a larger configuration. There are now three laser printers, all PostScript, located at different locations in the building (connected to `//1/dev/ser1`, `//2/dev/ser1`, and `//3/dev/ser1`). The configuration looks like this:



Here's the configuration file to set this up:

```

# This is the spool setup file.
# First, the plain-text printer:
txt:de=lp1,lp2,lp3:af=/usr/spool/lp.log:\
    ci=text2ps $(fname) >$(spfile):pr#50:
# Second the postscript printer
ps:de=lp1,lp2,lp3:af=/usr/spool/ps.log:pr#60
# Third the gif
gif:de=lp1,lp2,lp3:af=/usr/spool/gif.log:\
    ci=gif2ps $(fname) >$(spfile):pr#5:
# Now define the laser printers.
-lp1:dv=//1/dev/ser1:\
    ok=echo job $(jobid) completed | mail $(username):\
    ab=echo job $(jobid) aborted | mail $(username):
-lp2:dv=//2/dev/ser1:\
    ok=echo job on $(na) completed | mail $(username):\
    ab=echo job on $(na) aborted | mail $(username):
-lp3:dv=//3/dev/ser1:\
    ok=echo file $(fname) printed ok | mail $(username):\
    ab=echo file $(fname) aborted | mail $(username):
  
```

The above configuration uses the same three “logical” printers described earlier (**txt**, **ps**, and **gif**). These logical printers, however, now feed three physical printers. The server selects a free printer from the set of dequeues (**lp1**, **lp2**, **lp3**) The job will be dispatched on the first available printer, and the printers will be examined for availability in the order they are specified in the configuration.

In this example, when a job completes normally, a mail message indicating success is sent to the submitter. When a job completes abnormally, a mail message indicating failure is sent to the submitter.



Chapter 5

Setting up a Second Filesystem

In this Chapter...

Partitioning the pathname space

Setting up a DOS filesystem

Introduction

A filesystem rooted at slash (/) may be composed of one or more *physical filesystems* grafted together. A physical filesystem is a separate disk or partition. One of the physical filesystems is typically assigned to be the root (/), while the other filesystems are mounted as subdirectories. These subdirectories don't exist on disk but rather in an independent space called the prefix tree. The prefix tree maps pathnames to I/O managers, which in turn map to disks and devices. For more information on the prefix tree, see the chapter on "I/O Namespace" in the *Users Guide*.

Partitioning the pathname space

The following examples should help clarify how the pathname space is partitioned. We'll look at the following configurations:

- a hard disk and a floppy
- two hard disks
- two or more QNX partitions
- local and remote hard disks



In these examples, we assume that **Fsys** as well as the appropriate drivers are running, and that the **mount -p** command has been done on your hard disk.

A hard disk and a floppy

The hard disk is mounted as slash (/) and forms the root of the filesystem. The floppy is mounted as **/fd0**:

```
mount /dev/hd0t77 /  
mount /dev/fd0 /fd0
```

Any reference to a pathname starting with **/fd0** will be directed to a QNX filesystem on the floppy. For example:

```
ls -aR /fd0    Show all files on the floppy
```

Two hard disks (same node)

The first hard disk is mounted as slash (/) and forms the root of the filesystem. The second hard disk is mounted as **/usr2**:

```
mount /dev/hd0t77 /
mount /dev/hd1t77 /usr2
```

Any reference to a pathname starting with **/usr2** will be directed to a QNX filesystem on the second hard drive. For example:

```
ls -aR /usr2
```

Show all files on second hard drive

Two or more QNX partitions

You can have three QNX partitions on a single hard drive. The primary partition should be type 77 (see **fdisk** utility). A second partition should be assigned type 78, and a third should be assigned type 79. For example:

```
mount /dev/hd0t77 /
mount /dev/hd0t78 /usr2
```

Local and remote hard disks

In a local area network, you might have disks with QNX filesystems on more than one machine. You may configure these filesystems to be:

- independent
- primary/secondary
- linked independent

Independent

In this configuration, you treat each machine as an independent, self-contained filesystem. To access a file on a remote machine, you would precede a pathname with the remote machine's node number. For example:

```
/etc/motd          file on local filesystem
//10/etc/motd      file on remote filesystem (node 10)
```

The ability to specify a filesystem for a particular node will always work and is the most general mechanism for accessing remote files.

Primary/secondary

In this configuration, you treat one filesystem as the primary and you mount the second filesystem as a subdirectory under the primary. For example, assume node 1 has the primary and node 2 has the secondary mounted as **/usr2**. Node 1 boots from hard disk and has

its root set to its local disk by a **mount** utility built into the operating system image (with **Fsys** and a driver). Its system initialization file would invoke the **prefix** utility to mount the remote filesystem as follows:

```
prefix -A /usr2=//2/usr2
```

Node 2 boots over the network from node 1 and has its filesystem root set to **/=//1/** using the **-r** option of the **sinit** utility built into the operating system. Its system initialization file would invoke the **mount** utility to mount the local filesystem as follows:

```
mount /dev/hd0t77 /usr2
```

The Filesystem Manager (**Fsys**) and its driver may be built into the image, but they are more likely started from the system initialization file. In other words, node 2 boots like a simple diskless workstation, then starts its filesystem after booting.

Both nodes 1 and 2 will access the filesystem on node 1 as **/** and the filesystem on node 2 as **/usr2**.

Linked independent

In this configuration, you treat each machine as an independent self-contained filesystem, but you link portions of them together via the **prefix** utility. For example, assume that the filesystem on node 1 has a **/usr1** directory while the filesystem on node 2 has a **/usr2** directory. You could map each user directory into the other filesystem's space as follows:

On node 1:

On node 2:

```
prefix -A /usr2=//2/usr2    prefix -A /usr1=//1/usr1
```

Other than this link, each filesystem is self-contained with its own copies of **/bin** etc. The advantage is greater redundancy: if one department uses node 2, and node 1 goes down, the department using node 2 can continue to work (except that the files under **/usr1** won't be available).



In the examples in this section, you'll notice that local filesystems are associated with block special files (e.g. `/dev/hd0t77`), while remote filesystems are associated with pathname prefix mappings. This prefix mapping redirects requests to a remote filesystem that will be associated with a remote block special file.

Setting up a DOS filesystem

The **Dosfsys** filesystem manager provides totally transparent access to DOS files and directories that reside on a DOS disk. **Dosfsys** can support up to eight drives (**a** to **h**).

You can create files on DOS disks, read them, write to them, and delete them with most QNX programs. You can also create, read, and delete directories on DOS disks with standard QNX utilities, such as **mkdir**, **ls**, and **rmdir**. Most standard QNX utilities will work with DOS files, provided that the DOS file structure allows for the functionality required by the utility.

Your own programs will also be able to process DOS files just as they process QNX files, by using the standard I/O functions such as *open()*, *read()*, *write()*, *close()*, *seek()*, etc. When you read DOS directories, they'll be presented to you in QNX format.

Invocation modes

Dosfsys has three invocation modes:

```
Dosfsys [-S|-s] [-m] [dos_drive=qnx_drive [,R]] ... &
```

```
Dosfsys -i [-n node] [dos_drive_path]...
```

```
Dosfsys -x [-n node]
```

The **-i** option lets you get information about the currently adopted DOS drives. The **-x** option terminates the **Dosfsys** server. When this command is issued, no new *open()* requests will be accepted, and the server will terminate once all active files (i.e. files that are still open) are closed.

If you don't specify **-i** or **-x**, **Dosfsys** will start up and try to adopt the specified drives.

Starting Dosfsys

When you start **Dosfsys**, it performs the following actions:

- opens the specified drive(s)
- adopts the root DOS name (**/dos**)
- registers the name **qnx/dosfsys** with the local Process Manager

If no options have been specified, or if either **-S** or **-s** has been specified, **Dosfsys** scans the **/dev** directory for valid DOS drives to adopt. It looks for the following device names:

Device name on disk	Drive assignment
/dev/fd0	/dos/a
/dev/fd1	/dos/b
/dev/hd0t1	/dos/c
/dev/hd0t4	/dos/c
/dev/hd0t6	/dos/c
/dev/hd1t1	/dos/d
/dev/hd1t4	/dos/d
/dev/hd1t6	/dos/d

Dosfsys name adoption

Dosfsys can adopt up to eight drives (**a** to **h**). As mentioned above, **Dosfsys** will adopt the name **/dos** as a system prefix. It will also manage each specific drive's name as **/dos/a**, **/dos/b**, etc. These names aren't registered in the system prefix tree but are kept internally by **Dosfsys**. This will be transparent to the user except for the fact that the user won't be able to create files or directories at the **/dos** root.

DOS devices

A DOS device could be one of the following:

- a DOS partition on a hard disk
- a floppy diskette
- an image of a DOS partition or diskette

To create an image of a DOS diskette or DOS partition, you use the QNX **cp** utility. For example, to copy an image of a DOS floppy in your floppy drive 0, you could use the following:

```
cp /dev/fd0 /usr/qnx/dosa
```

and then invoke **Dosfsys** as follows:

```
Dosfsys a=/usr/qnx/dosa &
```

The same could be done with a hard disk partition. **Dosfsys** will handle these images just as it would the actual device.

For all *non-removable* devices, **Dosfsys** immediately reads the DOS boot parameter block (BPB) as well as part of the file allocation table (FAT) at startup. For *removable* devices, the BPB and the FAT are read only when the drive is being accessed.

When **Dosfsys** has a *non-removable* device open, the device is locked for READ/WRITE so no other process can access this device without going through **Dosfsys**. Removable devices are kept open and locked only during accesses (e.g. during reading or writing to the disk). Note that unless you specify the **R** option, all drives have READ/WRITE access.

DOS version support

The **Dosfsys** manager supports all DOS partitions formatted with DOS 2.1 or later, including standard primary DOS partitions and DOS large partitions (DOS 4.0 >32M). Hard disks, 5¼" floppies, and 3½" floppies are supported.

The **Dosfsys** manager also supports DOS extended partitions (type 5). To make these extended partitions accessible to **Dosfsys**, you must mount them explicitly via the QNX **mount** utility. To do this, you use **mount** as follows:

```
mount -p /dev/hd0t5
```

This will mount all “real” partitions within the DOS extended partition. For example, after issuing the above **mount** command, you might see the following drive mounted:

```
/dev/hd0t5t1
```

The ending **t1** indicates that this is a type 1 DOS partition within the type 5 extended partition—this is the way DOS handles extended partitions.

This drive can now be adopted by **Dosfsys**. For example:

```
Dosfsys -s e=/dev/hd0t5t1
```

Since **Dosfsys** doesn't automatically scan the `/dev` directory for *extended* partitions (scanning is done when `-s` or `-S` is specified), you must specify the drives in these partitions on the command line. Once adopted, these drives will be accessed in the same way as "standard" DOS drives.

DOS partition types

These are the standard DOS hard disk partition types:

Partition type:	Description:
1	DOS primary partition (12-bit FAT)
4	DOS primary partition (16-bit FAT; ≤ 32M)
5	DOS extended partition (DOS 3.3 or later)
6	DOS primary partition (DOS 4.0 or later; >32M)

DOS text files

DOS uses a structure for text files that's different from the one used in QNX (by text files we mean line-oriented files containing lines of ASCII text separated by line separator sequences). In DOS, each line of a text file is terminated with a carriage return sequence (CR/LF), while in QNX each line is terminated by a line-feed character (LF). The **Dosfsys** manager doesn't translate these files. All files are treated "as is." Therefore, you may need to use the QNX **textto** utility to convert your text files before copying them to or from QNX and DOS disks.

Note also that the text files created by some DOS programs may contain a SUB character (^Z) as the last character of the file. This is also treated as is.

DOS binary files

Since **Dosfsys** doesn't translate the contents of files, binary files may be copied to or from the QNX/DOS partitions as is.

QNX to DOS character and name mapping

DOS doesn't permit certain characters within a filename; it also doesn't allow certain filenames.

In DOS, you can't include any of these characters in a filename:

/ \ [] : * | + = ; , ?

In DOS, you can't use these filenames:

CLOCK\$	CON	PRN
NUL	AUX	COM1
COM2	COM3	COM4
LPT1	LPT2	LPT3

If you attempt to create a file that contains one of the invalid DOS characters or whose filename is invalid, you will be denied access.

Since all DOS filenames and filename characters are allowed under QNX, no validation is required on these filenames.

DOS also maps all alphabetical characters to uppercase, so **Dosfsys** maps these characters to uppercase when creating a DOS filename; it maps a filename to lowercase when returning the filename to a QNX application.

DOS volume labels

DOS uses the concept of a volume label, which is an actual directory entry at the root of a DOS filesystem. To distinguish between the volume label and an actual DOS file, **Dosfsys** places an equal sign (=) as the first character of the volume label name. **Dosfsys** treats this directory entry as a zero-length, read-only file whose permissions cannot be changed.

DOS/QNX permission mapping

DOS doesn't support all of the permission bits that QNX does. The DOS attribute bits are as follows:

READ_ONLY
HIDDEN
SYSTEM
VOLUME_LABEL
DIRECTORY
ARCHIVE

Dosfsys uses the following mapping logic to handle the QNX to DOS attribute-bit translations:

- if the entry is a directory, set the DOS DIRECTORY file bit
- if the entry is a file, and if all of the QNX WRITE bits are off, set the DOS READ_ONLY bit

The following mapping logic is used to handle the DOS to QNX permission-bit translations:

- set the QNX READ permission bits for user, group, and other
- if the entry isn't a volume label, and if the entry isn't read only, set the QNX WRITE permission bits for user, group, and other
- if the entry is a directory, set the QNX DIRECTORY and EXECUTE bits for user, group, and other
- if the entry is a file, set the QNX REGULAR FILE bit



If a file is written to, the DOS ARCHIVE bit is also set.

File ownership

Although the DOS filesystem doesn't support user IDs and group IDs, **Dosfsys** will not return an error code if an attempt is made to change the group ID or user ID with the **chown** utility or *chown()* library function. An error isn't returned because a number of utilities make use of the *chown()* library function, which could result in many error messages being displayed.

All files under **Dosfsys** are owned by the superuser (uid=0, group=0) with access to all.

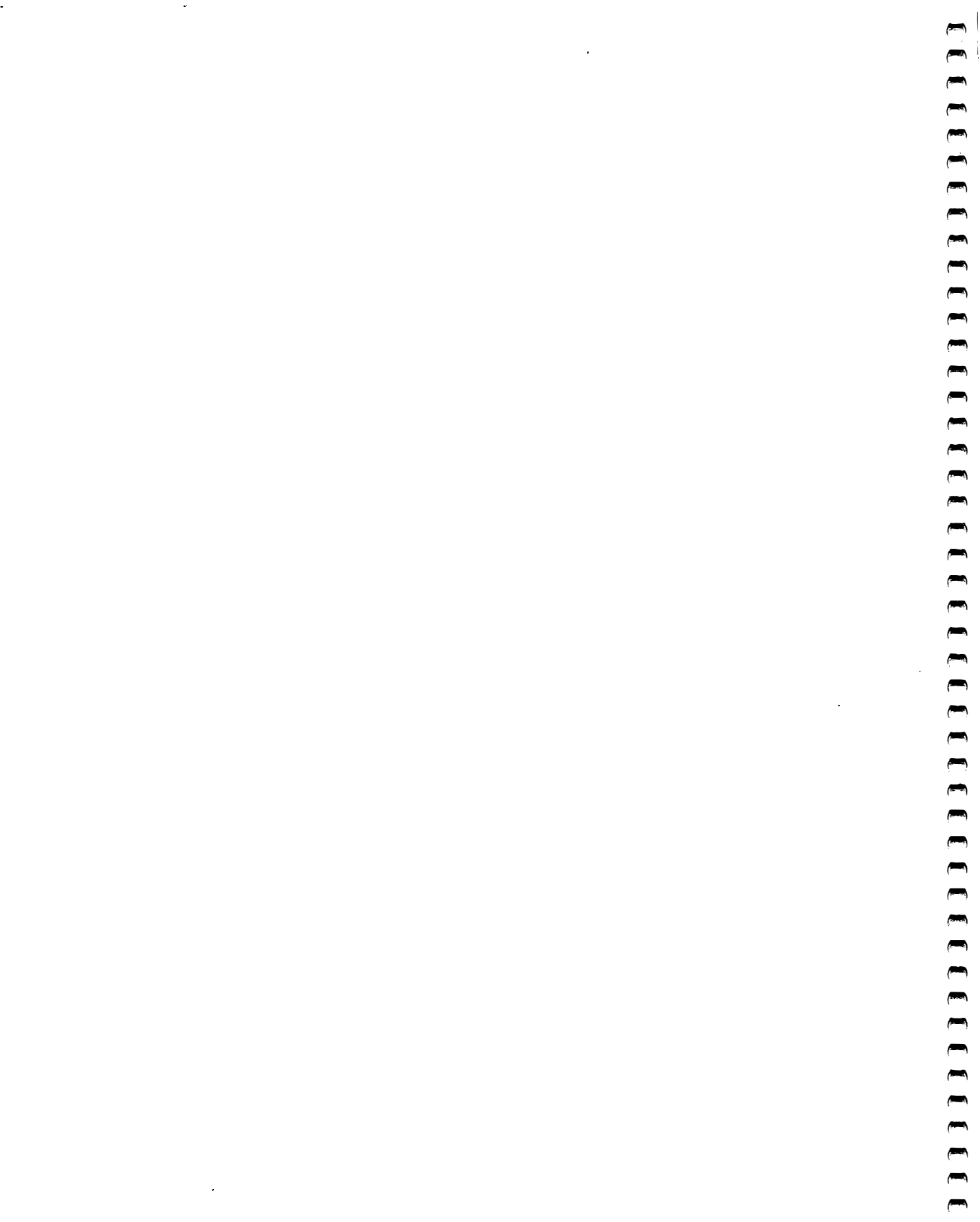
Error codes returned by **Dosfsys**

If a request made to **Dosfsys** isn't supported, the EOPNOTSUPP error code will be returned to the application making the request. Examples of requests not supported by **Dosfsys** include:

LINK
BLOCK_READ
BLOCK_WRITE
MOUNT_PARTITION
MOUNT_RAMDISK
PIPE
DISK_GET_ENTRY
RECORD LOCKING

If **Dosfsys** detects a corrupt filesystem, it will return EBADFSYS, at which point you may wish to run the **CHKDSK** utility under DOS to correct the problem.

The DOS filesystem structure is such that the root directory's size is fixed at format time and cannot be resized. If it does become full, an error will be returned (ENOSPC).



Chapter 6

Making Backups

In this Chapter...

Introduction

When to back up

Backup formats

Backup media

Compression

Archive examples

Introduction

This section deals with making a copy of your data to guard against hardware, software, or human error that may destroy the original. If your data is important to you, you should regularly perform a set of backup procedures that would allow you to restore lost data with minimal cost to you in time and money. Remember: hard disks *do* fail and people *do* make mistakes. It's too late to start a backup after your data is gone!

You can back up an entire filesystem or only portions of it. Users may elect to back up their own files only, usually to floppy diskette. To back up large portions of the filesystem with files owned by many users, you'll need to have read permissions on these files. The superuser (**root** login account) has such privileges.

When to back up

You should back up often enough so that you can recover data that is still current or can be made current with minimal work. In a software development group, this may range from a day to a week. Each day of out-of-date backup will generally cost you a day of redevelopment. If you're saving financial or point-of-sale data, then daily or even twice-daily backups are common.

Backup formats

QNX supports a variety of backup formats that can be classified into two groups:

- archives
- regular filesystems

An *archive* consists of one or more files, merged into a single unit with its own directory of contents. This single unit can be saved either to a regular QNX file or to a raw block device like a floppy or tape.

Saving to a *regular filesystem* simply involves copying the files. In this case, the destination must be a device with a mounted QNX filesystem on it.

Archive backups

QNX supports three major archive utilities:

- **cpio**
- **tar**
- **pax**

Both **cpio** and **tar** are implemented as links to **pax**, which is capable of reading and writing both **cpio** and **tar** formats. Both **cpio** and **tar** are common on UNIX systems. The **pax** utility is a cover utility for **cpio** and **tar**, so it doesn't support its own archive format. By default, **pax** will use the **tar** format when creating an archive.

The **pax** utility will detect when you've reached the end of a media on a volume and will prompt you to insert the next volume to be used for the save. The result is a backup spanning several media (diskettes, tapes, etc.).

Unfortunately, the **tar/cpio** format does *not* label the media with volume IDs. If you mixed up your media or inserted the wrong one out of order, you would end up restoring bad data.

To safeguard against this possibility, QNX is shipped with the **vol** utility, which labels each volume with its sequence number and therefore prevents you from inserting out-of-sequence media. The **vol** utility will, by default, step over block one of all media. This is important for floppies and cartridge disks that contain a QNX signature in block one. This signature contains the size of the diskette (360K, 1.2M, etc.) and allows for automatic remounting of removable media by the filesystem. Quantum ships its distribution diskettes using **pax** to create an archive, **freeze** to compress the data, and **vol** to write to the floppies.



If you wish to save data for restoration on a UNIX system other than QNX, don't use **freeze** or **vol**, since you won't find those utilities to do the restore at that end. Instead, use **pax** to save and restore directly to the target media.

Filesystem backups

You back up to a filesystem by copying files, probably with the **cp** utility. If your destination media is a floppy diskette, the **cp** utility will prompt you for more diskettes, but remember that no single file can be larger than the diskette size. If you wish to back up to floppy, we recommended that you use one of the archive utilities.

Backup media

Your choice of backup media will be determined by available hardware and cost. There are four common choices:

- floppy
- tape
- removable disk
- fixed disk

Floppy

Floppies are the most widely available device for backups. Their major shortcoming is their limited size. Since the QNX **pax** and **vol** utilities let you span media, your only concern will be having to feed several floppy diskettes into the drive. If you have to deal with more than four or five floppies, this will make the procedure unpleasant enough that backups may occur less frequently. You might want to consider compressing your data as described below.

In order to back up/restore from a floppy diskette, you must make sure the floppy driver has been started. This is described in Part II, *Managers, Drivers, & Utilities* (under **Fsys.floppy**). For most people, the following command line will start the driver (assuming that **Fsys** is already running), which will then create a block special file for each floppy:

Command line	Block special files
Fsys.floppy	/dev/fd0 (floppy drive A) /dev/fd1 (floppy drive B)

If you use an archive utility for your backups, it will read and write directly to the floppy block special file. If you elect to use the **cp** utility, you'll have to mount a filesystem on the block special device. For example:

```
mount /dev/fd0 /fd0
```

Remember to format and initialize new floppies. The following example would initialize a high density 5¼" floppy diskette:

```
fdformat /dev/fd0 -s 1.2m  
dinit /dev/fd0
```

You may now treat the floppy as a QNX filesystem mounted as **/fd0**.

Tape

QNX provides a driver for the QIC-02 interface standard. This is a standard ¼-inch cartridge tape that can hold from 60 to over 300 megabytes of data on a single tape.

To back up/restore from a tape, you must make sure the tape driver has been started. This is described in Part II of this guide under **Fsys.qic02**. For most people, the following command line will start the driver, which will then create a block special file for the tape:

Command line	Created block special files
Fsys.qic02	/dev/tp0

The archive utilities will read and write directly to the tape's block special file. You can't mount a filesystem on this type of block special file.

When the QIC-02 driver receives a request to read or write, it does so by starting at the current tape position. If you're starting a new backup, you'll need to erase and rewind the tape. A number of tape position and control functions are provided by the **tape** utility

described in Part II of this guide. For example, the following command would erase, then rewind the tape in preparation for an archiving procedure:

tape erase

Removable disk

Removable hard disks are popular in both magnetic and optical formats. The optical formats hold more data, but they're slower. Most units use a SCSI interface to the computer, so you may want to consider making your internal fixed disk a SCSI drive. Unlike floppies and tapes, a removable hard disk lets you avoid using the archive utilities like **pax**. Instead, you'd likely use **cp** to copy your data to a real filesystem on the cartridge. This allows you to recover single files very easily and quickly.

Fixed disk

You can place a second hard disk in your machine or you can back up to a hard disk on another machine in the network. This will give you an online backup. However, it's still a good idea to maintain off-site storage.

Compression

You can use a compression utility to reduce the amount of space required to store data. The amount of compression will depend on the nature of the data you're saving. Some databases containing large amounts of null data may compress up to 90%. Other data might compress less than 10%. Although compressing may save media space, it has two side effects:

- Compressing requires a fair amount of computation and may slow down the saving of data.
- You can't recover compressed data should a defect develop, such as a bad block, on your saved backup media. Potentially, all data following a bad block can be lost. In comparison, uncompressed data will simply have a "hole" in the middle.

You may use the **freeze** utility to compress your data and the **melt** utility to restore it. Both these utilities will act on a stream of data as well as on files. This ability to act as a filter lets you connect them to

one of the standard archivers via a pipe. For example, Quantum distributes its operating system in compressed form on floppies using **pax**, **freeze**, and **vol**.

Archive examples

Compressed floppy archive

Collect files under **/usr** into a **tar** format archive, compress the archive and write it out to as many floppies as are needed, adding sequence numbers to the diskettes:

```
pax -w -x ustar /usr | freeze | vol -w /dev/fd0
```

Read data off floppies, uncompress it back into an archive and restore the files:

```
vol -r /dev/fd0 | melt | pax -r
```

UNIX-compatible floppy archive

Save files under **/usr/brianc** in a **tar** format archive for restoration on a UNIX system:

```
pax -w -x ustar /usr/brianc >/dev/fd0
```

Save files under **/usr/brianc** in a **cpio** format archive for restoration on a UNIX system:

```
pax -w -x cpio /usr/brianc >/dev/fd0
```

Restore data in a **tar** or **cpio** format archive from another UNIX system and place all files under **/usr/unix**:

```
pax -r -s ",/,usr/unix/," </dev/fd0
```

Tape archive

Start a new tape archive and save all files onto a QIC-02 tape:

```
tape erase  
pax -w / >/dev/qic02
```

Append files that have changed since the date of the file **lastsave** to the end of an existing archive tape. After the save, update the time of **lastsave**:

```
tape forward
find / -newer lastsave | pax -w >/dev/qic02
touch lastsave
```

Restore all the files on a tape that are under the directory
/usr/robo:

```
tape rewind
pax -r "/usr/robo/*" </dev/qic02
```

Cartridge/optical

Copy all files from the filesystem on node 1 to the filesystem on
node 2:

```
cp -Rp //1/ //2/
```

In the following example, the disk on node 2 is probably a very large
optical with sufficient room for many disk contents. A full backup is
done each Friday and a partial backup of modified files could be
done each day of the week:

```
cp -Rp //1/ //2/fri
cp -Rp -a date //1/ //2/mon
cp -Rp -a date //1/ //2/tue
.
.
.
```

Chapter 7

Building a Custom Operating System Image

In this Chapter...

Introduction

Constructing a build file

Introduction

QNX is a modular operating system composed of a microkernel and one or more processes that provide services. For example, a process named **Fsys** provides filesystem services, while a process named **Dev** provides device services.

When you build an operating system image, you select those services you wish to be available immediately after boot, and include the processes that provide these services into a custom-built operating system image. You create this image with the **buildqnx** utility. The image can be booted from disk by the QNX partition loader or booted over the network using the **netboot** utility.

Constructing a build file

The **buildqnx** utility takes a text file containing a list of process names as input and produces a binary image file containing these processes as output. This is called a “build” file. The build files are kept in the directory **/boot/build**, and the image files are kept in the directory **/boot/images**.

You can create an image by invoking **buildqnx** directly or by using the **make** utility and the **Makefile** in the **/boot** directory. For example, you could use:

```
buildqnx build/ws images/ws
```

OR

```
make b=ws
```

Each program you want included in the created image occupies two lines in the build file. Empty lines are ignored. The first line is the pathname of the program you want included. The second line starts with a **\$** followed by a number representing the size of the heap (alloc space) to be used, followed by a command.

The following example would select the QNX Filesystem Manager (**Fsys**) and start it with its **-x** option to specify that 1/2 the cache is to be used to cache executables.

```
/bin/Fsys  
$ 0 Fsys -x 2
```

Setting the heap size

When an image is created for a process, space is reserved in the image's data segment for local memory allocation. If this space is set too small, the system will have to dynamically grow the data segment at runtime, which can result in minor memory fragmentation. If the space is set too large, you'll end up wasting memory. For more information on setting appropriate heap sizes, see the documentation on **Proc** in Part II, *Managers, Drivers, & Utilities*.

Selecting processes for an image

The processes you include in an operating system image are determined by several factors. You can typically group images into three classifications:

- images that boot from disk
- images that boot over the network
- images for embedded systems

For images that boot from disk or over the network, you can start most processes after booting. You do so by placing their command line in the system initialization file that QNX executes after boot (see the section "System initialization file" in Chapter 1). This lets you keep the boot image small and simple.

When you're building an image, remember that there are two required processes:

- the Process Manager/Microkernel (**/boot/sys/Proc**)
- the system shared library (**/boot/sys/Slib**)

If you want to use the QNX network, you'll also need to include the Network Manager (**/bin/Arcnet**)

Disk images

For hard disk booting, you need to include:

- the two required system processes (**Proc** and **Slib**)
- the Filesystem Manager (**Fsys**)
- the driver required to access the drive (**Fsys.driver**)
- the Network Manager (**Arcnet**) if you want to use the network

The Makefile

The **Makefile** under the directory **/boot** contains an entry for making a generic hard disk boot. You can provide command-line macros to change the defaults. The following macros are defined:

Macro	make argument	Default, if not specified	Description
\$b	b=buildfile	<i>no default</i>	build file
\$d	d=driver	Fsys.wd	disk driver
\$h	h=hard disk	/dev/hd0	hard disk file
\$i	i=imagefile	images/buildfile	image file
\$p	p=parameters	<i>none</i>	hard disk parameters

This is the makefile:

```
FILES = @build/$b

# Define some defaults which may be overridden
# by h=xx and d=xx

d      = Fsys.wd
h      = /dev/hd0
i      = $b

$b:    images/$b

# Each OS image file is dependent on the core
# modules it is made from.

images/$b: build/$b $(FILES)
    buildqnx -v d=$d h=$h p=$p build/$b images/$i
```

The following examples will help clarify the **Makefile** usage:

Use build file **/boot/build/hard** and make an image **/boot/images/hard** that contains a Western Digital driver and a **mount** command that mounts a QNX filesystem on **/dev/hd0t77**:

```
make b=hard
```

Use build file **/boot/build/hard** and make an image **/boot/images/hard** that contains a PS/2 SCSI driver instead of the Western Digital driver and a **mount** command that mounts a QNX filesystem on SCSI ID 6:

```
make b=hard d=Fsys.ps2s h=/dev/hd6
```

Use build file **/boot/images/hard** and make an image **/boot/images/ps2s** that contains a PS/2 SCSI driver instead of the Western Digital driver and a **mount** command that mounts a QNX filesystem on SCSI ID 6. The image is written to the image file **ps2s** rather than the default, which is the name of the **b=** option (in this case **hard**).

```
make b=hard d=Fsys.ps2s h=/dev/hd6 i=ps2s
```

Copying an image to **./boot**

Once you've built an image, it won't become the new boot image until you copy it to the **./boot** file. However, before you do this, you should make a backup of the current **./boot** file by copying it to the **./altboot** file:

```
cp ./boot ./altboot
```

If for any reason your new image doesn't work properly, you can press <Esc> when prompted during the boot process and load the **./altboot** file instead of the **./boot** file. When you select the alternate boot image, the normal check for the **/etc/config/sysinit.node** file is replaced by a check for the **/etc/config/altsysinit** file. You should ensure that the **altsysinit** file contains the latest copy of your working **sysinit** file:

```
cp /etc/config/sysinit.node /etc/config/altsysinit
```

The **hard** build file doesn't contain the Device Manager **/bin/Dev**. The Device Manager and its drivers are usually started in the system initialization file *after* the system boots.



If the system initialization file isn't executed, the Device Manager will *not* be started. As a result, your keyboard and system console won't function.

The build file for hard disks contains an entry for the network manager (**/bin/Arcnet**). This entry is mandatory if you are running QNX on a network. **Arcnet** will automatically terminate if no network hardware is present.

A network link can be used to perform remote diagnostics. For example, assume that node 1 is running with access to an available filesystem and node 8 was just booted but appears dead (the keyboard is ignored and there's no **login** prompt). Entering the following commands from node 1 might provide some clues as to the state of node 8:

```
sin -n 8
sin -n 8 info
prefix -n 8
```

Network images

You normally use a network image to boot a diskless workstation over the network. As such, the boot image doesn't need to contain a local filesystem or a disk driver. The **Makefile** under **/boot** contains an entry for making a generic workstation boot.

The following example uses the build file **/boot/build/ws** and makes an image **/boot/images/ws**:

```
make b=ws
```

The **buildqnx** utility defines a macro, namely **\$(node)**, which is the current node number. When you boot over the network, you have the option of loading a pre-built image or building one on the fly. If you build an image on the fly—which is recommended—then you won't need to build one manually as shown above. This option is specified in the **/etc/boot/netboot** file and documented with the **netboot** utility.



When the **netboot** utility invokes **buildqnx** to build an image on the fly, the image file is *not* written to disk.

**Embedded
images**

An embedded system doesn't need to contain a filesystem or a network manager. As such, it might not be able to load a new process after booting. If you wanted to load a new process, you'd have to build an image containing *all* the processes needed to run the system. This image would then have to be transferred to the embedded system (most likely in ROM).



Chapter 8

Disk & File Recovery

In this Chapter...

Introduction

Overview of QNX 4 disk structure

File maintenance utilities

Disk recovery procedures

What to do if your system will no longer boot

Recovering lost files and directories

Introduction

The QNX filesystem achieves high throughput without sacrificing reliability. The filesystem is designed to be as robust as possible; however, there will always be situations in the real world where disk corruption will occur. Hardware will fail eventually, power will be interrupted, and users will no doubt reboot the computer at just the wrong time.

The QNX filesystem has been designed to tolerate such catastrophies. It is based on the principal that the integrity of the filesystem as a whole should be consistent at all times—any single failure will damage no more than the contents of files opened for writing at the time of a crash.

Effectively, this means that the most common types of failures—power failure, reboot, CPU crash—usually leave the QNX filesystem intact.

If a crash occurs, we've provided recovery utilities that let you examine the filesystem to determine if any damage was done to files there were open for writing at the time of the crash. These same utilities can also fix such damage, and in many cases will completely restore the filesystem.

Sometimes the damage may be more severe. For example, it's possible that a hard disk will develop a bad block in the middle of a file, or worse, in the middle of a directory or some other critical block. Again, the utilities we've provided can help you determine the extent of such damage. You can often rebuild the filesystem in such a way as to avoid the damaged areas. In this case, some data will be lost, but with some effort, a large portion of the affected data may be recovered.

Overview of QNX 4 disk structure

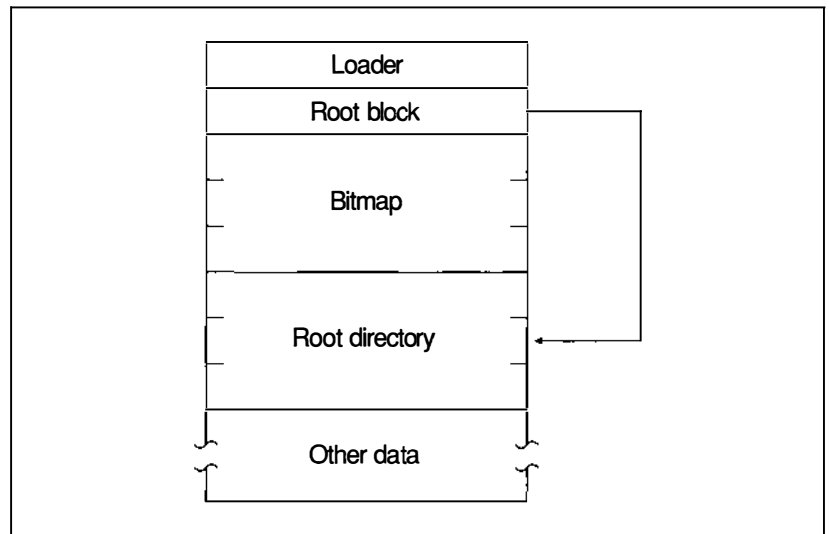
In this section, we describe how the QNX filesystem stores data on a disk. Reading this section should enable you to recognize and possibly correct filesystem damage if you ever have to rebuild a filesystem.

If you have a C development package, the header file `<sys/fsys.h>` contains the definitions for all terms used in this section.

For an overall description of the QNX filesystem, see the *User's Guide*, Chapter 5, The Filesystem Manager.

Partition components

A QNX filesystem may be an entire disk (in the case of floppies) or it may be one of many partitions on a hard disk. Within a disk partition, a QNX filesystem contains the following components:



The following blocks are always found, in order, on a QNX 4 disk partition:

- 1) loader block
- 2) root block
- 3) bitmap blocks
- 4) root directory

Loader block

The first block of a QNX partition, the loader block contains the bootstrap loader that loads the QNX OS into memory.

Root block

The second block of a QNX partition, the root block contains the directory entry for the root (`/`), the inode entries for the inode file, and a label field.

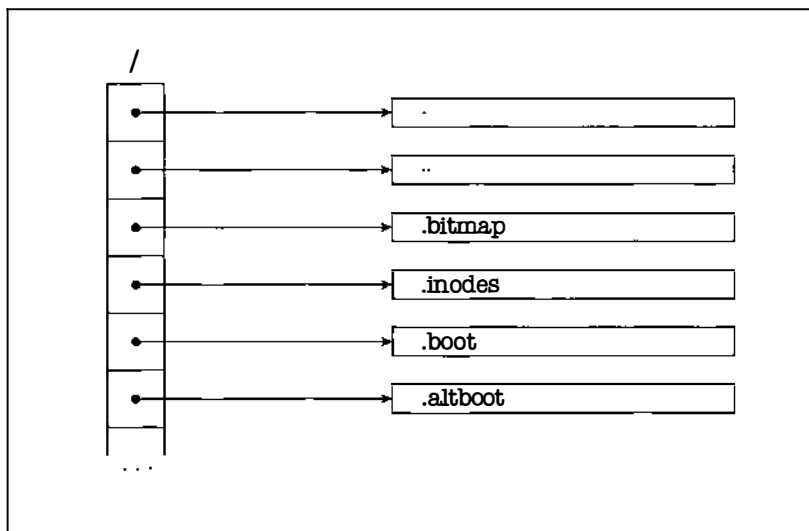
Bitmap blocks Several consecutive blocks follow the root block. The bitmap blocks form the bitmap for the QNX partition. One bit exists for each block on the partition, thus one bitmap block will be used for every 4096 disk blocks (corresponding to 2M of disk space).

If the value of a bit is zero, its corresponding block is unused. Unused bits at the end of the last bitmap block (for which there are no corresponding disk blocks) are turned on.

Bit assignments start with the least significant bit of byte 0 of the first bitmap block—which corresponds to QNX block #1.

Root directory The root directory follows the bitmap blocks. The root directory is a “normal” directory (see “Directories,” below). It is initially created by the **dinit** utility with enough room for 32 directory entries (4 blocks).

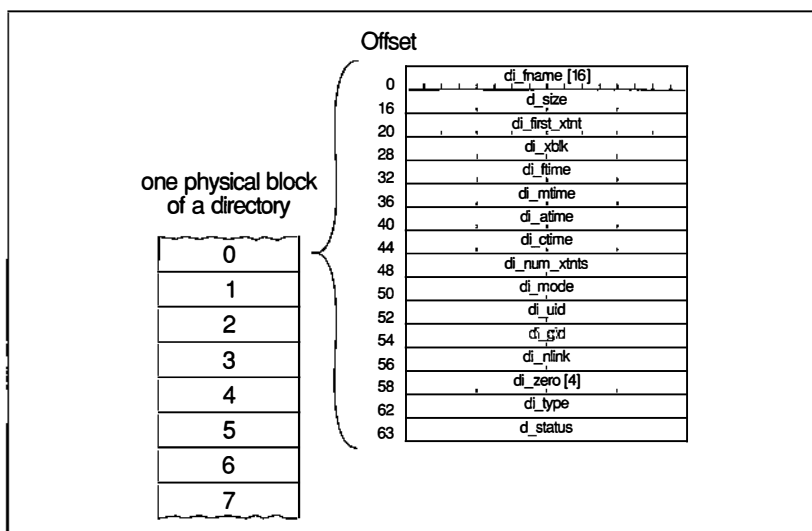
As the following illustration shows, the root directory (/) contains directory entries for several special files that always exist in a QNX filesystem. The **dinit** utility creates these files when the filesystem is first initialized.



- /.** A link to the directory: **/**
- /..** Also a link to the directory: **/**
- /.bitmap** Represents a read-only file consisting of the bitmap blocks.
- /.inodes** A normal file of at least one block on a floppy/RAM disk and 16 blocks on other disks, **/.inodes** is a collection of inode entries. The first entry is reserved and used as a signature/info area. The first bytes of the **.inode** file are "IamTHE.inodeFILE".
- /.boot** Represents an OS image file that will be loaded into memory during the standard boot process. This file will be of zero-length if no boot file exists.
- /.altboot** Represents an OS image file that will be loaded into memory during the alternate boot process. This file will be of zero-length if no alternate boot file exists.

Directories

A directory is simply a file that has special meaning to the filesystem. A directory file contains a collection of directory entries as shown in the following illustration:



Normal directory entry.

The type of directory entry is determined by the bits in the **d_status** field, as follows:

Bit 3 (FILE_LINK)	Bit 0 (FILE_USED)	Comment
Ø	Ø	unused directory entry
Ø	1	normal, used directory entry
1	x	link to an entry in / .inodes (which may be used or unused)

The first directory entry is always for the file “.” and includes a directory signature (“I♥QNX”). This entry refers to the directory itself by pointing to the entry within the parent directory that describes this directory.

The second entry is always for the “.” file. This entry refers to the parent directory by pointing to the first block of the parent directory.

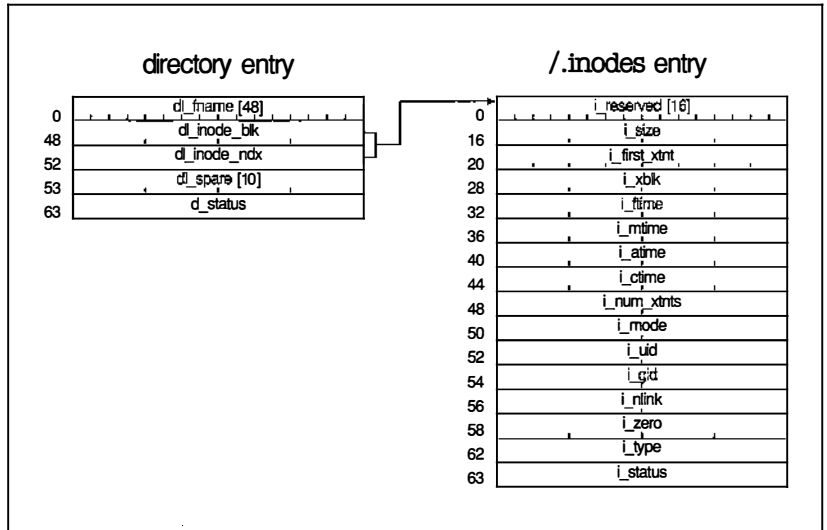
Every directory entry either defines a file or points to an entry within the / **.inodes** file. Inode entries are used when the filename exceeds 16 characters or when two or more names are linked to a single file.

The first extent (if any) of a file is described in the directory/inode entry. Additional file extents require a linked list of extent blocks whose header is also in the directory/inode entry. Each extent block in the chain points to between 1 and 60 extents.

Links

Files with names greater than 16 characters long and *links* to other files are implemented with a special form of directory entry. These entries are identified with the **_FILE_LINK** bit (0x08) of the **d_status** field being set.

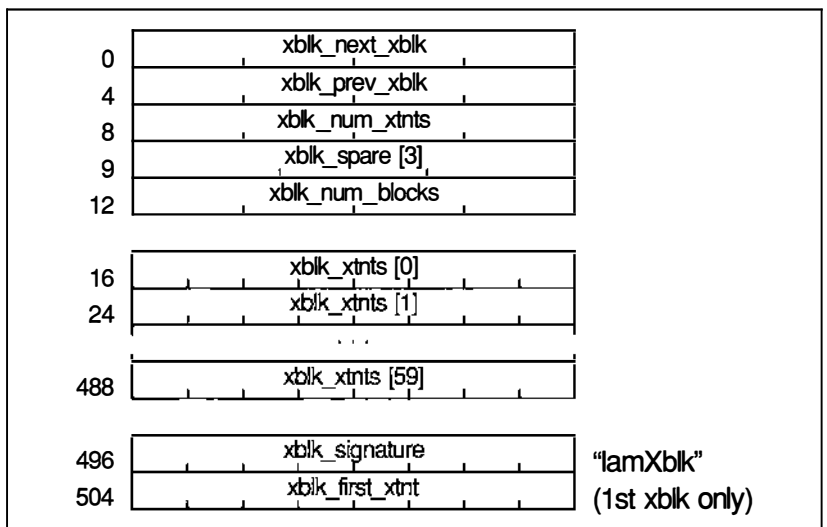
For these files, a portion of the directory entry is moved into the / **.inodes** file.



Extent blocks

Extent blocks are used for any file that has more than a single extent. The directory entry **di_xblk** points to one of these extent blocks, which in turn defines where the second and subsequent extents are to be found.

An extent block is exactly one 512-byte disk block with the following form:



Each extent block contains:

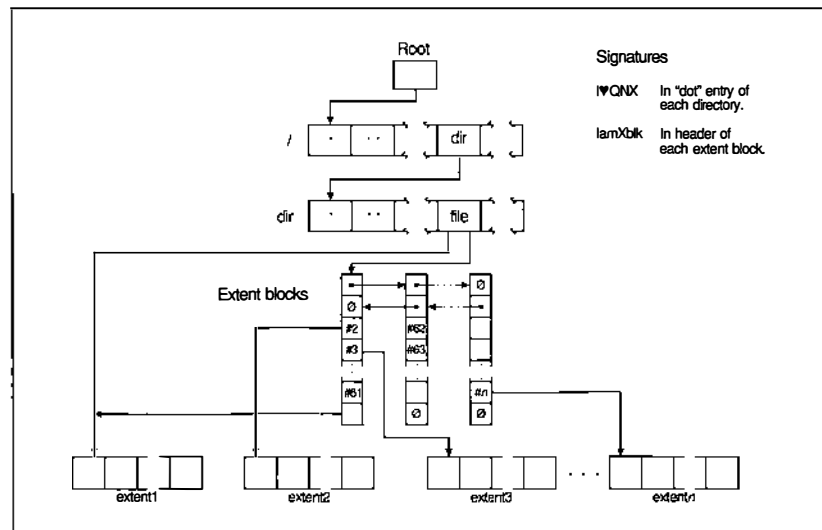
- forward/backward pointers
- a count of extents
- a count of all the blocks in all the extents defined by this extent block
- pointers and block counts for each extent
- a signature (“IamXblk”)

The first extent block also contains a redundant pointer to the first file extent (also described within the directory/node entry). This allows you to recover all data in the file by locating this block alone.

Files

Files or file extents are groupings of blocks described by directory/inode entries; they have no structure imposed on them by the QNX filesystem.

Most files in QNX 4 have the following overall structure:



File maintenance utilities

The following utilities let you create and maintain the disk/file structure:

- **fdisk**
- **dinit**
- **chkfsys**
- **zap**
- **spatch**

fdisk

The **fdisk** utility creates and maintains the *partition block* on a hard disk. This block is compatible with other operating systems and may be maintained by other OS versions of **fdisk** (although ours has the advantage of recognizing QNX-specific information). If the partition loader is missing or damaged, **fdisk** can create it.



We recommend you keep a hard copy of the partition table information for every disk in your network.

dinit

The **dinit** utility creates (but **Fsys** maintains) the following:

- boot block
- root block
- bitmap blocks
- root directory
- inode file (**/ .inodes**)

chkfsys

The **chkfsys** utility is your principal filesystem maintenance tool. This utility:

- checks the directory structure of an entire disk partition, reports any inconsistencies, and fixes them, if possible
- verifies overall disk block allocation
- writes a new `.bitmap`, upon your approval

The **chkfsys** utility assumes that the root block is valid. If the root block isn't valid, **chkfsys** will complain and give up—you'll need to try restoring the root block with the **dinit** utility.

zap The **zap** utility lets you remove files or directories from the filesystem without returning the used blocks to the free list. You might do this for several reasons, including:

- the directory entry is damaged
- two files occupy the same space on the disk (an error)

To remove zapped file entries and recover the used disk blocks, you use the **chkfsys** utility.

spatch The **spatch** utility lets you browse the raw disk and patch minor problems. You can sometimes cure transient disk problems by reading and writing the failing block with **spatch**.

Disk recovery procedures

Using **chkfsys**

The **chkfsys** utility is your principal tool for checking and restoring a potentially damaged filesystem. It can identify and correct a host of minor problems as well as verify the integrity of the entire disk system as a whole.

Normally, **chkfsys** requires that the filesystem be idle and that no files are currently open on that device. You'll have to shut down any processes that have opened files or that may need to open files while **chkfsys** is running.

To run **chkfsys** on an idle filesystem called `/dev/hd0t77`, you would simply type:

```
chkfsys /dev/hd0t77
```

The utility will scan the entire disk partition from the root down, building an internal copy of the bitmap and verifying the consistency of all files and directories it finds in the process. When it has finished processing all files, **chkfsys** will compare the internal bitmap to the bitmap on the disk. If they match, **chkfsys** is finished. If any discrepancies are found, **chkfsys** will—upon your approval—rewrite the bitmap with data consistent with the files it was able to find and verify.

In addition to verifying block allocation (bitmap), **chkfsys** attempts to fix any problems it finds during the scan. For example, **chkfsys** can:

- “unbusy” files that were written during a crash
- fix the file size in a directory entry to match the real data

When to run **chkfsys**

It’s a good idea to run **chkfsys** as part of your regularly scheduled maintenance procedures—this lets you verify that the data on your disk is intact.

It’s especially important to run **chkfsys** after a system crash so that you can identify those files, if any, that have been damaged.

If you “zap” any files for any reason—possibly because **chkfsys** suggested you do so—you’ll need to run **chkfsys** at some point to recover the *lost* blocks on the disk.

Using **chkfsys** in a live system

If you’re running an application that can’t afford downtime, you might not be able to shut down your system long enough to run **chkfsys**. In order to provide a *comprehensive* verification of the disk, **chkfsys** normally requires exclusive use of the filesystem. But you can identify potential problems in a live system by using a special, read-only mode of **chkfsys**:

```
chkfsys -f /dev/hd0t77
```

The **-fix** mode won’t be able to correct problems, but it will give you a feeling for the overall sanity of your filesystem.

If serious problems are detected that must be fixed right away, you can use another special mode—the “Rambo” mode:

```
chkfsys -R /dev/hd0t77
```

Remember that you must *not* use “Rambo” mode unless you are sure that:

- the affected files are *not* presently opened
- the affected files will *not* be opened during the rebuild process
- any files currently opened will *not* grow during the process

Using Rambo mode will fix most errors, but since you’re fixing the disk underneath a live filesystem with opened files, it will *not*:

- attempt to recover bitmap blocks
- check for block allocation errors
- attempt to rebuild the `/ .inodes` file



There is some risk to running **chkfsys** on a live system—both **chkfsys** and the filesystem are reading and possibly writing the same blocks on the disk. Also, the filesystem has internal cached data about files and directories that cannot be updated when **chkfsys** makes a change. Static changes, in-place, on files or directories that **Fsys** doesn’t currently have opened will *probably* not cause problems.

Recovering deleted files

If a file is accidentally deleted, it’s sometimes possible to recover the file using the **zap** utility. To recover a file using **zap**:

- the directory entry for that (now deleted) file must *not* be reused
- the disk blocks previously used by the file must *not* be reassigned to another file

Still, if you act quickly enough, or if you’re lucky enough, there’s a chance that the following command might restore the file:

```
zap -u filename
```

Recovering from a bad block in the middle of a file

Hard disks occasionally develop bad blocks as they age. In some cases, you might be able to recover most or even all the data in a file containing a bad block.

Some bad blocks are the result of power failures or of weak media on the hard disk. In these cases, sometimes simply reading then rewriting a block will “restore” the block for a short period of time. This may allow you to copy the entire file somewhere else before the block goes bad again. This procedure certainly can’t hurt, and is often worth a try.

To examine the blocks within a file, you use the **spatch** utility. When you get to a bad block, **spatch** should report an error, but it may have actually read a portion of “good” bytes from that block. Writing that same block back will often succeed. At the same time, **spatch** will rewrite a correct CRC that will make the block good again (but with possibly incorrect data).

You can then copy the entire file somewhere else, and then **zap** the previously damaged file. To complete the procedure, you mark the marginal block as bad (by adding it to the `/.bad_blks` file), then run **chkfsys** to recover the remaining good blocks.

If this procedure fails, you can use the **spatch** utility to copy as much of the file as possible to another file, and then **zap** the bad file and run **chkfsys**.

What to do if your system will no longer boot

If a previously working QNX system suddenly stops working and will no longer boot, then either the hardware has failed or the data on the hard disk has been damaged.

The following steps can help you identify the problem. Where possible, corrective actions are suggested.

Step 1 Try booting from floppy or across the network

If you don't have a network...

If you don't have access to a QNX network, then you'll need to boot from the QNX boot floppy that was used to install your system onto the hard disk. The filesystem will already be running in this case.

If you do have a network...

If you have a network to boot over, try booting your machine over the network. Once the machine is booted, you'll need to start up a local filesystem:

```
Fsys &
```

Step 2 Start the hard disk driver

You now have to start the appropriate hard disk driver. For example, to start a disk connected to a standard Western Digital compatible controller, you would type:

```
Fsys.wd
```

This should create a block special file called `/dev/hd0` that represents the entire hard disk.

Step 3 Run `fdisk`

Running the `fdisk` utility will immediately give you useful information about the state of your hard disk.

The `fdisk` utility might report one of several types of problems:

Error reading block 1

Probable cause—If `fdisk` reports an error trying to read the partition block (which is the first physical block on the disk), then chances are that either the disk controller has failed or the hard disk has failed.

Remedy—If the disk is good, replacing the controller card *might* allow you to continue using the disk. Otherwise, you have little option but to replace the hard drive, reinstall QNX, and restore your files from backup.

Wrong disk parameters

Probable cause—If **fdisk** successfully reads block 1, but reports incorrect disk parameters (#heads, #cylinders, #sectors/cylinder), then your hardware has probably “lost” its information about this hard drive. The most probable cause is that the battery for the CMOS memory is running low.

Remedy—Rerunning the hardware *setup* procedure (or the programmable option select procedure on a PS/2) will normally clear this up. Of course, replacing the battery will make this a more permanent fix.

Bad partition information

Probable cause—If the disk size is reported correctly by **fdisk**, but the partition information is wrong, then the data in block 1 of the physical disk has somehow been damaged.

Remedy—Use **fdisk** to recreate the correct partition information. It’s a good idea to keep a hard copy of the correct partition information available, in case you ever have to do this step.

Step 4 Create a raw volume

At this point, you have verified that the hardware is working (at least for block 1) and that a valid partition is defined for QNX. You now need to create a raw volume for the QNX partition itself:

```
mount -p /dev/hd0
```

This should create a volume called **/dev/hd0t77**, or its equivalent. If the partition information is correct, this shouldn’t fail.

Step 5 Mount a QNX filesystem

At this point, you still don’t know the health of the filesystem on the QNX partition. Your goal now is to run the **chkfsys** utility on the disk to determine—and possibly fix—the filesystem. But first you’ll

need to create a mount point for a QNX filesystem. Since the root (/) already exists (on a floppy or on a remote disk on the network), you can mount the local hard disk partition with the name **/hd**:

```
mount /dev/hd0t77 /hd -r
```

Depending on the state of the QNX partition, the **mount** may or may not fail.

If the **mount** fails

If the **mount** fails, the first portion of the QNX partition is probably damaged (since **Fsys** will refuse to mount what it considers to be a corrupted filesystem).

In this case, you can use the **dinit** utility to overlay sufficient good information onto the disk to satisfy **Fsys**:

```
dinit -hr /dev/hd0t77
```

The **-r** option tells **dinit** to rewrite:

- the root block
- the bitmap (with *all* blocks allocated)
- the constant portions of the root directory.

After doing this, you will need to rebuild the bitmap with **chkfsys**, even on a good partition.

You should now be able to reissue the **mount** command and once again try to create a mount point for a QNX filesystem called **/hd**.

Step 6 Run *chkfsys*

At least a portion of your QNX filesystem should now be accessible. You can use **chkfsys** to examine the filesystem and recover as much data as possible.

If you booted:	you enter this command:
from floppy	<code>/hd/bin/chkfsys /dev/hd0t77</code>
across the network	<code>chkfsys /dev/hd0t77</code>

If the disk is unrecoverable

In either case, you should make note of any problems reported, and allow **chkfsys** to fix as much as it can. What you do next depends on the result of running **chkfsys**.

If, for any reason, your disk is completely unrecoverable, read the next section “Recovering lost files and directories.” In some cases, you may need to reinstall QNX from floppy, and restore your disk from your backup files.

If significant portions of the filesystem are irreparably damaged, or important files are lost, then restoring from backup might be your best alternative.

If the filesystem is intact

If your filesystem is intact, yet the machine still refuses to boot from hard disk, then either of the following is probably damaged:

- the partition loader program in physical block 1
- the QNX loader in the first block of the QNX partition

To rewrite a partition loader, use **fdisk**:

```
fdisk /dev/hd0 loader
```

To rewrite the QNX loader, use **dinit**:

```
dinit -b /dev/hd0t77
```

You should now be able to boot your system.

Recovering lost files and directories

You may sometimes find that files or directories have been completely lost due to disk corruption. If after running **chkfsys** you know that certain key files or directories were *not* recovered, then you might be able to use **spatch** to recover some or all of this data.

Before attempting this, you should first familiarize yourself with the details of a QNX filesystem, as described in the section “Overview of QNX 4 disk structure” in this chapter. You should also study the documentation for the **spatch** utility (see the *Utilities Reference*).

The **lost+found** directory

The **spatch** utility can take a range of disk blocks and *link* them into the filesystem as a real file (or directory) under the **lost+found** directory.

A space for the **lost+found** directory always exists and has room for at least 126 entries on a hard disk.

After you use **spatch** to link all such pieces of the disk into the **lost+found** directory, you should immediately run **chkfsys** to rebuild the bitmap and verify the integrity of the filesystem that now includes these new files.

Once **chkfsys** validates the disk, you can safely move these files (or directory trees) to other directories on the disk.

It's a good idea to keep the **lost+found** directory empty so that there will be as much room as possible to recover files in any future disk recovery episodes.

Using **spatch**

The **spatch** utility has a special menu item, called **Mark**, that allows you to identify a range of disk blocks.

The first time you select **Mark**, the **spatch** utility will mark the current disk block as the first block in a range of blocks. You can then move to the last block of the region, and select **Mark** a second time. This will identify the region of blocks to **spatch**. A special menu will now appear giving you the option of:

- removing all marks
- linking the marked blocks into **lost+found** as a file
- linking the marked blocks into **lost+found** as a directory
- copying the marked blocks to a file
- copying with append the marked blocks to a file
- ignoring the second mark, allowing you to select a different one

If you choose to link or copy the blocks into a file/directory, **spatch** will ask if you first want to perform a scan of the entire disk looking for an *xblk* that references this range of blocks. If you say yes, then

spatch will perform the scan, and if it finds a block, it will give you the option of linking in all the extents defined by that *xblk* and any other associated *xblks*.

The *xblk* scanning allows potentially large disk files (or directories) to be recovered in one piece.

If you choose not to scan for *xblks*, or no such *xblk* is found, then a file/directory will be created that has only a single extent containing just these blocks.

If you copy, you should specify a file on another disk (filesystem).

Remember to always run **chkfsys** after linking or copying a number of files or directories.



Chapter 9

The Quantum Update System

In this Chapter...

Introduction

Reporting bugs

Free software

Electronic mail

Conferencing system

About our hardware

How to phone us

Logging in

Using QUICS

Introduction

The Quantum update system, called QUICS, runs 24 hours per day on a computer that you may call via modem to:

- download fixes and changes to the operating system, utilities, and libraries
- download a variety of free software from various sources
- send electronic mail to Quantum
- engage in a conference with other QNX users on a large variety of topics

Reporting bugs

Quantum's software refresh policy provides prompt response. For example, if you report a bug, perhaps in a utility or library routine, you can often download a revised version within hours of reporting the problem. We maintain the most current versions of all programs, including operating system modules, for download. Along with these, we provide a list of all changes made to the system. You may want to sign on once a month and make a log of this list.

Free software

We also maintain a growing list of free software that may be of interest to you. It includes useful utilities, games, and megabytes of public domain source provided by ourselves and by our customers. This source can often help you understand some of the intricacies of QNX.

Electronic mail

You can use the private mail facility to communicate privately with Quantum employees or with any other user of QUICS.

Conferencing system

If you'd like to make a comment or ask a question that you feel other users may be interested in, then you'll want to use our conferencing system. Its command structure is very similar to that used by *Byte* magazine's conferencing system.

All Quantum developers are members of the conferencing system, as are most of our more active third-party developers. It's not unusual to pose a question and have several comments on it within an hour or so. You'll find topics on realtime, hardware, utilities, C, and so on.

You don't need to be a QNX customer to use the conferencing system. You can access it from a terminal or a terminal emulation program running under DOS.

About our hardware

At the time of this printing, the QUICS computer supports seven modem lines and eight X.25 channels into the Datapac public packet switching network.

All modems will support a variety of baud rates and data compression protocols. The following table shows the hardware configuration at the time of printing:

Hunt group	Baud rates	Modem type	Protocols
613-591-0934	19200 to 300	HST-9600	MNP5,ARQ
613-591-0935	38400 to 300	HST-14.4K	MNP5,ARQ,v42bis
613-591-0940	38400 to 300	HST-Dual-Std	MNP5,ARQ,v42bis,v32
613-591-1735	38400 to 300	HST-Dual-Std	MNP5,ARQ,v42bis,v32
613-591-3009	38400 to 300	HST-Dual-Std	MNP5,ARQ,v42bis,v32,v32bis
613-591-3128	19200 to 300	Telebit	MNP5,PEP
613-591-6093	19200 to 300	Telebit	MNP5,PEP

The modems are on a hunt group with the following primary number:

(613) 591-0934 First number of the hunt group.

On the update system, we maintain an up-to-date configuration on the number of modems, their types, and phone numbers. You can read this when you give us a call.

To connect to QUICS via X.25, you have to make a pre-paid call or register with us for an X.25 account number (allowing you to make collect calls). Our X.25 network address is:

3020 85701416

Quantum's Datapac address

Datapac's network code

How to phone us

Since **qtalk** is shipped with every system, we describe how to contact us from within **qtalk**. We assume that your modem is connected to the QNX device `/dev/ser1`. If it isn't, you'll have to tell **qtalk** where your modem is.

Step 1 Set your serial port settings

Set your serial port settings to the following values. Refer to the section on configuring modems if you need help with this.

```
par=none
bits=8
```

You'll also need to set your baud rate to whatever rate your modem will support.

Step 2 Start **qtalk** and have your modem dial us

If you are in North America and have a Hayes-compatible modem, you can use **qtalk**'s dialing directory capability. The default dialing directory, `/etc/config/dial.dir`, provides an entry called

quics, which contains Quantum's phone number. For more information, see the documentation on **qtalk** in the *Utilities Reference*.

```
qtalk quics [-m modem] (Hayes-compatible modem)
```

OR

```
qtalk [-m modem] (other modem)
```

The **-m modem** option is needed only if your modem is *not* connected to the **/dev/ser1** device.

Logging in

Once you have successfully established communications with QUICS, you'll be asked to log in. If this is your first time signing onto QUICS, you should log in as user ID **newuser**.

Login: **newuser**

Next you'll be asked to enter information about you and your company as well as the name you wish to create as your own personal user ID.

Since there are many thousands of QUICS users, we ask that you avoid simple names (like your first name alone) when selecting your user ID. A good rule that works well is to select a name consisting of your first two initials, followed by your last name.

For example, the user ID for "Johann Sebastian Bach" would be:

jsbach

From now on, you log in with your user ID (instead of **newuser**) each time you access QUICS.

Using QUICS

After logging in, you'll see the main menu. QUICS is a fully menu-driven system. We encourage you to explore the system to get familiar with its many capabilities.

By the way, one of the submenus lets you download a manual on how to use QUICS.

If at any time you experience difficulty, just give our technical support line a call:

(613) 591 0941

We at Quantum look forward to seeing you on QUICS!

Managers, Drivers, & Utilities

Arcnet	Arcnet network manager (QNX) 113
buildqnx	build a custom version of QNX (QNX) 115
cpio	copy file archives in and out (UNIX) 121
Dev	device manager (QNX) 125
Dev.con	console driver (QNX) 127
Dev.par	parallel device driver (QNX) 129
Dev.ser	serial device driver (QNX) 131
Dosfsys	DOS filesystem manager (QNX) 133
emu87	80x87 floating point emulator (QNX) 137
fcats	display compressed files (QNX) 139
fdisk	create QNX disk partition (QNX) 141
freeze	compress and uncompress files (QNX) 145
Fsys	Filesystem Manager (QNX) 149
Fsys.aha	Adaptec SCSI host adapter (QNX) 151
Fsys.floppy	floppy disk controller (QNX) 153
Fsys.ps2e	driver for the PS/2 ESDI controller (QNX) 155
Fsys.ps2m	driver for the IBM PS/2 MFM Disk Controller (QNX) 157
Fsys.ps2s	driver for the PS/2 SCSI controller (QNX) 159

Fsys.qic02 QIC-02 tape driver (QNX) 161
Fsys.wd driver for PC/AT fixed disk interface (QNX) 163
license license software for use (QNX) 165
lpsrvr print server (QNX) 167
melt uncompress files (QNX) 169
Mouse mouse driver (QNX) 171
netboot service boot requests from the network (QNX) 173
netnode display or change the Arcnet node permit bitmap (QNX) 175
netpoll display or change default polling parameters of a node (QNX) 177
pax portable archive interchange (POSIX) 179
Proc Process Manager (QNX) 185
sinit system initialization (QNX) 189
tape control a tape driver (QNX) 191
tar read and write tape archive files (UNIX) 193
termdef prompt for terminal type (QNX) 197
vol manage multi-volume archives (QNX) 199

Buildfile syntax: **Arcnet** [-i *irq*] [-m *mem*]

Options:

- i *irq* The interrupt line to use (IRQ). This option overrides automatic IRQ detection.
- m *mem* The memory segment to use. This option overrides automatic memory detection.

Examples: Use automatic IRQ and memory detection:

Arcnet

Set the QNX Arcnet board to use IRQ 3:

Arcnet -i 3

Set the QNX Arcnet board to use IRQ 3; set its memory segment to be located at CE00 hex:

Arcnet -i 3 -m CE00

Description: The **Arcnet** network manager communicates directly with the QNX Arcnet network board. It provides the low-level services the Process Manager and Kernel require to support transparent networking.

If you want to use the network, the **Arcnet** manager must be built into the operating system image. You cannot start **Arcnet** in your **sysinit** file or at the command line.

The QNX Arcnet board uses an interrupt line (IRQ) and 8K of memory. The **Arcnet** manager should automatically determine these values, as configured on the QNX Arcnet board. The command-line options are meant for cases where **Arcnet** is unable to automatically detect the interrupt or the memory segment.



Automatic IRQ detection does not work on IRQ 2. To use IRQ 2, you must specify the **-i irq** option.

If you experience any problems with the QNX Arcnet network, remember to run the **netinfo** utility—this gives you more information about recent significant network events, and about the current state of the network board.

Arcnet

Arcnet network manager (QNX)



If for some reason **Arcnet** cannot initialize the hardware or its runtime environment, it terminates. If this happens, a message is printed directly to the current console with an integer error code. Record this code and contact Quantum for more information.

Exit status: 0 Successful.
 >0 An error occurred.

See also: **netnode, netpoll**
Utilities Reference: **netinfo**

Syntax: `buildqnx [-v] [macro=name] build_file os_file`

Options:

- `-v` Be verbose.
- `macro=name` Define a macro.
- `build_file` An input file of module names.
- `os_file` An output image file.

Examples:

```
cd /boot
buildqnx build/ws images/ws.400
```

Description: You use the **buildqnx** utility to build a QNX boot image. A boot image is a collection of individual processes packed into a single file image. This image is loaded into memory at boot time. Once the image is in memory, control is transferred to the first process in the image—this process starts each of the other processes.

Each major QNX service is provided by a process. You can add or delete an operating system service by adding or deleting the process that manages it. Most processes may be started after the system boots, and as such do not need to be included in an image.

For example, let's say you've booted from disk and the boot image didn't contain the DOS filesystem manager (**Dosfsys**). At this point you're getting your files and commands from a local QNX filesystem. You could start **Dosfsys** by typing its name on the command line and running it as a background process, as follows:

```
Dosfsys &
```

It's necessary to start some processes before the operating system loads. For example, if a diskless workstation boots over the network, it will need to have a Network Manager in the image to access the remote filesystem. Likewise, if you boot from a hard disk, you can't start a filesystem by loading it off the disk, since you don't yet have a filesystem. In this case, you would build an image containing **Fsys**, **Fsys.driver**, and **mount** so that the filesystem is available immediately upon loading.

buildqnx

build a custom version of QNX (QNX)

As a general rule, you should build into the image only the minimum services needed, and start other services after boot. In some environments, you may want to freeze an OS and a larger number of services by building a large image that you don't change. If you're in a special realtime environment where dynamic loading isn't possible, you may also choose to build a complete load image of all the processes you need.

The core kernel modules and images are placed under the `/boot` directory. This directory also contains a Makefile, which references a set of build files that name the processes to be included in an image.

The Quantum kernel processes that make up the required operating system services are kept in the `/boot/sys` directory. The following lists Quantum processes that are often included in an image:

Type	Process	Description
A	Proc	Process Manager, including the microkernel
A	Slib	Shared Library
B	Fsys	QNX Filesystem Manager
B	Fsys.floppy	NEC 765 floppy driver
	Fsys.wd	Western Digital 1002-1007 hard disk driver
	Fsys.ps2m	PS/2 MFM hard disk driver
	Fsys.ps2e	PS/2 ESDI hard disk driver
	Fsys.ps2s	PS/2 SCSI hard disk driver
C	Fsys.aha	Adaptec SCSI hard disk driver
	Dosfsys	DOS Filesystem Manager (requires Fsys)

Type	Process	Description
C	Dev	Device Manager
C	Dev.con	Monochrome, EGA, VGA console text driver
	Dev.par	Parallel driver
	Dev.ser	8250, 16450, 16550 RS-232 serial driver
D	Arcnet	QNX Arcnet Network Manager/driver
E	sinit	System initialization process
F	Debugger	Low-level system debugger (not for general use)

The process types are as follows:

- A Must be present in an image as the first two processes.
- B Needed if you want to use the network.
- C Can be started later in the **sysinit** file—we recommend this method.
- D Needed if you want to use the network.
- E Usually included to execute a shell file of processes that you start dynamically.
- F Needed for low-level system work.

You must include the Process Manager and Shared Library in the boot image. If you want to use the network, then you must also include the Arcnet Network Manager. The Filesystem Manager and Device Manager typically require that you include one or more drivers with them if they are to be useful.

The list of programs to build into the image is kept in a build file under **/boot/build**. Each program occupies two lines in this file (empty lines are ignored). The first line is the pathname of the program. The second line starts with a **\$** followed by a number representing the size of the heap (alloc space), followed by a

buildqnx

build a custom version of QNX (QNX)

command line. Your program will start at priority 10, with a round-robin scheduling algorithm.

The following is a simple build file for creating an image to boot from a hard disk with a Western Digital controller on a single-node computer. The `sinit` utility is used to execute the system initialization file.

```
sys/Proc
$ 38000 Proc

sys/Slib
$ 1 Slib

/bin/Fsys
$ 16000 Fsys

/bin/Fsys.wd
$ 1000 Fsys.wd

/bin/mount
$ 4000 mount -p /dev/hd0 /dev/hd0t77 /

/bin/Arcnet
$ 1000 Arcnet

/bin/sinit
$ 2000 sinit
```

Likewise, the following build file will create an image to boot a workstation over the network. Note the macro `$(node)`, which will be replaced by the current node ID.

```
sys/Proc
$ 38000 Proc

sys/Slib
$ 1 Slib

/bin/Arcnet
$ 1000 Arcnet
```

```
/bin/sinit
$ 2000 sinit -r //$(node)/

sys/Debugger
$ 1 Debugger
```

You should start the processes that were omitted from the image in the system initialization file, which is executed by the **sinit** process. The **sinit** process will attempt to execute the **/etc/config/sysinit.node** file on the current node. If the file doesn't exist, **sinit** will attempt to execute the **/etc/config/sysinit** file.

The following is an example of a simple system initialization file:

```
export TZ=EST5DST4,M4.1.0/3,M10.5.0/3
rtc hw
Dev &
Dev.con -n 3 &
reopen /dev/con1
Dev.ser &
Fsys.floppy &
tinit -t con2 con1
```

Starting the Device Manager and console driver after boot presents a minor problem. You can't start the console driver from the keyboard, since your system doesn't yet recognize a keyboard. But you can start it from the system initialization file. You should start the Device Manager and console driver right after setting the date, as the next two commands in the file. You then execute the shell command:

```
reopen /dev/con1
```

which causes the shell to reopen its standard input and standard output.

Note the last command, **tinit**, which starts up user logins on two consoles.

Let's say the above build file is in **/boot/build/test**. You could build an image with the **make** utility, using the following:

buildqnx

build a custom version of QNX (QNX)

```
cd /boot
make b=test
```

An image, called **test**, is placed in the **/boot/images** file. To make this image bootable from a hard disk, copy it to the **/.boot** file in the root directory of the hard disk. Before copying the file, you may wish to make a copy of the existing boot image—in case you made a mistake in your build file—by copying the existing **/.boot** to **/.altboot**, as follows:

```
cp /.boot /.altboot
cp /boot/images/test /.boot
```

We have provided a Makefile and a few standard build files in the **/boot** directory that you can use as models:

```
make b=hard d=driver_name    hard disk image
make b=ws                    workstation image
```

Exit status: 0 Successful.
 >0 An error occurred.

See also: **netboot, sinit**
Utilities Reference: **tinit**

Syntax: Read/list an archive:

cpio -i[Bcdfmrtuv] [*pattern...*]

Write an archive:

cpio -o[Bacv]

Copy files:

cpio -p[adlmruv] *directory*

Options:

- B** Cause input/output to be blocked 5120 bytes to the record. Can be used only with the **-o** or **-i** options for data directed to or from character special files.
- a** Reset access times of input files after they've been copied. When the **-l** option is also specified, the access times of linked files aren't reset. Can be used only with the **-o** or **-i** options.
- c** Write header information in ASCII. Can be used only with the **-i** or **-o** options. Note you should always use this option to write portable files.
- d** Create directories as needed. Can be used only with the **-i** or **-p** options.
- f** Copy in all files except those in patterns. Can be used only with the **-i** option.
- l** (the letter "el") Whenever possible, link files rather than copy them. Can be used only with the **-p** option.
- m** Retain previous modification times. This option won't work on directories that are being copied. Can be used only with the **-i** or **-p** options.
- r** Interactively rename files. The system will ask you whether to rename pattern each invocation. Read and write permissions for the controlling terminal (**/dev/tty**) are required for this option. If you type a null line, the file is skipped. Should be used only with the **-i** or **-o** options.

cpio

copy file archives in and out (UNIX)

- t** Print a table of contents of the input. No files are created. Can be used only with the **-i** option.
- u** Copy files unconditionally. Usually an older file will not replace a new file with the same name. Can be used only with the **-i** or **-p** options.
- v** Be verbose. Print the names of the affected files. Can be used only with the **-i** option. Provides a detailed listing when used with the **-t** option.

[pattern]

Simple regular expression given in the name-generating notation of the shell.

directory

The destination directory.

Examples:

Copy out the files listed by the **ls** utility and redirect them to the file **newfile**:

```
ls | cpio -o > ../archive
```

Use the output file **archive** from the **cpio -o** utility, extract those files that match the patterns *memo/al* and *memo/b**, create the directories below the current directory, and place the files in the appropriate directories:

```
cat archive | cpio -id "memo/al" "memo/b*"
```

Take the filenames piped to **cpio** from the **find** utility and copy or link those files to another directory named **newdir**, while retaining the modification time:

```
find . -depth -print | cpio -pdlmv newdir
```

Description:

The **cpio** utility produces and reads files in the format specified by the POSIX **cpio** Archive/Interchange File Format. It operates in three modes.

The **-i** option (copy in) extracts files from the standard input, which is assumed to be the product of a previous **cpio -o**. Only files with names that match patterns are selected. Multiple patterns may be

specified. If no patterns are specified, the default for patterns is to select all files. The extracted files are conditionally created and copied into the current directory, and possibly any levels below, based on the options used. The permissions of the files will be those of the previous **cpio -o** invocation. The owner and group of the files will be that of the current user unless the user has appropriate privileges, which causes **cpio** to retain the owner and group of the files of the previous **cpio -o** invocation.

The **-o** option writes the archive to the standard output.

The **-p** option (pass) reads the standard input to obtain a list of pathnames of files that are conditionally created and copied into the destination directory based upon the options used.

If an error is detected, the cause is reported and the **cpio** utility will continue to copy other files. The utility will skip over any unrecognized files encountered in the archive.

The following restrictions apply to the **cpio** utility:

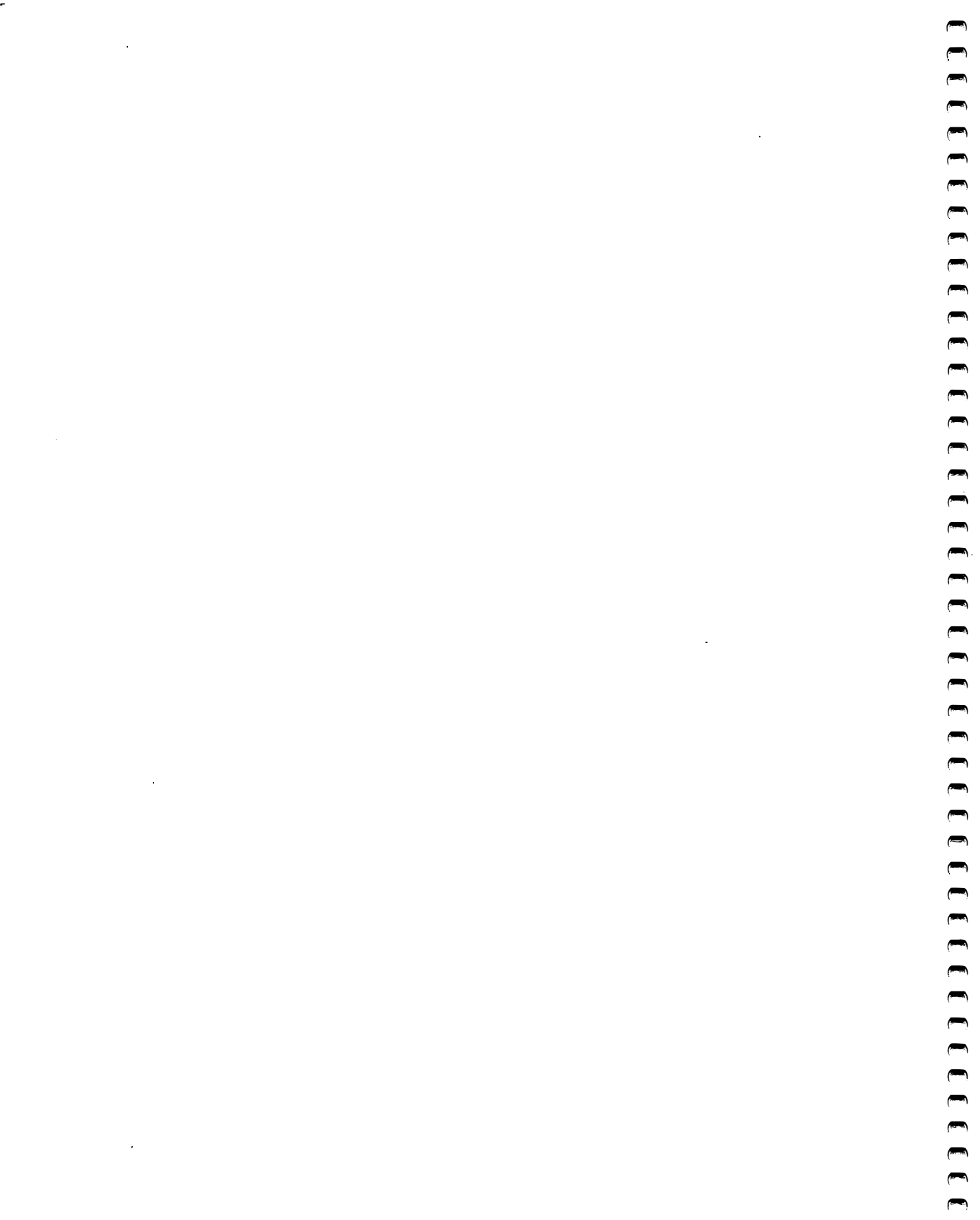
- Pathnames are restricted to 256 characters.
- Appropriate privileges are required to copy special files.
- Blocks are reported in 512-byte quantities.

Exit status:	0	All input files were copied.
	2	The utility encountered errors in copying or accessing files or directories. An error will be reported for nonexistent files or directories or for permissions that don't allow the user to access the source or target files.



It is important to use the **-depth** option of the **find** utility to generate pathnames for **cpio**. This eliminates problems **cpio** could have trying to create files under read-only directories. Note also that the controlling terminal (**/dev/tty**) is used to prompt the user for information when the **-i** or **-r** options are specified.

See also: **pax, tar**



Syntax: **Dev** [-n *max_device*] &

Options: -n *max_device* The maximum number of terminal devices to support (default is 12).

Examples: **Dev** & Start **Dev**; use defaults.
 Dev -n 100 & Allow 100 devices.

Description: The **Dev** manager is responsible for managing all *terminal devices* in a QNX system.

Once **Dev** is started, you can then start individual driver processes, which register themselves with **Dev** to support additional devices. For example:

Driver	Description
Dev.con	console I/O
Dev.par	parallel I/O
Dev.ser	serial I/O

Dev always supports the NULL device (called **/dev/null**).

See also: **Dev.con**, **Dev.ser**, **Dev.par**



Syntax: **Dev.con** [-C *number*] [-I *number*] [-N *name*]
 [-O *number*] [-n *number*] &

Options:

- C *number* The size of the canonical input buffer (default is 256).
- I *number* The size of the raw input buffer (default is 256).
- N *name* The prefix to use for the device name (default is **con**).
- O *number* The size of the output buffer (default is 2048).
- n *number* The maximum number of virtual consoles (default is 1).

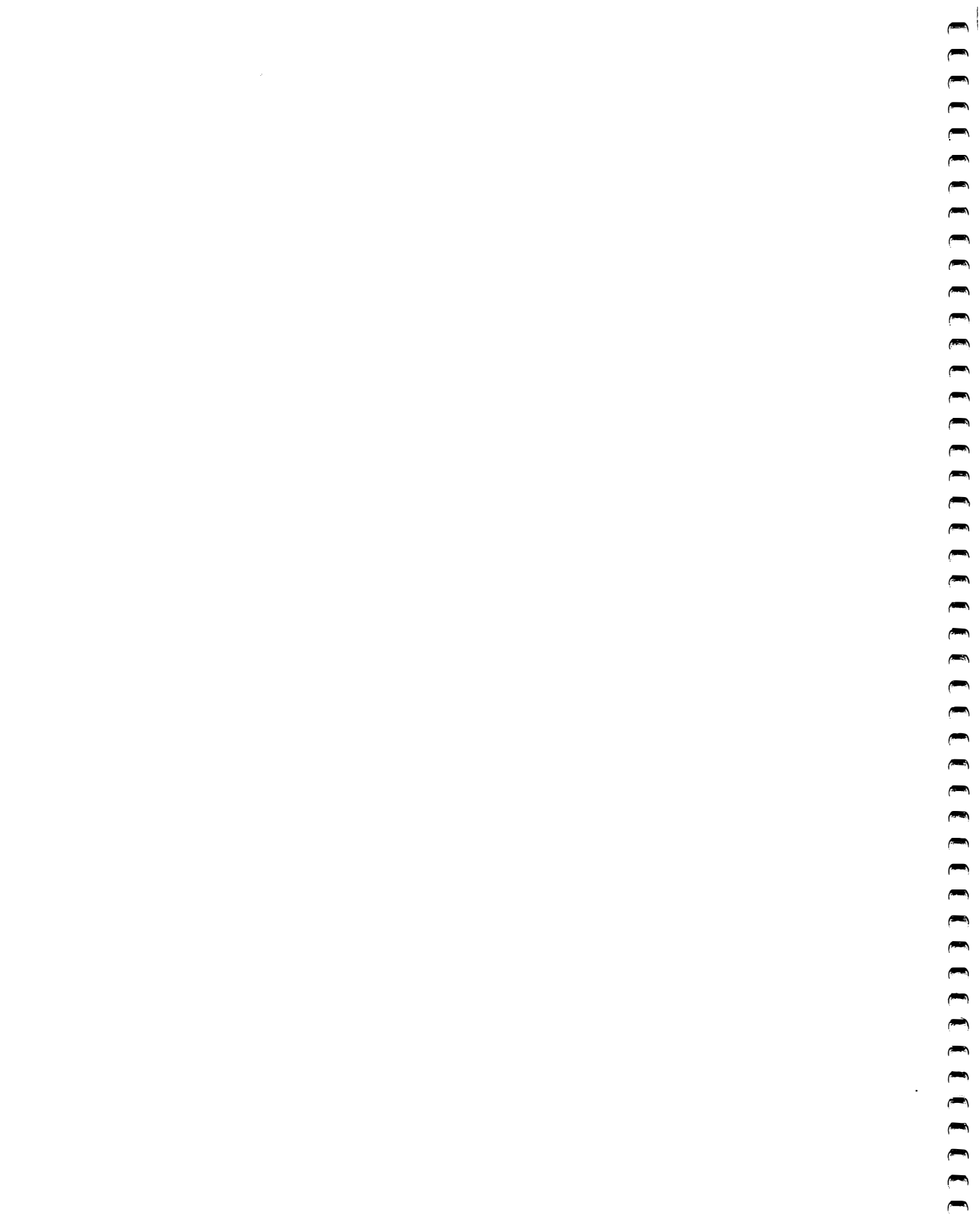
Examples:

- Dev.con** & Start **Dev.con**; use defaults.
- Dev.con** -n 4 & Allow 4 virtual consoles.

Description: The **Dev.con** driver manages all *console* devices in a QNX system.

Remember that you can start **Dev.con** only after the Device Manager (**Dev**) has been started.

See also: **Dev**, **Dev.ser**, **Dev.par**



Syntax: **Dev.par** [-b *number*] [-N *name*]
 [-p *number*] [-O *number*] &

Options:

- b *number* Use the printer number, as defined by the BIOS, where *number* can be 1, 2, 3, or 4.
- N *name* The name of the device (default is **par**).
- O *number* The size of the output buffer (default is 1000).
- p *number* The I/O address (in hex) of the parallel port (required only if you're using a non-standard parallel port that the BIOS is unable to find).

Examples: Support LPT1 as **/dev/par**:

Dev.par &

Support LPT2 as **/dev/prn**:

Dev.par -b 2 -N prn

Support port at 278 hex as **/dev/laser**

Dev.par -p 278 -N laser &

Description: The **Dev.par** driver manages all *parallel* devices in a QNX system.

You must start a copy of **Dev.par** for each parallel device **Dev.par** is to support.

If you start **Dev.par** without specifying any I/O ports, **Dev.par** uses the first parallel port found by the system BIOS (LPT1). If there aren't any parallel ports, **Dev.par** terminates.

If you need to have more than one parallel port supported, remember to assign a unique name to each of the ports.

Remember that you can start **Dev.par** only after the Device Manager (**Dev**) has been started.

See also: **Dev**, **Dev.ser**, **Dev.con**



Syntax: **Dev.ser** [*options*] &

Options: *port* [, *irq*] A serial I/O port (in hex) and its corresponding interrupt (in decimal).

-C *number* The size of the canonical input buffer (default is 256).

-D Set initial state of DSR to off.

-E Set all subsequent ports to “raw” mode (default).

-F Disable hardware flow control on all subsequent ports.

-I *number* The size of the raw input buffer (default is 2048).

-N *name* The name of the device (default is **ser**).

-O *number* The size of the raw output buffer (default is 2048).

-R Set initial state of RTS to off.

-b *number* The initial baud rate for subsequent ports (default is 9600).

-d Set initial state of DSR to on.

-e Set all subsequent ports to “edited” mode.

-f Enable hardware flow control on all subsequent ports (default).

-h *number* The flow control high-water mark for subsequent ports (default is $\frac{3}{4}$ of input buffer).

-l *number* The flow control low-water mark for subsequent ports (default is $\frac{1}{4}$ of input buffer).

-r Set initial state of RTS to on.

Examples: Scan for COM1 and COM2:

Dev.ser &

Dev.ser

serial device driver (QNX)

Set port at 3F8 as a raw device at 1200 baud, and port at 2F8 as an edited device at 9600 baud:

```
Dev.ser -E -b 1200 3f8,4 -e -b 9600 2f8,3 &
```

Specify a standard COM1 port at 3F8 as well as four ports on a multiport card:

```
Dev.ser 3f8,4 280,3 288,3 290,3 298,3 &
```

Description: The **Dev.ser** driver manages all *serial* devices in a QNX system.

If you don't specify any serial ports on the command line, **Dev.ser** scans the hardware, looking for a port at 3F8 (COM1) or a port at 2F8 (COM2), or both. If COM1 is present, **Dev.ser** assumes that it uses interrupt IRQ 4; if COM2 is present, **Dev.ser** assumes that it uses interrupt IRQ 3. If the scan shows that neither of COM1 nor COM2 is present, **Dev.ser** terminates.

Remember that you can start **Dev.ser** only after the Device Manager (**Dev**) has been started.

See also: **Dev**, **Dev.con**, **Dev.par**

Syntax: **Dosfsys** [-S|-s] [-m] [*dos_drive=qn_x_drive* [,R]]... &

Dosfsys -i [-n *node*] [*dos_drive_path*]...

Dosfsys -x [-n *node*]

Options: [,R] Mounts the DOS drive as a read-only drive.

-S Same as -s, except the drives are adopted as read-only.

-i Gets drive information from the **Dosfsys** manager on the specified node. If no node number is specified, the local **Dosfsys** manager is used.

-m Suspends adoption of the indicated drives; only displays the drives that would be adopted.

-n *node*

The node the **Dosfsys** manager resides on.

-s Causes **Dosfsys** to scan the /dev directory for drives to adopt.

-x Causes the **Dosfsys** manager on the specified node to exit. If no node is specified, the local **Dosfsys** manager is used.

dos_drive

A DOS drive letter, ranging from **a** to **h**.

dos_drive_path

The full name of the adopted DOS drive (defaults: /dos/a for **a** drive, /dos/b for **b** drive, etc.)

qn_x_drive

The name of the QNX drive or file to open (e.g. /dev/hd0t4). The drive or file can be local or it can reside on another node in the network.

Examples: 1) Scan the /dev directory and mount the corresponding drives:

Dosfsys & OR **Dosfsys** -s &

This syntax is the norm for most environments.

Dosfsys

DOS filesystem manager (QNX)

- 2) Adopt **drivea** as **/dos/a** (read-only) and
//8/disks/drivec as **/dos/c**:

```
Dosfsys a=drivea,R c="//8/disks/drivec &
```

- 3) Adopt **/dev/fd0** as **/dos/a** and **/dev/wd0t4** as **/dos/c**:

```
Dosfsys a=/dev/fd0 c=/dev/wd0t4 &
```

Description: The **Dosfsys** filesystem manager provides transparent access to DOS files and directories that reside on a DOS disk. **Dosfsys** can support up to eight drives (**a** to **h**).

You can create files on DOS disks, read them, write to them, and delete them with most QNX programs. You can also create, read, and delete directories on DOS disks with standard QNX utilities, such as **mkdir**, **ls**, and **rmdir**. Most standard QNX utilities will work with DOS files, provided that the DOS file structure allows for the functionality required by the utility.

Your own programs will also be able to process DOS files just as they process QNX files, by using the standard I/O functions such as *open()*, *read()*, *write()*, *close()*, *seek()*, etc. When you read DOS directories, they'll be presented to you in QNX format.

The **-i** option lets you get information about the currently adopted DOS drives.

The **-x** option terminates the **Dosfsys** server. When this command is issued, no new *open()* requests will be accepted, and the server will terminate once all active files (i.e. files that are still open) are closed.

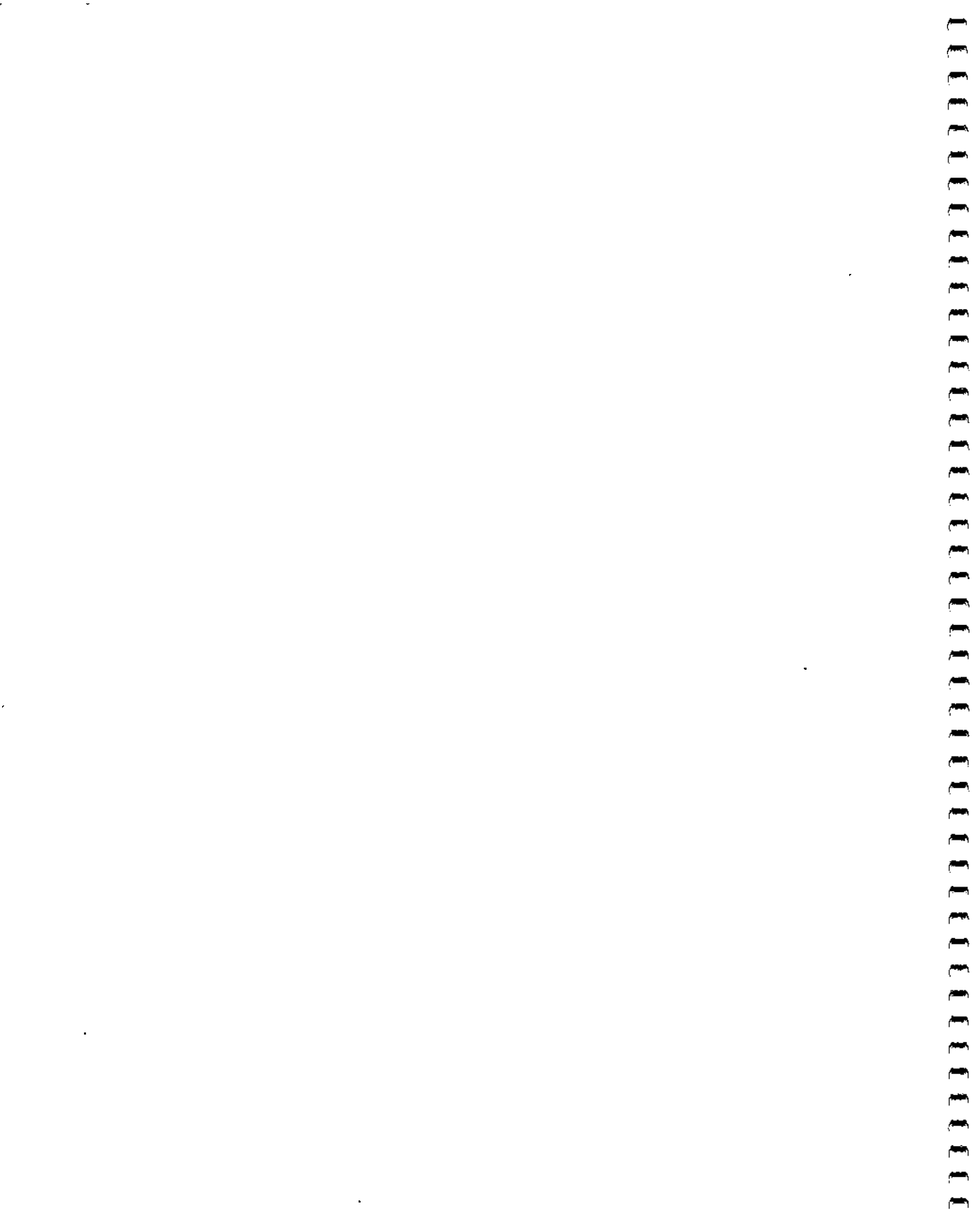
If you don't specify **-i** or **-x**, **Dosfsys** will start up and try to adopt the specified drives.

Exit status: If an error occurs at startup, **Dosfsys** displays an appropriate error message, then exits with one of the following exit codes:

- 1 Exit due to signal.
- 2 Can't register name.
- 3 Can't create proxy.
- 4 Can't register drive prefix.
- 5 Can't set up timer.
- 6 Invalid command-line option(s).
- 7 Error mounting drive.
- 8 Not enough memory for buffers.
- 9 Invalid drive specifier.
- 10 No drive specified on the command line.
- 11 **Dosfsys** must start up in background.

See also: **Fsys**

Utilities Reference: **textto**



Syntax: `emu87 [-f] &`

Options: `-f` Force the emulator to remain, even if floating point hardware is present.

Examples: Start the floating point emulator:

```
emu87 &
```

Description: The **emu87** process traps 80287 and 80387 floating point instructions and emulates them through software. By default, the emulator will quietly terminate if you attempt to start it on a machine equipped with floating point hardware. You can force the emulator to remain active by using the `-f` option.

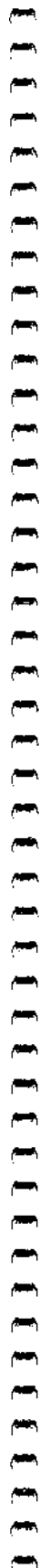
Note that the **emu87** process is required if you wish to run programs that need floating point support on a computer that doesn't have an 80X87 floating point coprocessor.

To remove the emulator, you use the **slay** utility:

```
slay emu87
```

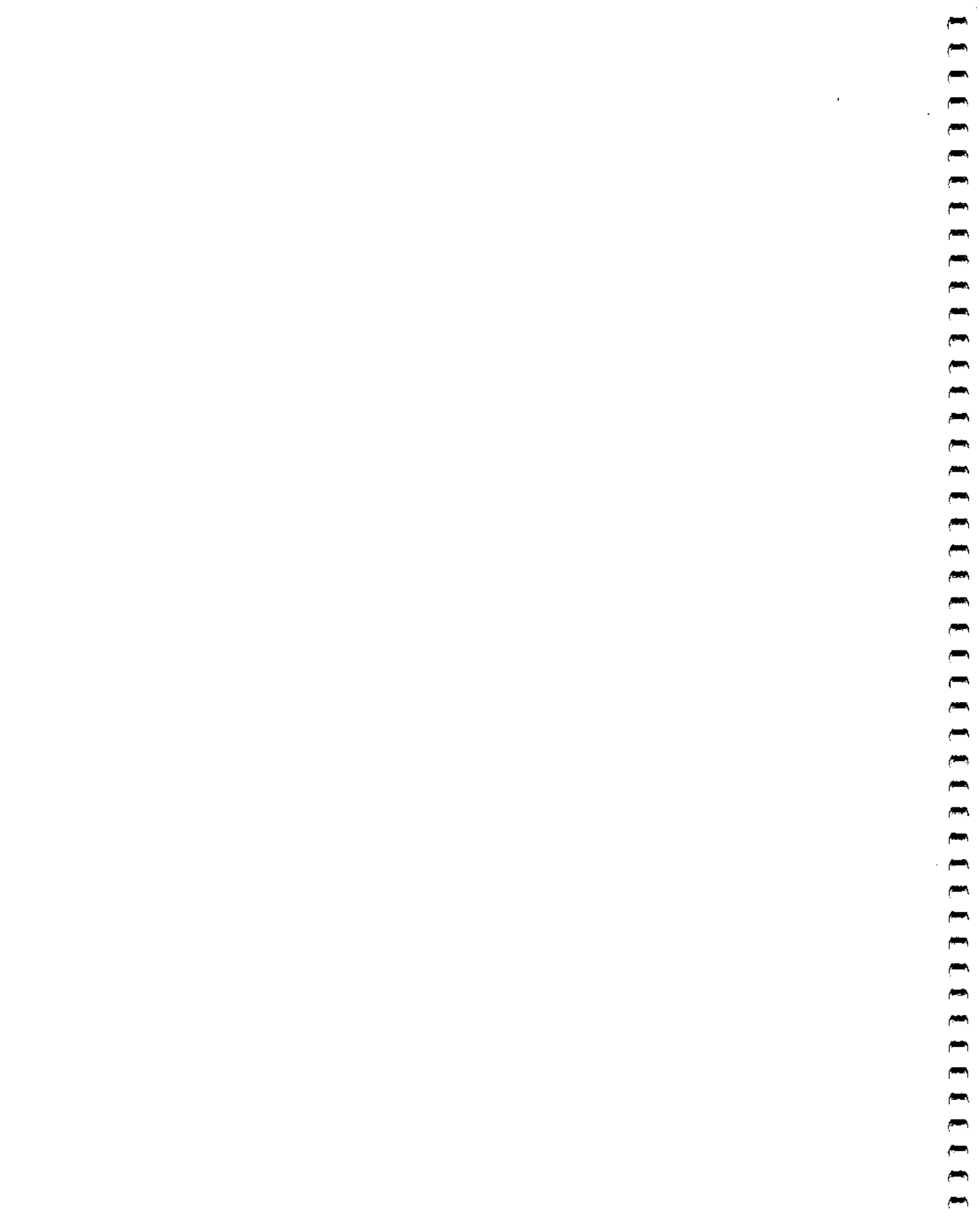


Never attempt to start or stop the emulator while processes are actively using floating point.



Syntax: `fcats [filename...]`

Description: See the **freeze** utility.



Syntax **fdisk** [-p] *drive* [*cmd*]

Options: -p Pause before starting.

drive Disk drive to partition.

cmd An optional installation command, which is one of:

- **add** [-f *slot*] QNX|*type* ALL|HALF|QUARTER|TENTH|*size*
- **boot** [QNX|QNX2|DOS|OS2|*type*|*slot*]
- **loader**
- **show**
- **query** [QNX|QNX2|DOS|OS2|FREE|TOTAL|*type*|*slot*]

where: *type* is the partition specifier (e.g. **t77**), as in the following list:

t77	QNX 4.0
t7, t8, or t9	QNX 2.1
t1, t2, t3, or t4	DOS

size is the size of the partition, in cylinders (e.g. **680**) or megabytes (e.g. **72m**).

slot is partition number **1, 2, 3, or 4**.

Refer to the description for more details on partition specifiers.

Examples: **fdisk** /dev/hd0

fdisk /dev/hd0 add qnx half

Description: The **fdisk** utility lets you partition a hard disk. The partition information matches that used by DOS. The information is kept on the first physical block on the disk.

fdisk

create QNX disk partition (QNX)

To create a QNX partition for the first time you must first start the filesystem and hard disk driver. For example:

```
Fsys &  
Fsys.wd &
```

You should now execute the **fdisk** command:

```
fdisk /dev/hd0
```

and partition your disk.

QNX doesn't automatically mount any partition. Once you've created the partition, you should mount the disk using this partition (see **mount**).

It's important to realize that the **fdisk** command only displays and updates the partition information on the disk. It doesn't directly affect your access to the drive (see **mount**).

The **fdisk** utility is a full-screen, interactive program that is fairly self-explanatory. When **fdisk** is invoked, it displays a screen similar to this one (assuming your disk is already partitioned):

```
FDISK  
Ignore Next Prev 1 2 3 4 Change Delete Boot Unboot Restore Loader Save Quit  
OS  
name type Cylinder Start End Number Blocks Size Boot  
1. DOS ( 4) 0 159 160 39165 19 MB  
2. QNX2.x ( 7) 160 419 260 63700 31 MB  
3. QNX ( 77) 420 854 435 106575 52 MB *  
4. ----- (---) -----  
Choose one of the partitions, by: typing the partition number OR moving the  
pointer with the UP/DOWN arrows; then, choose one of the actions on the top line  
of the screen.  
  
Size : 102 Mbytes Drive : /dev/hd0  
Loader: QNX Config: 7 Heads  
35 Sectors/cyl  
855 Cylinders  
0 = First  
854 = Last
```

You'll see the available commands displayed at the top of the screen. You select a command by typing its first letter, or by moving the cursor to the command (with the arrow keys), and pressing <Enter>.

The commands are:

Next	Move the pointer to the next entry.
Prev	Move the pointer to the previous entry.
1, 2, 3, or 4	Move the pointer to the indicated entry.
Change	Change the selected entry.
Delete	Delete the selected entry.
Boot	Turn on the boot flag for the selected entry.
Unboot	Turn off the boot flag for the selected entry.
Restore	Restore the previous non-QNX bootstrap loader.
Loader	Change the bootstrap loader to the QNX loader.
Save	Save all changes and quit.
Quit	Quit without saving changes.

If you're changing a partition entry, you will be required to enter the partition type number and the start and end cylinders. The other information will be calculated for you by the **fdisk** utility.

Command mode:

The **fdisk** utility can also be run as a non-interactive utility using command-line options. This is useful for designing a custom installation procedure. The command mode syntax is:

```
fdisk drive cmd
```

The commands are:

```
add [-f slot] QNX|type ALL|HALF|QUARTER|TENTH|size
```

Add a new partition entry of the size and type specified. If **fdisk** cannot locate sufficient un-allocated disk space to satisfy your request, it will allocate the largest available portion of the disk (if any).

Using the **-f** (force) option deletes any existing partition in the specified partition number of the partition table. All allocated blocks are freed for reuse before attempting to allocate the requested space. Also, **add** will normally use the first available partition number, but the **-f** option will force it to use *slot*, the specified number.

fdisk

create QNX disk partition (QNX)

boot [**QNX** | **QNX2** | **DOS** | **OS2** | *type* | *slot*]

Boot will turn on the boot flag for the indicated partition. If another partition was already flagged as the primary boot partition, the flag will be turned off for it.

loader

Write the QNX loader to the disk.

show

Show the **fdisk** partition entries.

query [**QNX** | **QNX2** | **DOS** | **OS2** | **FREE** | **TOTAL** | *type* | *slot*]

Exit with a return code which is the size of the indicated entry in megabytes.

where: *type* is the partition specifier (e.g. **t77**), as in the following list:

t77	QNX 4
t7, t8, or t9	QNX 2
t1, t2, t3, or t4	DOS

size is the size of the partition, in cylinders (e.g. **680**) or megabytes (e.g. **72m**).

slot is partition number 1, 2, 3, or 4.



If you wish your disk to contain both QNX and DOS partitions, we recommend that you create the DOS partition **first**, using DOS commands.

Exit status: 0 Successful.

>0 An error occurred.

See also: **dinit**, **fdformat**, **mount**

Caveats: After changing *any* partition that is **mounted**, you **must** reboot. If you don't reboot, you could damage the filesystem on the new partition.

Syntax: `freeze [-Vcdfv] [[+n1,...,n8] [filename]]...`

- Options:**
- v** Display the program's version number and compilation options.
 - d** Uncompress the files from the archive (melt).
 - f** Force creation of the `.F` file. This option creates the `.F` file, even if one already existed, without prompting you for confirmation. The `.F` file will be created even if the compressed file is larger than the original.
 - c** Write the results of the freeze/melt operation to standard output. No files are changed.
 - v** Display a counter. For each file being frozen, display a kilobytes counter, and, after the file is frozen, print a message giving the percentage of the input file's size that has been saved by compression.
 - z** If the output of a melt (option **-d**) is to a tty, allocate a larger output buffer so screen output is in full screen chunks.

+n1,...,n8

A list of 8 numbers separated by commas, specifying the values for the static Huffman table.

Examples: Freeze all files in the current directory:

```
freeze *
```

Extract all C source and header files:

```
melt *. [ch] .F
```

Expand files from a compressed pax archive on floppy:

```
vol -r /dev/fd0 | melt | pax -r
```

View the concatenated contents of all compressed files in the current directory:

```
fcats *.F | more
```

freeze

compress and uncompress files (QNX)

Description: The **freeze** utility compresses the specified files or the standard input. If a file becomes smaller, it's replaced by a file with the extension **.F**, (you can use the **-f** option to force the creation of the **.F** file, even if the compressed file is larger). If no files are specified, the compression is applied to the standard input and is written to the standard output.

If you don't specify the **-f** option, and you run **freeze** in the foreground, you're prompted as to whether the file should be overwritten.



When a **.F** file is created, the original file is removed.

The following links to **freeze** are equivalent to using **freeze** with these options:

This link:	is equivalent to:
melt	freeze -d
fcatt	freeze -cd

You can restore compressed files to their original form through either of two ways:

- specifying the **-d** option to **freeze**
- running **melt** on the **.F** files or on the standard input

When you specify filenames, the ownership, modes, access times, and modification times are maintained between the file and its **.F** version. As a result, **freeze** can be used for archival purposes, yet can still be used with **make** after melting.

The **freeze** utility uses the Lempel-Ziv algorithm on the first pass and the dynamic Huffman algorithm on the second one. The size of “sliding window” is 8K and the maximum length of “matched string” is 256. The positions on the window are coded using a static Huffman table.

A two-byte magic number is prepended to the file to ensure that neither melting of random text nor refreezing of already frozen text is attempted. In addition, the characteristics of the static Huffman table being used during the freeze are written to the file so that these characteristics may be adapted to concrete conditions.

The amount of compression you obtain depends on the size of the input file and the distribution of character substrings and their probabilities. Typically, text files (e.g. C programs) are reduced by 60% to 75%, while executable files are reduced by 50%. Compression is generally much better than that achieved by LZW coding or by Huffman coding, although it takes more time to compute.

An argument preceded by a + defines the values to use for the Huffman table. You may want to explicitly state these values in order to modify the compression algorithm.

Exit status:

- 0 Normal exit.
- 1 An error occurred.
- 2 The last file grew after freezing.

See also: `cpio`, `pax`, `tar`



Syntax: **Fsys** [*options*] &

Options: **-c** *cache_size*

The size of the cache in units of 1K. The default is $\frac{1}{8}$ of the available free memory, up to a maximum of 2048K.

-f *ocb*

The maximum number of *open control blocks* that may be open at one time in the filesystem (default is twice the number of processes).

-i *inode_table_size*

The maximum number of distinct files that may be open at any time (default is $\frac{2}{3}$ of the number of processes).

-l *lock_table_size*

The number of locks that can be simultaneously outstanding (default is 200 entries).

-r *ramdisk_size*

The size of an optional ramdisk, in units of 1K. The minimum size is 4K; the maximum size is 8192K. There is no default size.

-x *cache_proportion*

The maximum proportion of the cache that may be used for caching load modules. For example, **-x 2** would fill up to $\frac{1}{2}$ the cache with load files. The default is 0, which will not cache load modules.

Examples: **Fsys** &

Description: The Filesystem Manager (**Fsys**) provides a standardized means of storing and accessing data on disk subsystems. **Fsys** is responsible for handling all requests to open, close, read, and write to files.

The **Fsys** manager lets you define *ramdisks*. A ramdisk is a block of memory that's accessed by the filesystem as if it were a disk drive. It can be effectively used for temporary files or to augment the cache by storing frequently used data, such as programs or tables. Only one ramdisk is supported by the Filesystem Manager. The ramdisk is

Fsys

Filesystem Manager (QNX)

defined as a block special file with the name `/dev/ram`. To define the root name for the filesystem on the ramdisk, you use the `mount` utility.

Note that since a ramdisk is entirely RAM-based, its contents are lost when the system is rebooted.

Normally, QNX filesystems don't cache load modules. But you can use the `-x` option to specify that up to $\frac{1}{n}$ of the cache may be used to cache frequently used load modules (where n is the *cache_proportion* argument for this option).

The maximum number of distinct files that may be open at any time may be less than the total number of open files, since files with links have only one entry and the same file opened by several processes at the same time has only one entry.

See also: `mount`, **Fsys** drivers (e.g. **Fsys.wd**)

Syntax: **Fsys.aha** [-a *controller*] [-i *interrupt*]
[-m *maxsize*] [-n *name*] [-x on[, off]]

Options:

- a *controller* The controller address (default is to scan for the AHA controller).
- i *interrupt* The interrupt level used by the controller (default is 14).
- m *maxsize* The maximum number of blocks that will be transferred from the SCSI controller to memory in one I/O operation (default is 36).
- n *name* The name of the block special device associated with the SCSI drives. The default is **hd**, which yields a device name of **/dev/hdn**, where *n* is a unit number assigned by **Fsys**.
- x on[, off] Specify the bus on time and bus off time. The default values are 5 μ sec and 9 μ sec, respectively. These values control the amount of time the host adapter will “own” the bus. The default values were chosen to prevent DMA overrun in **Fsys.floppy** or any other DMA device. If no other DMA device is used, values of 11 and 4, respectively, are appropriate for avoiding conflict with dynamic memory refresh. This may provide higher disk throughput.

Description: **Fsys.aha** is the driver for the Adaptec SCSI disk controllers, including these models:

AHA-1540A/1542A
AHA-1540B/1542B
AHA-1642

These devices are generically referred to as AHA devices (Adaptec Host Adapter).

Fsys.aha

Adaptec SCSI host adapter (QNX)

The **Fsys.aha** driver:

- doesn't require any specification of drive parameters—this information is provided by the drive(s)
- uses the Direct Memory Access technology on the AHA controller to transfer data between memory and the SCSI device
- allocates DMA buffers of *maxsize* × 512 for each fixed disk device it detects on the SCSI BUS

The **Fsys.aha** driver makes its code and data available to the local Filesystem Manager (**Fsys**), which then invokes that code and data via direct calls. Once **Fsys** has “adopted” this code and data, **Fsys.aha** terminates as a process.



This driver works only with the Adaptec SCSI controllers, and not with other manufacturers' controllers.

See also: **Fsys.ps2s**, **Fsys** drivers

AHA-1540, 1540A/1542A User's Manual

IBM AT Technical Reference

Caveats: The AHA controller has configuration parameters for setting the transfer speed of its DMA circuitry (the **Fsys.aha** driver doesn't change these values). Setting the transfer speed too high may affect operation of DMA-based devices, including the **Fsys.floppy** driver.

Any drive detected on the SCSI BUS is labeled with its physical unit number. Thus, it's possible to have **/dev/hd2** without having a **/dev/hd1** or a **/dev/hd0**. The current version of **Fsys.aha** supports only PUNs (*Physical Unit Number*); later versions will support LUNs (*Logical Unit Number*) as well.

Since the **-x** option may affect the operation of critical component functions, you should use it with caution. In particular, you should ensure that your chosen values don't interfere with dynamic memory refresh cycles.

Syntax: **Fsys.floppy** [-a *controller*] [-i *interrupt*] [-m *maxsize*]
[-n *name*]

Options:

- a *controller* The controller address (default is 1f0 hex).
- i *interrupt* The interrupt level used by the controller (default is 6).
- m *maxsize* The maximum number of blocks that will be transferred from the diskette to memory in one operation (default is 36).
- n *name* The name of the block special device associated with the floppy. The default is **fd**, which yields a device name of **/dev/fdn**, where *n* is a unit number assigned by **Fsys**.

Description: **Fsys.floppy** is the driver for IBM PC/AT/PS2-compatible floppy disk controllers. These controllers are based on the NEC 765 FDC, the INTEL 8237 DMA controller, and some extra control hardware defined by IBM for the PC.

Since these controllers use the 8237 DMA controller to transfer data between memory and the device, the driver will attempt to allocate up to $maxsize \times 512$ bytes upon startup.

Ideally, *maxsize* should be the number of sectors in a cylinder, allowing the driver to read an entire cylinder in a single request. The default is 36, which is the number of sectors in a cylinder on a 1.44M drive.

The **Fsys.floppy** driver makes its code and data available to the local Filesystem Manager (**Fsys**), which then invokes that code and data via direct calls. Once **Fsys** has “adopted” this code and data, **Fsys.floppy** terminates as a process.



This driver is sensitive to the drive type specified in the non-volatile setup information in AT and PS/2 computers. Specifying the wrong type of drive during this setup might result in sporadic errors.

Fsys.floppy

floppy disk controller (QNX)



The driver recognizes and mounts the proper parameters for diskettes from QNX 4.0, QNX 2.1, and several versions of DOS. If you need to specify additional parameters to the driver, you use the *qnx_ioctl()* C function.

See also: **Fsys**, other **Fsys** drivers

Caveats: You can specify the controller address (via option **-a**), but your floppy disk controller hardware must match the IBM specification, which involves more than the NEC 765 FDC.

Syntax: **Fsys.ps2e** [-a *controller*] [-d *dma_channel*] [-i *interrupt*]
[-m *maxsize*] [-n *name*]

Options:

-a <i>controller</i>	The controller address (default is 3510 hex).
-d <i>dma_channel</i>	The DMA Channel to use (default is 5).
-i <i>interrupt</i>	The interrupt level used by the controller (default is 14).
-m <i>maxsize</i>	The maximum number of blocks that will be transferred from the disk to memory in one operation (default is 36).
-n <i>name</i>	The name of the block special device associated with the hard drive. The default is hd , which yields a device name of /dev/hdn , where <i>n</i> is a unit number assigned by Fsys .

Description: **Fsys.ps2e** is the driver for the IBM PS/2 ESDI Fixed Disk Drive Adapter.

The **Fsys.ps2e** driver:

- doesn't require any specification of drive parameters—this information is provided by the drive(s)
- uses the PS/2 DMA controller to transfer data between the Adapter and the system memory
- allocates *maxsize* × 512 bytes for a DMA buffer

The **Fsys.ps2e** driver makes its code and data available to the local Filesystem Manager (**Fsys**), which then invokes that code and data via direct calls. Once **Fsys** has “adopted” this code and data, **Fsys.ps2e** terminates as a process.



This driver works only with the PS/2 ESDI, and not with the PC/AT ESDI controllers.



Fsys.ps2m

driver for the IBM PS/2 MFM Disk Controller (QNX)

Syntax: **Fsys.ps2m** [-a *controller*] [-c *drivespec*] [-d *dma_channel*]
[-i *interrupt*] [-m *maxsize*] [-n *name*]

Options:

-a <i>controller</i>	The controller address (default is 320 hex).
-c <i>drivespec</i>	The parameters of the drive. You specify these in the following format: <i>heads, cylinders, sectors[, disknum]</i> If you specify this option, you must include <i>heads</i> , <i>cylinders</i> , and <i>sectors</i> . If you omit <i>disknum</i> , unit 0 is assumed.
-d <i>dma_channel</i>	The DMA channel to use (default is 3).
-i <i>interrupt</i>	The interrupt level used by the controller (default is 14).
-m <i>maxsize</i>	The maximum number of blocks that will be transferred from the disk to memory in one operation (default is 34).
-n <i>name</i>	The name of the block special device associated with the hard disk. The default is hd , which yields a device name of /dev/hdn , where <i>n</i> is a unit number assigned by Fsys .

Description: **Fsys.ps2m** is the driver for the IBM PS/2 MFM Disk Controller.

The **Fsys.ps2m** driver:

- transfers data between the drive and memory using the PS/2 DMA Controller
- allocates *maxsize* × 512 bytes of memory as a DMA buffer.

In order to obtain drive parameters, **Fsys.ps2m** examines values established by the BIOS of your computer. If necessary, you can override these default parameters via the **-c** option.

Fsys.ps2m

driver for the IBM PS/2 MFM Disk Controller (QNX)

The **Fsys.ps2m** driver makes its code and data available to the local Filesystem Manager (**Fsys**), which then invokes that code and data via direct calls. Once **Fsys** has “adopted” this code and data, **Fsys.ps2m** terminates as a process.

driver for the PS/2 SCSI controller (QNX)

Syntax: **Fsys.ps2s** [-a *controller*] [-i *interrupt*] [-m *maxsize*]
[-n *name*]

Options:		
-a <i>controller</i>	The controller address (default is 3540 hex).	
-i <i>interrupt</i>	The interrupt level used by the controller (default is 14).	
-m <i>maxsize</i>	The maximum number of blocks that will be transferred from the disk to memory in one operation (default is 34).	
-n <i>name</i>	The name of the block special device associated with the hard disk. The default is hd , which yields a device name of /dev/hd<i>n</i> , where <i>n</i> is a unit number assigned by Fsys .	

Description: **Fsys.ps2s** is the driver the IBM PS/2 SCSI Disk Drive Adapter.

The **Fsys.ps2s** driver:

- doesn't require any specification of drive parameters—this information is provided by the drive(s)
- uses the PS/2 SCSI DMA controller to transfer data between the Adapter and the system memory
- allocates $maxsize \times 512$ bytes for a DMA buffer.

The **Fsys.ps2s** driver makes its code and data available to the local Filesystem Manager (**Fsys**), which then invokes that code and data via direct calls. Once **Fsys** has “adopted” this code and data, **Fsys.ps2s** terminates as a process.



This driver works only with the IBM PS/2 SCSI, and not with the Adaptec SCSI controllers—or any other SCSI controllers—for the IBM PS/2.



Syntax: `Fsys.qic02 [-L type] [-a address] [-d dma]
 [-i interrupt] [-m max_buffers] [-n name]`

~~-t~~ ~~-L~~ *type* The type of QIC-02 unit, where *type* can be one of:

`archive`
`tecmar_internal`
`tecmar_external`

The default is `archive`.

`-a address` The base address, in hex, of the controller (default is 200).

`-d dma` The DMA channel to use (default is 3).

`-i interrupt` The interrupt level to use (default is 5).

`-m max_buffers` The maximum number of blocks that can be transferred to or from the tape controller (default is 32).

`-n name` The name to associate with the device. The default is `tp`, which yields a device name of `/dev/tp0`.

Description: `Fsys.qic02` is the driver for tape controllers conforming to the QIC-02 specification.

The `Fsys.qic02` driver makes its code and data available to the local Filesystem Manager (`Fsys`), which then invokes that code and data via direct calls. Once `Fsys` has “adopted” this code and data, `Fsys.qic02` terminates as a process.

See also: `Fsys` drivers



Syntax: **Fsys.wd** [-a *controller*] [-c *drivespec*] [-i *interrupt*]
[-m *maxsize*] [-n *name*] [-7|8] [-s]

- Options:**
- a *controller* The controller address (default is 3f0 hex).
 - c *drivespec* The parameters of the drive. You specify these in the following format:

heads, cylinders, sectors [, disknum]

If you specify this option, you must include *heads*, *cylinders*, and *sectors*. If you omit *disknum*, unit 0 is assumed.
 - i *interrupt* The interrupt level used by the controller (default is 14).
 - m *maxsize* The maximum number of blocks that will be transferred from the disk to memory in one operation (default is 34).
 - n *name* The name of the block special device associated with the hard disk. The default is **hd**, which yields a device name of **/dev/hdn**, where *n* is a unit number assigned by **Fsys**.
 - 7 Don't reprogram the Fixed Disk Control Register.
 - 8 Enable a bit in the Fixed Disk Control Register to allow drives with more than seven heads.
 - s Don't program the drive parameters into the **wd** controller.

Description: **Fsys.wd** is the driver for the Fixed Disk interface defined by IBM for the PC/AT computer. This interface is based upon a Western Digital 1010-05 Winchester Disk Controller. Many controllers are register-level compatible with this controller, including AT MFM, RLL, IDE, WD1007ESDI, etc.

In order to obtain drive parameters, **Fsys.wd** examines values established by the BIOS of your computer. If necessary, you can override these default parameters via the **-c** option.

Fsys.wd

driver for PC/AT fixed disk interface (QNX)

The **-m** *maxsize* option defines the largest single read or write the driver will support. The default value of 34 is appropriate for a non-interleaved drive. For other configurations, you can alter this value. For example, you can specify 17 for a 2:1 interleave.

The **Fsys.wd** driver makes its code and data available to the local Filesystem Manager (**Fsys**), which then invokes that code and data via direct calls. Once **Fsys** has “adopted” this code and data, **Fsys.wd** terminates as a process.

See also: **Fsys**, **Fsys** drivers

Syntax: **license** [*source* [*destination*]]

Options: *source* The source of the serial number, which can be one of:

 /**dev/fd***n* A floppy drive that the serial number
 is transferred from (*n* can be either 0
 or 1).

directory A directory that all serial number files
 are transferred from.

file A specific serial number file to
 transfer.

 The default is **/dev/fd0**.

destination The pathname of the **etc/licenses** directory that is
 to be updated (default is the **/etc/licenses**
 directory in the current filesystem).

Examples: Copy the serial number from the floppy diskette in drive 0 to the
 /etc/licenses directory:

```
license /dev/fd0
```

Copy the serial number from the floppy diskette in drive 0 on node
10 to the **/etc/licenses** directory:

```
license //10/dev/fd0
```

Copy all serial numbers from the **/etc/licenses** directory on
node 1 to the **/etc/licenses** directory on node 5:

```
license //1/etc/licenses //5/etc/licenses
```

Description: The **license** utility transfers serial numbers to your hard disk.
 These serial numbers are used by the operating system and some
 applications to verify the licensing of the software. The serial
 numbers are placed in files under the **/etc/licenses** directory.

When you receive a new license for your network or for an
application that is licensed to you, you should place the diskette
containing the license serial number in a floppy drive and run the
license utility.

license

license software for use (QNX)



Before you run the **license** utility, make sure the floppy driver has been started.

The utility will copy the serial number from the diskette to your hard disk. Note that the new serial number will not become effective until you reboot.

Nodes that boot over the network will inherit serial numbers from the node they boot from. If you have more than one machine that boots from disk in a local area network, you can specify a *directory* to the **license** utility to copy all serial numbers from one hard disk to another.

Note that if you don't specify a *destination*, the **license** utility always writes to the **/etc/licenses** directory in the current filesystem.

Syntax: `lpsrvr [-L logfile] [-c file] [-n name] [-q] &`

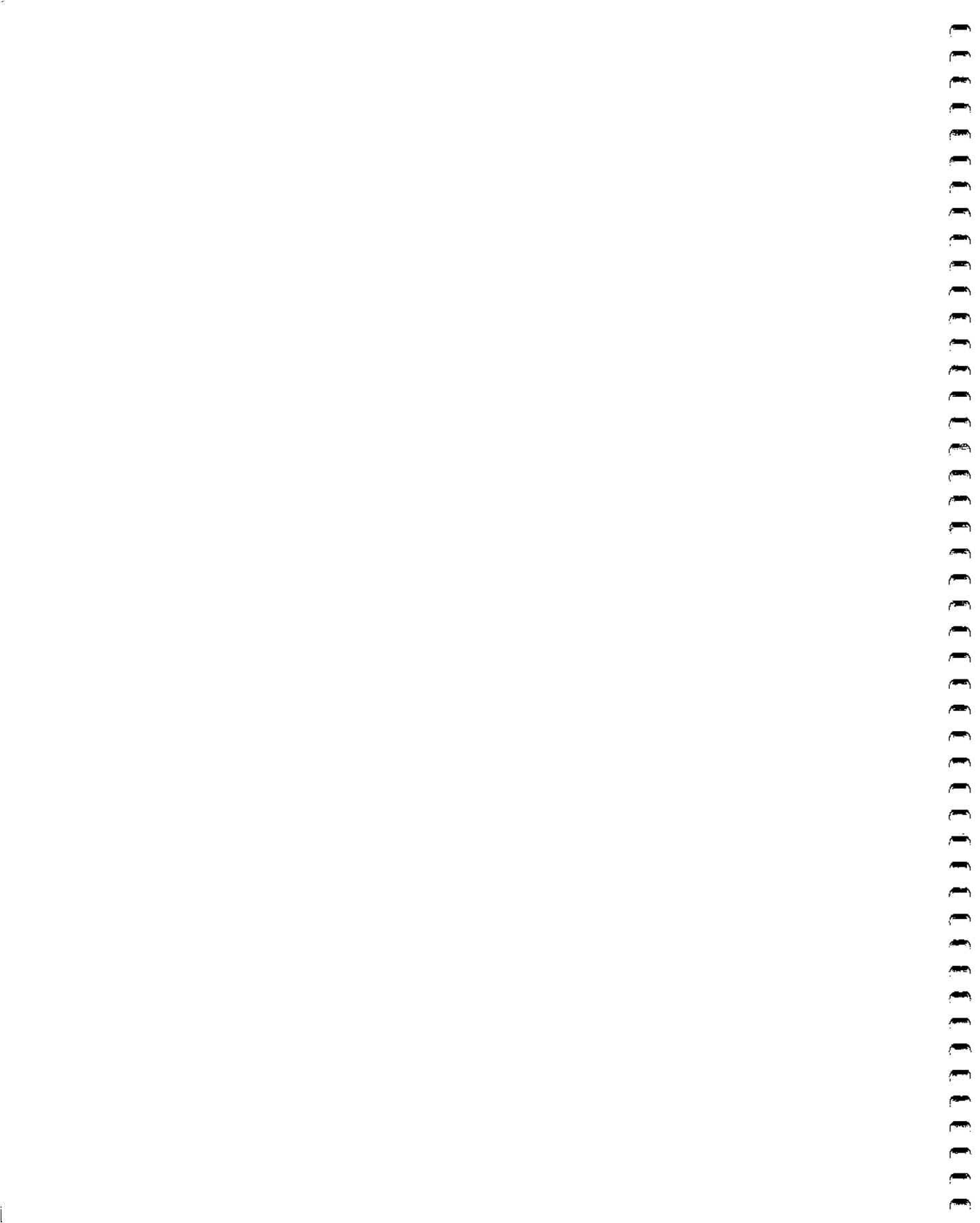
Options:

- `-L logfile` Print log information into *logfile* rather than to standard error.
- `-c file` Use *file* for configuration information. Default is the `/usr/spool/lp/.setup` file.
- `-n name` Register using *name*, rather than the `/qnx/spooler` file.
- `-q` Be quiet. Don't generate log information.

Description: The `lpsrvr` utility is the print server. The utility accepts requests for printing, rescheduling, and examination of print jobs. When `lpsrvr` starts, it scans the configuration file to find out about existing printers, then scans for any residual work from a previous invocation.

Files: `/usr/spool/lp/.setup`

See also: *Utilities Reference:* `lp`, `lpc`, `lpq`, `lprm`



Syntax: **melt** [-Vc**f**v] [*filename...*]

Description: See the **freeze** utility.



Syntax: **Mouse** [-b [-i *interrupt*] [-p *address*]] [-g *number*]
 [-n *name*] [-s *baud*] [-x] [-y] *type* <*serial_port* &

Mouse test

Options: -b Assume a bus mouse (default is serial).
 -g *number* The gain to apply to mouse motions (default is 1).
 -i *interrupt* The hardware interrupt to use for the bus mouse
 (default is 5).
 -p *address* The I/O address of the bus mouse, in hex
 (default is 23C).
 -n *name* The name to register (default is **qnx/mouse**).
 -s *baud* Set the serial port to this baud rate (default is
 1200 baud).
 -x Reverse x motions.
 -y Reverse y motions.
 type The type of mouse (see Description).
 <*serial_port* The port your serial mouse is attached to.

Examples: Assume a Microsoft Serial Mouse on /dev/ser1:

Mouse msoft </dev/ser1 &

Assume a standard bus mouse:

Mouse -b msoft &

Test an installed mouse driver:

Mouse test

Description: The **Mouse** driver manages the mouse on a QNX system. It accepts and interprets the stream of data generated by a moving mouse, and processes this into an internal first-in first-out queue of significant mouse events. Application programs can “read” these mouse events

Mouse

mouse driver (QNX)

by sending messages to the Mouse server via a set of mouse-specific functions.

To minimize buffer requirements, yet ensure a consistent mouse position, **Mouse** “compresses” unread mouse motions. A mouse event is always generated when the user presses or releases a mouse button.

To ensure that mouse motions apply only to the visible console, and don’t interfere with background programs, **Mouse** tracks the system console and flushes all mouse events if the visible console is changed.

At the time of printing, the following mouse *types* are supported:

msoft	Standard 200 DPI Microsoft Mouse
msys	Mouse Systems Mouse
ps2	IBM PS/2 Integrated Mouse

To see a list of currently supported mouse types, you type:

```
use mouse
```

See also: **Dev**, **Dev** drivers, C language *mouse_* functions

Syntax: `netboot [-f netfile] [-n netname] [-v] &`

Options: `-f netfile` The name of the network configuration file. This file contains information on each node that can boot over the network. The default is:

`/etc/config/netboot`

`-n netname` The registered name of the Network Manager process to receive network boot requests. The default is:

`qnx/arcnet`

`-v` Be verbose. Print each boot request to the standard output.

Examples: `netboot &`

Description: The **netboot** utility accepts boot requests from machines attempting to boot over the network. The **netboot** utility must be running as a background process on the boot machine before another (booting) machine can boot. Upon receiving a boot request, **netboot** opens the network configuration file and uses the node ID of the booting machine as a key to locate the boot information. An operating system is then prepared and transmitted to the requesting machine.

The network configuration file is an editable text file containing one line for each node that may be booted. Blank lines and lines that start with a double quote (") are ignored. Lines must be in the following format:

`node-number F=imagefile | f=buildfile`

You may use spaces or tabs to separate each token. Note that you may use `*` instead of *node-number* to match any requesting node. If you do this, make sure the line is **last** in the file, since all nodes will match and look no further.

netboot

service boot requests from the network (QNX)

If an image file (**F=**) is specified, it must contain the path of an operating system image that was built via the **buildqnx** utility. If a relative pathname is used, it is assumed to be relative to the **/boot** directory. For example:

F=images/ws is equivalent to: **F=/boot/images/ws**

If a build file (**f=**) is specified, the **buildqnx** utility will be invoked on the fly to build an operating system image that is piped to **netboot** and shipped as an image over the network. If a relative pathname is used, it is assumed to be relative to the **/boot** directory. For example:

f=build/ws is equivalent to: **f=/boot/build/ws**

Build files are much smaller than operating system images, so if you have a number of different images you want loaded, you can save considerable disk space by building them on the fly. Images are sometimes useful for freezing a particular operating system image that you wish to be immune from subsequent operating system module updates. In this case, you would have to build a new image using **buildqnx** to activate any new modules.

QNX is a modular operating system. As such, you don't need to build into the transmitted image all modules you may require. Once the operating system loads, you can start many subsequent modules from the command line, or more typically, from a shell script that is executed right after boot. This is described in more detail under the **buildqnx** and **sinit** utilities.

You may run **netboot** on more than one machine in the network. This is advisable in larger networks to speed up booting when many requests are active at one time.

See also: **buildqnx, netnode, netpoll, sinit**

Utilities Reference: **netinfo**

Syntax: `netnode [-a|d nid[-end_nid]]`

Options:

- `-a nid[-end_nid]` Allow `qnx_vc_attach()` to node *nid*, or to a range of nodes, from *nid* to *end_nid*.
- `-d nid[-end_nid]` Disallow `qnx_vc_attach()` to node *nid*, or to a range of nodes, from *nid* to *end_nid*.

Examples: To all of your `/etc/config/sysinit.xxx` files, add the line

```
net_disallow
```

Create a `/usr/bin/net_disallow` script file, which might look something like:

```
netnode -d1:4 -d7 -d11:255
```

This would be for a 10-node network where nodes 1, 2, 3, 4, and 7 are always running a different operating system; nodes 11 through 255 are masked off, because they are known to not exist.

Description: The **netnode** utility lets you display or change the Arcnet node permit bitmap to optimize network response.

Arcnet internally maintains a bitmap of all 255 possible nodes, which is used to determine whether or not a `qnx_vc_attach()` is permitted to another node on the network. When Arcnet first starts up, by default all nodes are permitted.

The benefit of the node permit list is to speed up commands like **sin net** on a network containing a mix of nodes in which some are running QNX 4.0 and others aren't. If an application on a QNX 4.0 node attempts to attach a virtual circuit to a node that isn't running QNX 4.0, the destination node would normally accept the low-level control packet from the QNX 4.0 node, but not recognizing the packet, would discard it. The application on the QNX 4.0 node, patiently waiting for a response, is eventually released by the QNX 4.0 self-polling, but this typically happens slowly.

The **netnode** utility can be used to avoid this situation. By knowing which nodes "do not exist", the system can avoid sending packets to non-QNX nodes in the first place.

netnode

display or change the Arcnet node permit bitmap (QNX)

The **-a** option, which re-allows virtual circuits to the node or node range, is added for completeness, but will probably be rarely used. The list of masked-off nodes should not change often.

To see the current list of disallowed nodes, invoke **netnode** with no arguments.



QNX 4.0 nodes that aren't on the node permit list cannot communicate with each other.

See also:

netboot

Utilities Reference: **netinfo**

Syntax: `netpoll [-N node] [-i idle_time] [-n num_polls]
[-p poll_period]`

Options:

<code>-N node</code>	The node on which to change the poll parameters. The default is the node from which the utility is invoked.
<code>-i idle_time</code>	The length of time, in seconds, that the system will allow a <i>virtual circuit</i> (VC) to remain idle before checking the VC's integrity (default is 10). The integrity check consists of transmitting a network poll packet to the remote node.
<code>-n num_polls</code>	The number of times that the system will transmit a network poll packet to a remote node before it concludes that the remote node is permanently unreachable (default is 6).
<code>-p poll_period</code>	The length of time, in seconds, between checks for idle VCs (default is 10).

Examples: The following example:

- 1) reduces to an average of 10 seconds the time allowed to detect a node "disappearing" ($poll_period \times num_polls$), at the expense of increased CPU consumption
- 2) increases the frequency of the VC polling
- 3) decreases the maximum number of allowed polls:

```
netpoll -p 2 -i 2 -n 5
```

The following example:

- 1) increases to an average of 200 seconds the time allowed to detect a node "disappearing" ($poll_period \times num_polls$)
- 2) decreases the frequency of the VC polling
- 3) increases the maximum number of allowed polls:

```
netpoll -p 20 -i 20 -n 10
```

netpoll

display or change default polling parameters of a node (QNX)

Description: In QNX, virtual circuits (VCs) provide transparent, detached interprocess communication between processes residing on two different nodes in a network. If a node abruptly powers down, or if it can no longer be communicated with, the resources associated with the VCs to that node should be recovered.

Each node polls its own VCs, checking their integrity. The **netpoll** utility lets you view or change the default polling parameters of a specific node.

To display the polling parameters for a node, don't specify any parameters; if the node is remote, specify only the **-N node** option.

See also: **netboot**, **netnode**

Utilities Reference: **netinfo**

Syntax: List archive contents:

```
pax [-cimopuvy] [-f archive] [-s replstr]... [-t device]
      [pattern...]
```

Read an archive:

```
pax -r [-cimnopusvy] [-f archive] [-s replstr]...
      [-t device] [pattern...]
```

Write an archive:

```
pax -w [-adimuvy] [-b blocking] [-f archive]
      [-s replstr]... [-t device] [-x format] [pathname...]
```

Copy files:

```
pax -rw [-ilmopusvy] [-s replstr]... [pathname...] directory
```

Options:

-a Append the files specified by *pathname* to the specified archive.

-b *blocking*

Block the output at *blocking* bytes per write to the archive file. A **k** suffix multiplies *blocking* by 1024, a **b** suffix multiplies *blocking* by 512, and an **m** suffix multiplies *blocking* by 1048576 (1 megabyte). In QNX, the maximum buffer size is 32K-1. If not specified, *blocking* is automatically determined on input and is ignored for **-rw** (copy files).

-c Complement the match sense of the *pattern* operands.

-d Don't create intermediate directories not explicitly listed in the archive. This option is applied only if the **-r** option is specified.

-f *archive*

Use *archive* as the pathname of the input or output archive, overriding the default of standard input for **-r** or standard output for **-w**.

-i Interactively rename files. Substitutions specified by **-s** options are performed before requesting the new filename from the user. A file is skipped if an empty line is entered. If EOF is encountered, **pax** exits with an exit status of 0.

pax

portable archive interchange (POSIX)

- l When possible, link rather than copy files.
- m Don't keep file modification times.
- n When **-r** is specified, but **-w** is not, treat the *pattern* operands as ordinary filenames. Only the first occurrence of each of these files in the input archive is read. The **pax** utility exits with an exit status of 0 after all files in the list have been read. If one or more of the files in the list aren't found, **pax** writes a diagnostic to standard error for each of these files and exits with a nonzero exit status. The filenames are compared before any of the **-i**, **-s**, or **-y** options are applied.
- o Restore file ownership as specified in the archive. The invoking process must have appropriate privileges to accomplish this.
- p Preserve the access time of the input files after they have been copied.
- s *replstr*
Modify filenames according to the substitution expression. The syntax for the expression is:

 -s /old/new/[gp]

Any non-null character may be used as a delimiter (a / is used here as an example). Multiple **-s** expressions are applied in the order specified terminating with the first successful substitution. The optional trailing **p** causes successful mappings to be listed on standard error. The optional trailing **g** causes the old expression to be replaced each time it occurs in the source string. Files that substitute to an empty string are ignored both on input and output.
- t *device*
The *device* argument names the input or output archive device, overriding the default of standard input for **-r** and standard output for **-w**.
- u Copy each file only if it is newer than a preexisting file with the same name. This implies option **-a**.

-v Be verbose; list filenames as they are encountered. This option produces a table of contents listing on the standard output when both **-r** and **-w** are omitted; otherwise the filenames are printed to standard error as they are encountered in the archive.

-x *format*

Use this output archive format. The input format, which must be one of the following, is automatically determined when the **-r** option is used. The supported formats are:

cpio—the extended **cpio** interchange format specified in IEEE Std. 1003.1-1988.

ustar—The extended **tar** interchange format, also specified in IEEE Std. 1003.1-1988. This is the default archive format.

-y Interactively prompt for the disposition of each file. Substitutions specified by **-s** options (described above) are performed before the user is prompted for disposition. EOF or an input line starting with the character **q** causes **pax** to exit. Otherwise, an input line starting with anything other than **y** causes the file to be ignored. This option cannot be used in conjunction with the **-i** option.



Only the last of multiple **-f** or **-t** options take effect.

<i>directory</i>	The destination directory pathname for copies when both the -r and -w options are specified. The directory must exist and you must have the appropriate write permissions, or an error results.
<i>pathname</i>	A file whose contents are used instead of the files named on the standard input. When a directory is named, all of its files and (recursively) subdirectories are copied as well.
<i>pattern</i>	A pattern given in the standard shell pattern-matching notation. If no pattern is specified, the default is * , which selects all files.

pax

portable archive interchange (POSIX)

Modes of operation

If you don't specify **-r** or **-w**, then **pax** lists the contents of the specified archive. In this mode, **pax** lists normal files one per line. Hard link pathnames are listed as:

pathname == linkname

where *pathname* is the name of the file being extracted, and *linkname* is the name of a file that appeared earlier in the archive.

If **-v** is specified, then **pax** lists normal pathnames in the same format used by the **ls** utility with the **-l** option. Hard links are shown as:

<ls -l listing> == linkname

- w** Write the files and directories specified by *pathname* operands to the standard output, together with the pathname and status information prescribed by the archive format used. A directory *pathname* operand refers to the files and (recursively) subdirectories of that directory. If no *pathname* operands are given, then the standard input is read to get a list of pathnames to copy, one pathname per line. In this case, only those pathnames appearing on the standard input are copied.
- r** Read an archive file from the standard input; select for extraction only those files whose names match any of the *pattern* operands. The selected files are conditionally created and copied relative to the current directory tree, subject to the options chosen. By default, the owner and group of selected files will be that of the invoking process, and the permissions and modification times will be the same as those in the archive.

The supported archive formats are automatically detected on input. The default output format is **ustar**, but may be overridden by the **-x** format option described below.

- rw** Read the files and directories named in the *pathname* operands and copy them to the destination directory. A directory *pathname* operand refers to the files and (recursively) subdirectories of that directory. If no *pathname* operands are given, the standard input is read to get a list of pathnames to copy, one pathname per line. In this case, only those pathnames

appearing on the standard input are copied. The directory named by the directory operand must exist and must have the proper permissions before the copy can occur.

Examples: Copy the contents of the current directory to the floppy drive:

```
pax -w -f /dev/fd0 .
```

Copy the contents of **olddir** to **newdir**:

```
mkdir newdir
cd olddir
pax -rw . newdir
```

Read the archive **pax.out** with all files rooted in **/usr** in the archive extracted relative to the current directory:

```
pax -r -s "/usr/,, " -f pax.out
```

Description: The **pax** utility reads and writes archive files that conform to the archive/interchange file format specified in IEEE Std. 1003.1-1988. The utility can also read, but not write, a number of other file formats. Support for these traditional file formats (such as V7 **tar** and System V binary **cpio** format archives) is provided for backward compatibility and to maximize portability.

The **pax** utility will also support traditional **cpio** and System V **tar** interfaces if invoked with the name “**cpio**” or “**tar**” respectively.

The **pax** utility is capable of reading and writing archives that span multiple physical volumes. Upon detecting an end of medium on an archive that is not yet completed, **pax** will prompt you for the next volume of the archive and will let you specify the location of the next volume.

Combinations of the **-r** and **-w** command-line arguments specify whether **pax** will read, write, or list the contents of the specified archive, or will move the specified files to another directory.

When writing to an archive, the standard input is used as a list of pathnames if no *pathname* operands are specified. The format is one pathname per line. Otherwise, the standard input is the archive file,

pax

portable archive interchange (POSIX)

which is formatted according to one of the format specifications in IEEE Std. 1003.1-1988.

The user ID and group ID of the process, together with the appropriate privileges, affect the ability of **pax** to restore ownership and permissions attributes of the archived files. (See Archive/Interchange File Format in IEEE Std. 1003.1-1988.)

Note that the options **-a**, **-c**, **-d**, **-i**, **-l**, **-p**, **-t**, **-u**, and **-y** are provided for functional compatibility with the historical **cpio** and **tar** utilities. The option defaults were chosen based on the most common usage of these options, so some of the options have meanings different from those of the historical commands.

Files: The controlling terminal (**/dev/tty**) is used to prompt the user for information when the **-i** or **-y** options are specified.

Exit status: 0 All files in the archive were processed successfully.
>0 The **pax** utility aborted due to errors encountered during operation.

Caveats: Special permissions may be required to copy or extract special files.

Device, user ID, and group ID numbers larger than 65535 cause additional header records to be output. These records are ignored by some historical versions of **cpio** and **tar**.

The archive formats described in Archive/Interchange File Format have certain restrictions that have been carried over from historical usage. For example, there are restrictions on the length of pathnames stored in the archive.

When getting an **ls -l** style listing on **tar** format archives, link counts are listed as zero since the **ustar** archive format doesn't keep link count information.

On 16-bit architectures, the largest buffer size is 32K-1. This is due, in part, to using integers in the buffer allocation schemes. On many of these machines, however, it is not possible to allocate blocks of memory larger than 32K.

See also: **cpio**, **tar**

Buildfile syntax: **Proc** [*options*]

Options:

- a** Disable time accounting. This option saves $16 \times \text{num_procs}$ bytes of memory and slightly reduces the work done in the timer interrupt handler.

- d** *num_dev*
 The number of major device numbers supported (default is 32).

- f** *min_fds max_fds extra_fds*
 This option controls the availability of file descriptors (FDs):
 - min_fds* The minimum number of FDs that must be available for a process. This argument defaults to 16 and should be left at that value if POSIX-compliance is a requirement (POSIX requires that a process be capable of opening at least 16 files). You can save some memory by reducing this number.

 - max_fds* The maximum number of FDs that can be available for a process (default is 32).

 - extra_fds* The pool of extra FDs that can be allocated, as needed, to processes that exceed their minimum number of FDs.

- i** *irq* Make *irq* the highest-priority hardware interrupt in the system (default is 3).

- m** *num_shared_segs*
 The number of shared segments (managed by the *qnx_segment()*... family of functions) that will be available in this operating system image (default is 300).

-n *num_names*

The maximum number of names that can be registered locally by the *qnx_name_attach()* function (default is 30).

-p *num_procs*

The total number of real processes, virtual circuits, and proxies that can exist at any time (default is 150).

-r *prefix_len*

The size of the system prefix space (default is 400 characters).

-s *num_sessions*

The maximum number of sessions that will be supported (default is 32).

-t *num_timers*

The number of timers that will be supported (default is 40 per system). These timers are allocated to processes with the *mktimer()* function and returned with the *reltimer()* function.

-v *level*

Set the verbosity *level* (0 to 2; default is 1). Messages are always displayed on the console in the form:

QNX (*level*) : Message

The messages should be interpreted as follows:

Level 0 : An internal inconsistency was detected. Contact Quantum.

Level 1 : The Kernel was forced to take a planned, but uncommon, recovery action.

Level 2 : Expected actions have occurred as a result of a misbehaving process or remote computer.

Description:

The Process Manager (**Proc**) works closely with the Kernel to provide essential operating system services. The Kernel is in fact bundled with the code for **Proc** and shares the same address space. For these reasons, **Proc** is a required system process in all operating images that are built. You cannot start **Proc** after booting.

Most of the services provided by **Proc** require the use of memory within **Proc**. The amount of memory used is controlled by tables whose sizes are set using command-line options. Some tables are fixed in size, while others dynamically request more memory as needed. Unlike other system processes, **Proc** cannot grow its data segment at runtime. This means that you must provide sufficient internal memory (referred to as heap space) when you build **Proc** into an OS image. This heap space is provided as a number in a *build_file* used by the **buildqnx** utility. The following formula may be used to approximate **Proc**'s minimum heap space requirements:

$$\begin{aligned} \text{heap} = & \text{num_dev} \times 2 \\ & + \text{num_shared_segs} \times 1 \\ & + \text{num_names} \times 8 + (\text{total length of registered names}) \\ & + \text{num_procs} \times 148 + (\text{total length of unique command names}) \\ & + \text{each_code or data_segment} \times 12 \\ & + \text{prefix_len} \\ & + \text{num_sessions} \times 20 \\ & + \text{num_timers} \times 24 \\ & + 8000 \end{aligned}$$

The number of code and data segments will be determined by the programs you run. A small model program requires one code and one data segment. While the system is running, you may use the **sin info** utility to display the amount of free heap space available in **Proc**.



Syntax: `sinit [-i initfile] [-r root] [-s shell] [var=value]...`

- Options:**
- `-i initfile` Open this system initialization file (for defaults, see the Description below).
 - `-r root` Define an alias prefix for the filesystem root (/).
 - `-s shell` Invoke this shell (default is `/bin/sh`).
 - `var=value` Define an environment variable.

Examples:

```
sinit -r //1/
sinit -s /bin/myshell
```

Description: The `sinit` utility sets up the initial operating system environment immediately after boot. It is usually bound to the operating system by means of the `buildqnx` utility.

The `sinit` utility attempts to find and open the system initialization file for your node. This is a text file that contains a list of system initialization commands for which `sinit` invokes a shell to execute.

There are two types of boots: a *normal boot* and an *alternate boot*. All boots over the network are considered normal. When you boot from disk, you are prompted with the option of doing an alternate boot. The type of boot you choose determines which system initialization file will be used:

If you choose:	sinit tries to find and open:
a normal boot	<code>/etc/config/sysinit.<i>node</i></code> If this fails, it tries: <code>/etc/config/sysinit</code>
an alternate boot	<code>/etc/config/altsysinit</code> If this fails, it tries: <code>/etc/config/sysinit</code>

It's usually a good idea to keep the `sysinit` file generic and create custom `sysinit.node` files for each node as required. The `altsysinit` file is usually a backup of the `sysinit.node` file for the node that boots from its own hard disk.

sinit

system initialization (QNX)

Whenever you wish to get your files from another node (remote file server), you use the **-r root** option. If you boot from hard disk and you get your files from your local hard disk, you don't need to specify this option.

You can override the default search for **sysinit.node** or **altsysinit** by specifying an explicit filename with the **-i initfile** option.

The default shell invoked by **sinit** is **/bin/sh**. To override this default, you use the **-s shell** option. You should specify a complete pathname for the shell, starting with a slash (/).

You can cause **sinit** to create an environment variable using the **var=value** operand. These environment variables will be inherited by the system initialization shell. The time zone variable (TZ) is often set here.



The **sinit** utility always terminates; it doesn't stay resident in the system. If you wish to start up user sessions, you should use the terminal initialization utility, **tinit**, and place it in the appropriate **sysinit** file.

See also:

buildqnx

Utilities Reference: **tinit**

Syntax: **tape** [**bot...**] [**eod...**] [**erase...**] [**forward...**]
 [**retension...**] [**status...**]

Options:

bot	Position the tape to the beginning.
eod	Position the tape to the end of the data (perhaps to continue writing after a previous archive).
erase	Erase the contents of the tape.
forward	Position the tape to just after the next file mark.
retension	Position the tape to the beginning, then to the end, then back to the beginning.
status	Print information concerning the current condition of the tape drive.

Examples: Position the currently mounted tape such that data may be appended after previously written archives:

```
tape eod
```

Position the tape such that the third archive on the tape may be read:

```
tape bot forward forward
```

Position the tape such that the first archive may be read:

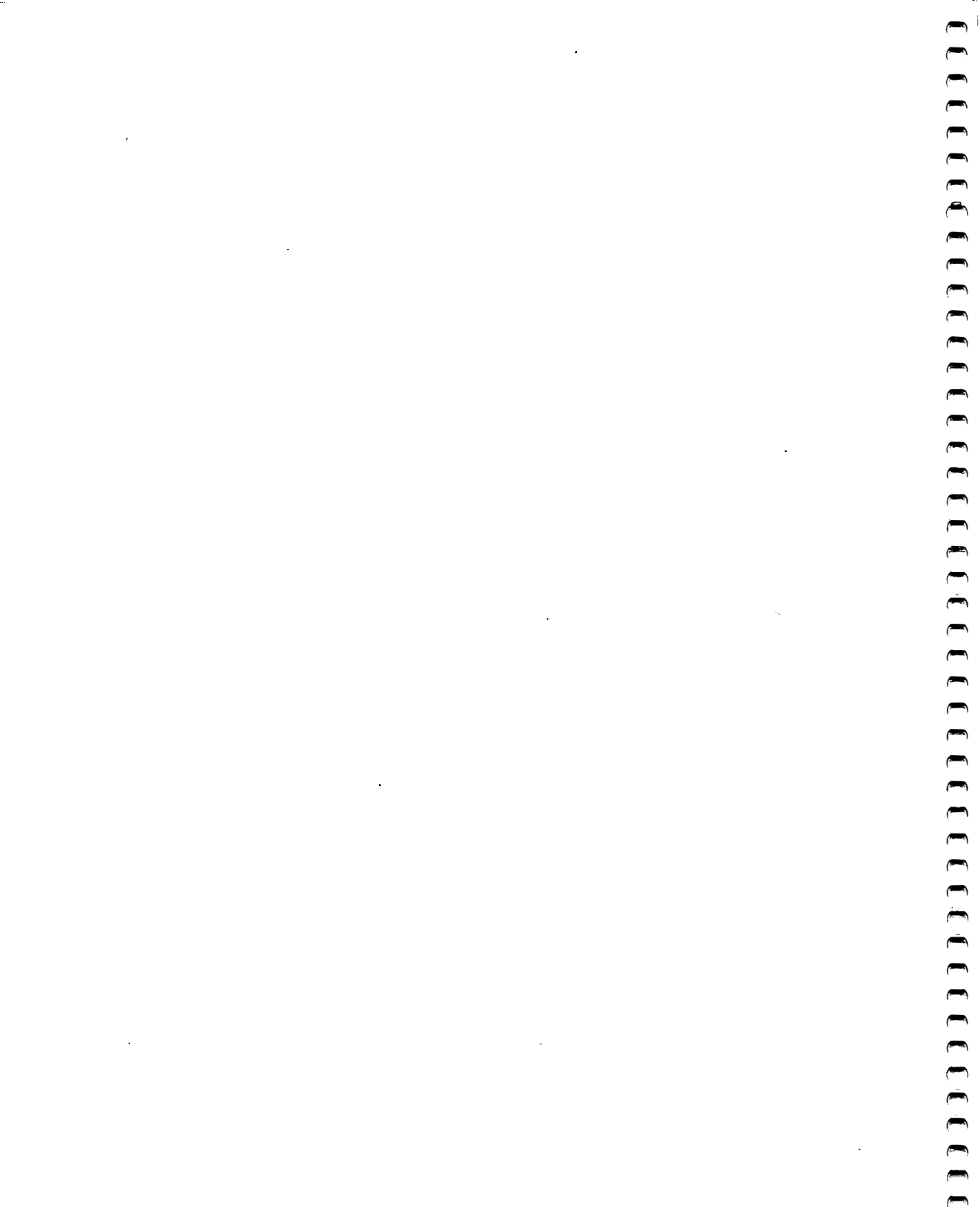
```
tape bot
```

Description: The **tape** utility provides a mechanism for positioning a mounted tape to a specific location in anticipation of being read from or written to by another utility (i.e. **pax**). You can also use **tape** to examine the current status of the tape unit.

Exit status:

0	The command(s) completed successfully.
>0	The utility aborted after encountering the first error.

See also: **cpio, pax, tar**



Syntax:

```
tar -c[bfvw] [filename...]  
tar -r[bvw] [filename...]  
tar -t[fv] [filename]  
tar -u[bvw] [filename]  
tar -x[flmovw] [filename...]
```

- Options**
- b Use the next argument on the command line as the blocking factor for tape records. The default is 1; the maximum is 20. This option should be used only with raw magnetic tape archives. Normally, the block size is determined automatically when reading tapes.
 - c Create a new archive. Writing begins at the beginning of the archive, instead of after the last file.
 - f Use the next argument on the command line as the name of the archive instead of the default, which is usually a tape drive. If the dash character (-) is specified as a filename, **tar** writes to the standard output or reads from the standard input, whichever is appropriate for the options given. Thus, **tar** can be used as the head or tail of a pipeline.
 - l (lowercase letter “e”) Report if all of the links to the files being archived cannot be resolved. If this option isn’t specified, no error messages are written to the standard output. This option is valid only with the -c, -r, and -u options.
 - m Don’t restore modification times. The modification time of the file will be the time of extraction. This option is invalid with the -t option.
 - o Cause extracted files to take on the user and group identifier of the user running the program rather than those on the archive. This option is valid only with the -x option.
 - r Write named files to the end of the archive.
 - t List the names of all of the files in the archive.
 - u Add named files to the archive if they aren’t already there, or if they have been modified since last written into the archive. This implies the -r option.

tar

read and write tape archive files (UNIX)

- w Print the action to be taken, followed by the name of the file, then wait for the user's confirmation. If you enter a word beginning with **y**, the action is performed. Any other input means "no". This option is invalid with the **-t** option.
- x Extract named files from the archive. If a named file matches a directory whose contents had been written onto the archive, that directory is recursively extracted. If a named file in the archive doesn't exist on the system, the file is created with the same mode as the one in the archive, except that the set-user-id and get-group-id modes are set only if you have appropriate privileges.

If the files exist, their modes are not changed except as described above. The owner, group, and modification time are restored if possible. If no *filename* argument is given, the entire contents of the archive is extracted. Note that if several files with the same name are in the archive, the last one will overwrite all earlier ones.

- v Be verbose. Usually, **tar** works silently, but the **-v** option causes it to print the name of each file it processes, preceded by the option letter. With the **-t** option, **-v** gives more information about the archive entries than just the name.

filename

The pathname of the file to be archived.

Examples: Display a verbose listing of the archive members in **dist.tar**:

```
tar -tvf dist.tar
```

Copy the contents of the current directory to the floppy drive:

```
tar -cf /dev/fd0 .
```

Make an archive, **backup.tar**, of all the C source and header files in the current directory:

```
tar -cvf backup.tar *. [ch]
```

- Description:** The **tar** utility reads and writes archive files that conform to the Archive/Interchange File Format specified in IEEE Std. 1003.1-1988.
- Files:** The controlling terminal (**/dev/tty**) is used to prompt the user for information when either or both the **-i** or **-y** options are specified.
- See also:** **cpio**, **pax**



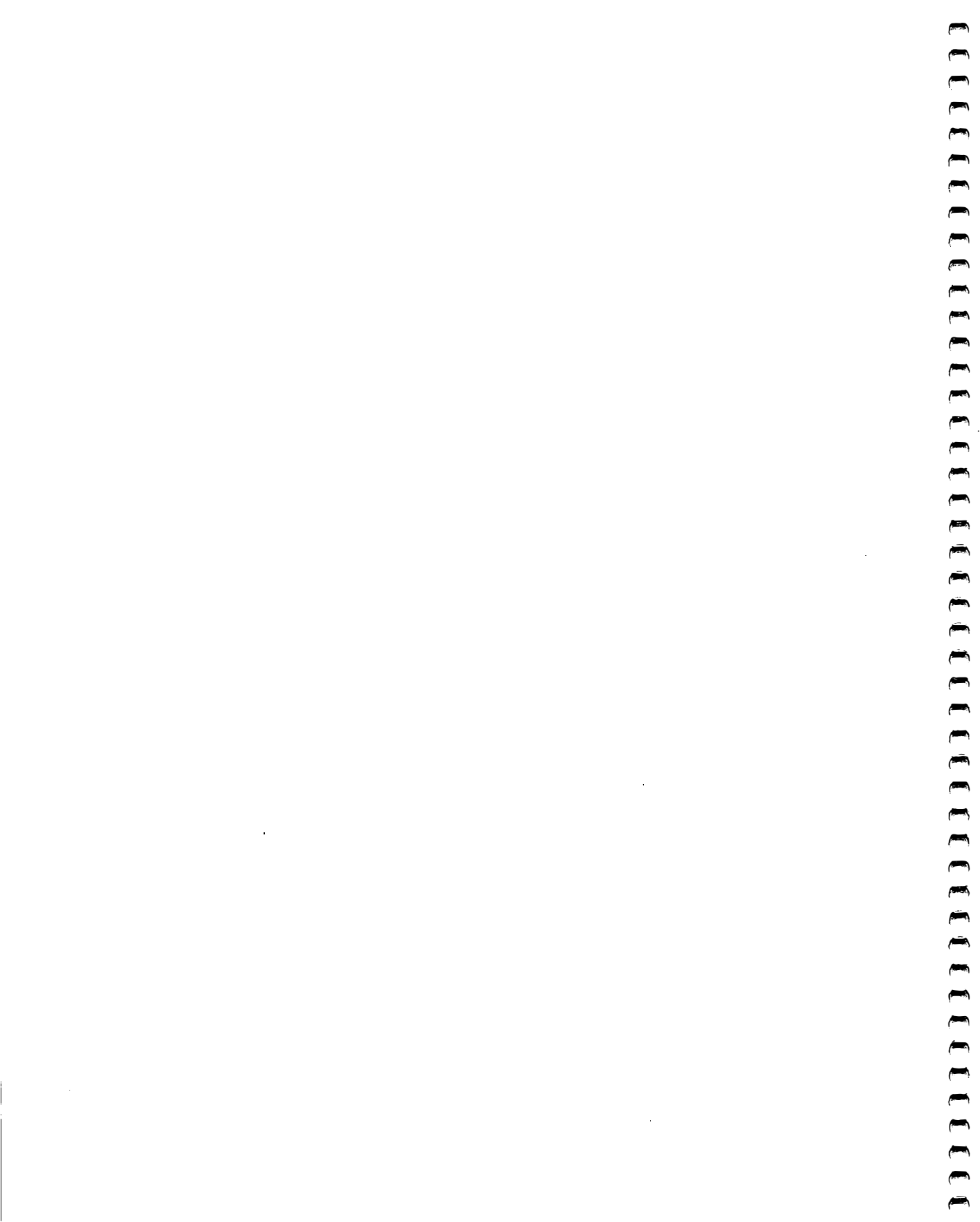
Syntax: `termdef [-c command]`

Description: The `termdef` utility prompts the user for their terminal type. You should find `termdef` useful as a front-end process on dial-up lines.

If the terminal name entered matches one of the supported `terminfo` entries, `termdef` will set the `TERM` environment variable accordingly, and will do an `exec()` into the `login` command.

A list of all available terminal types will be displayed if the user presses <Enter> without specifying a terminal name.

To launch a command other than `login`, use the `-c` option. The specified command will be launched instead.



Syntax: `vol -r|-w [-s blks_num] block_special_device`

Options:

- `-r` Read from the block special device.
- `-w` Write to the block special device.
- `-s blks_num` Skip this number of blocks at the start of each volume (default is 1).

Examples: Use **pax** to archive everything under the current directory and let **vol** split the data across as many diskettes (`/dev/fd0`) as required:

```
pax -w . | vol -w /dev/fd0
```

Restore the archive from `/dev/fd0`:

```
vol -r /dev/fd0 | pax -r
```

Description: You use the **vol** utility primarily with removable media, particularly floppy diskettes. At the beginning of each volume, **vol** writes a header containing the creation time and sequence number of the volume—this helps you detect out-of-sequence diskettes.

A media descriptor is located on the first block of a floppy diskette. For this reason, **vol** by default skips the first block of each volume it writes to. If you don't want it to skip the first block, you can specify the **-s** option with a value of zero.



If you specify **-s 0**, no blocks will be skipped and no header will be written at the beginning of each volume. If you specify the **-s** option when writing an archive, remember that you must also specify the same number of blocks when the archive is retrieved.

If you want to save files larger than a single volume, you specify the **-w** option. The **vol** utility will read the standard input and write it onto a block special device (usually diskettes), pausing at the start of each one.

If you want to reassemble and extract archives split across several volumes, you specify the **-r** option. The **vol** utility will read from a block special device and write the output in a single stream to standard output.

vol

manage multi-volume archives (QNX)

See also: `cpio`, `pax`, `tar`

Caveats: If you tell the `vol` utility to skip zero blocks on a floppy (`-s 0`), it might not be able to determine the media characteristics on retrieval.

The `vol` utility does *not* format media.

Appendix A

Time Zone Rules

This appendix describes the various rules used for establishing time zones and time changes (e.g. daylight savings time). You'll find a time zone chart* and a map at the end of this appendix.

You specify time zone information via the **TZ** environment variable. This variable contains a coded string that lets you specify the time zone, as well as the rules for changing to and from daylight savings time (if used in your area). The value of **TZ** has the following form (spaces have been inserted for clarity):

std offset dst offset, rule

where:

<i>std</i>	standard time
<i>offset</i>	offset of standard time from UTC time
<i>dst</i>	daylight savings time
<i>offset</i>	offset of daylight savings time from UTC time
<i>rule</i>	specifies when to change between standard time and daylight savings time.

In more detail, **TZ** is coded as follows:

TZ=std[offset[dst[offset] [, start[/time] , end[/time]]]]

* The time zone chart was excerpted from *OAG® Desktop Flight Guide-Worldwide Edition* (May 1991, Vol. 16, No. 3 pp. 1278-1279), published by Official Airline Guides, Inc., Oak Brook, Illinois.

The components of the **TZ** environment variable string are defined as follows.

std and dst Defines a string used to identify the time zone (e.g. **EST**). All characters are allowed, except for:

- a leading colon (:)
- digits
- a comma (,)
- a minus sign (-)
- a plus sign (+),
- ASCII NUL

std defines the name of the standard time zone and *dst* defines the name of the daylight savings time zone. If *dst* is omitted, then daylight savings time isn't used in this locale.

offset Indicates the value you must add to the local time to arrive at UTC. The form for *offset* is:

HH [: *MM* [: *SS*]]

The minutes *MM* and seconds *SS* are optional. The hour *HH* is required and can be a single digit. You must specify an hour between 0 and 24. You can specify minutes and seconds, if required, between 0 and 59. Time zones east of the Prime Meridian (through England) should be preceded by a minus (-); time zones west of the Prime Meridian can be optionally preceded by a plus (+). An *offset* following *std* is required, but the *offset* following *dst* may be omitted. In this case, daylight savings time is assumed to be one hour ahead of standard time.

rule Specifies when to change from standard time to daylight savings time and back again. If you don't specify *rule*, QNX assumes standard time is in effect. The *rule* has the following format:

date/time, date/time

where the first *date* specifies when the change to daylight savings time takes place and the second *date* specifies when the change to standard time takes place. Each *time* field specifies the exact time at which the change shall be made. The *date* field has one of the following formats:

- Jn*** Specifies the date as a Julian day where $1 \leq n \leq 365$. This format doesn't allow leap days to be referenced (e.g. February 28 is day 59 and March 1 is day 60; it's impossible to refer to February 29).
 - n*** This specifies the date as a zero-based Julian day where $0 \leq n \leq 365$. Leap days are counted and you can refer to February 29.
 - Mm.w.d*** This specifies the *d*th day ($0 \leq d \leq 6$) of week *w* of month *m* of the year ($1 \leq w \leq 5$, $1 \leq m \leq 12$, where week 5 means "the last *d* day in month *m* that may occur in either the fourth or fifth week"). Week 1 is the first week of month *m* in which the *d*th day occurs. Day zero is Sunday.
- The time has the same format as *offset* except that no leading sign (+ or -) is allowed. The default, if *time* isn't specified, is 02:00:00.

International Time Chart

International Time Chart

Standard Time: Legal time for each country fixed by law and based on the theoretical division of the world's surface into 24 zones each of 15° longitude with certain deviations due to frontiers or local option.
Daylight Saving Time (DST): Modified (advanced) legal time adopted by certain countries for part of year, especially during local summer.

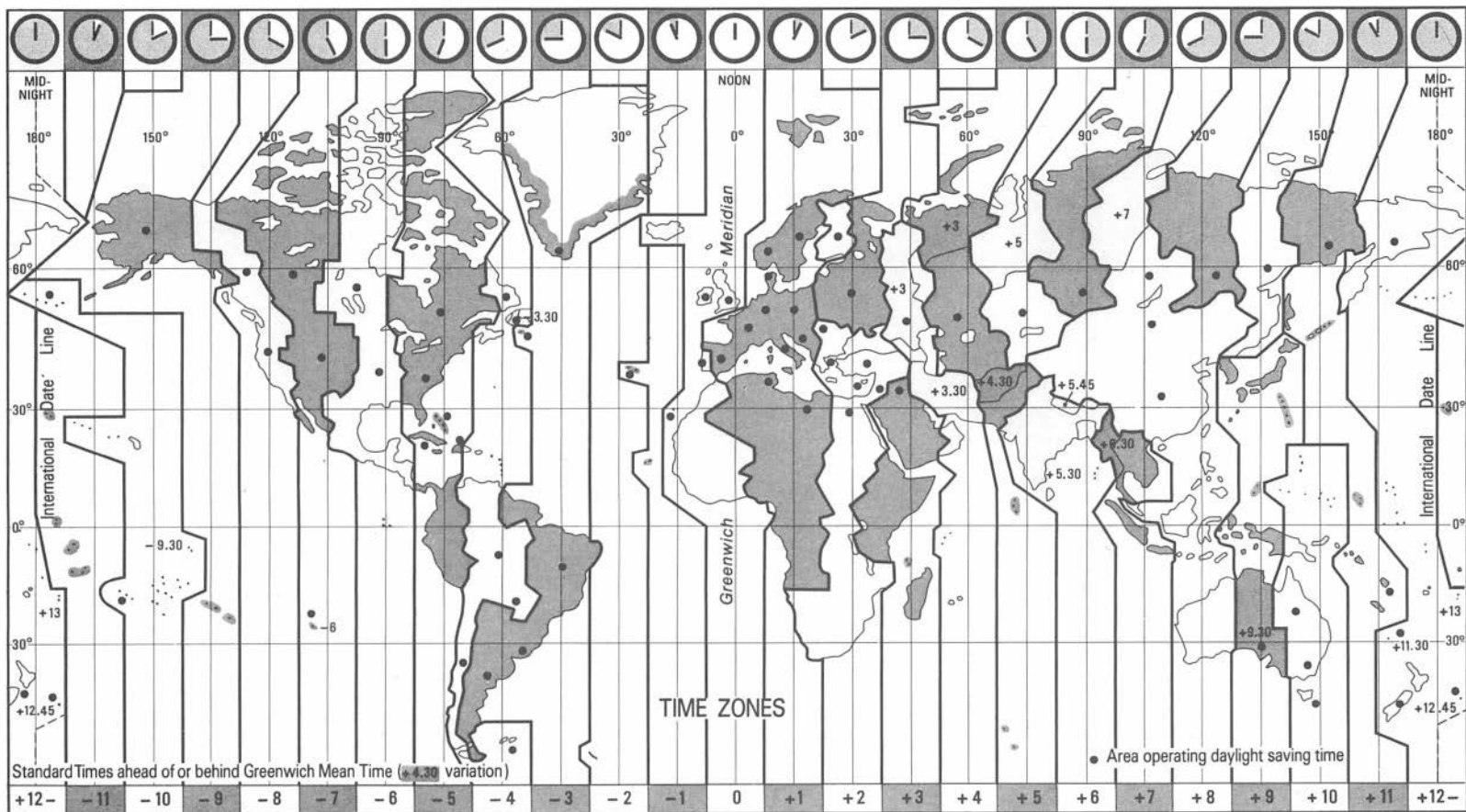
△ — Arizona and parts of Indiana do not observe DST.
 † — Province of Saskatchewan and certain Canadian cities remain on Standard Time all year.
 * — Except Broken Hill, N.S.W. follows South Australia times.
 ■ — Amapa, Amazonas, Para, Rondônia, Roraima, States of Brazil do not observe DST.
 E — Estimated Date Based On Previous Year.

Country	Standard Time		Daylight Saving Time		Country	Standard Time		Daylight Saving Time	
	Hours from GMT	Time at 1200 hrs GMT	Hours from GMT	Effective Period (first and last day)		Hours from GMT	Time at 1200 hrs GMT	Hours from GMT	Effective Period (first and last day)
Algeria	+1	13:00	+2	Mar. 31—Sep. 28, 1991	Lesotho	+2	14:00		
Algeria	+1	13:00			Liberia	GMT	12:00		
American Samoa	-11	01:00			Libyan Arab Jamahiriya	+2	14:00		
Andorra	+1	13:00	+2	Mar. 31—Sep. 28, 1991	Luxembourg	+1	13:00	+2	Mar. 31—Sep. 28, 1991
Angola	+1	13:00			Madagascar	+3	15:00		
Argentina	-3	09:00			Malawi	+2	14:00		
Aruba	-4	08:00			Malaysia	+8	20:00		
Australia					Maldives	+5	17:00		
Lord Howe Is.	+10 1/2	22:30			Mali	GMT	12:00		
New South Wales*	+10	22:00			Malta	+1	13:00	+2	Mar. 31—Sep. 28, 1991
Northern Territory	+9 1/2	21:30			Martinique	-4	08:00		
Queensland	+10	22:00			Mauritania	GMT	12:00		
South Australia and Broken Hill	+9 1/2	21:30			Mauritius	+4	16:00		
Tasmania	+10	22:00			Mexico				
Victoria, Australian Capital Territory (A.C.T.)	+10	22:00			Baja California Sur and N. Pacific Coast	-7	05:00		
Western Australia	+8	20:00			States of Sinaloa and Sonora				
Austria	+1	13:00	+2	Mar. 31—Sep. 28, 1991	Baja California Norte (Above 28th Parallel)	-8	04:00		
Bahamas (including Turks and Caicos Islands)	+5	07:00	-4	Apr. 7—Oct. 26, 1991	States of Durango, Coahuila, Nuevo Leon, Tamaulipas	-6	06:00		
Bahrain	+3	15:00			General Mexico, States of Campeche, Quintana Roo and Yucatan	-6	06:00		
Bangladesh	+6	18:00			Cities of Ensenada, Mexicali, San Felipe,	-8	04:00	-7	Apr. 7—Oct. 26, 1991
Barbados	-4	08:00			Tijuana				
Belgium	+1	13:00	+2	Mar. 31—Sep. 28, 1991	Midway Island	-11	01:00		
Belize	+6	06:00			Monaco	+1	13:00	+2	Mar. 30E—Sep. 29, 1991E
Benin Peoples Rep. (Dahomey)	+1	13:00			Mongolia (Ulan Bator)	+8	20:00		
Bermuda	-4	08:00	-3	Apr. 7—Oct. 26, 1991	Morocco	GMT	12:00		
Bhutan	+6	18:00			Mozambique	+2	14:00		
Bolivia	-4	08:00			Myanmar	+6 1/2	18:30		
Botswana	+2	14:00			Namibia	+2	14:00		
Brazil ■ — East (Including All Coast and Brasília)	-3	09:00			Nauru, Republic Of	+12	23:59		
West	-4	08:00			Netherlands	+1	13:00	+2	Mar. 31—Sep. 28, 1991
Territory of Acre	-5	07:00			Netherlands Antilles (incl. Southern St. Maarten)	-4	08:00		
Fernando De Noronha	-2	10:00			New Caledonia	+11	23:00		
British Virgin Islands	-4	08:00			New Zealand (Excluding Chatham Is.)	+12	23:59		
Brunei Darussalam	+8	20:00			Chatham Is.	+12 3/4	00:00		
Bulgaria	+2	14:00	+3	Mar. 31—Sep. 28, 1991	Nicaragua	-6	06:00		
Burkina Faso	GMT	12:00			Niger	+1	13:00		
Burundi	+2	14:00			Nigeria	+1	13:00		
Cambodia	+7	19:00			Niue Island	-11	01:00		
Cameroon, Republic Of	+1	13:00			Norfolk Island	+11 1/2	23:30		
Canada † — Newfoundland (Island Of)	-3 1/2	08:30	-2 1/2	Apr. 7—Oct. 26, 1991	Norway	+1	13:00	+2	Mar. 31—Sep. 28, 1991
Atlantic	-4	08:00	-3	Apr. 7—Oct. 26, 1991	Osman	+4	16:00		
Eastern	-5	07:00	-4	Apr. 7—Oct. 26, 1991	Pacific Islands Trust Territory				
Central	-6	06:00	-5	Apr. 7—Oct. 26, 1991	Caroline Is. (Excluding Pohnpei Is., Kusaie, and Pingelap)	+11	22:00		
Mountain	-7	05:00	-6	Apr. 7—Oct. 26, 1991	Kusaie, Pingelap, Marshall Is. (Excluding Kwajalein)	+12	23:59		
Pacific	-8	04:00	-7	Apr. 7—Oct. 26, 1991	Kwajalein	-12	00:01		
Yukon Territory	-8	04:00	-7	Apr. 7—Oct. 26, 1991	Mariana Island (Excluding Guam)	+10	22:00		
Cape Verde Islands	-1	11:00	-7	Apr. 7—Oct. 26, 1991	Palau Island	+9	21:00		
Cayman Islands	-5	07:00			Ponape Island	+11	23:00		
Central African Republic	+1	13:00			Pakistan	+5	17:00		
Chad	+1	13:00			Panama	-5	07:00		
Chile					Papua New Guinea (And Bougainville Is.)	+10	22:00		
Continental	-4	08:00			Paraguay	-4	08:00		
East Island	-6	06:00			Peru	-5	07:00		
China, People's Republic Of	+8	20:00			Philippines, Republic Of	+8	20:00		
Cocos (Keeling) Islands	+6 1/2	18:30	+9	Apr. 14—Sep. 14, 1991	Poland	+1	13:00	+2	Mar. 31—Sep. 28, 1991
Colombia	-5	07:00			Portugal				
Comoros	+3	15:00			Azores	-1	11:00	GMT	Mar. 31—Sep. 28, 1991
Congo	+1	13:00			Madara	GMT	12:00	+1	Mar. 31—Sep. 28, 1991
Cook Islands	-10	02:00			Marinland	GMT	12:00	+1	Mar. 31—Sep. 28, 1991
Costa Rica	-6	06:00			Puerto Rico	-4	08:00		
Cote D'Ivoire	GMT	12:00			Qatar	+3	15:00		
Cuba	-5	07:00	-4	Apr. 7—Oct. 12, 1991	Reunion	+4	16:00		
Cyprus	+2	14:00	+3	Mar. 31—Sep. 28, 1991	Romania	+2	14:00	+3	Mar. 31—Sep. 28, 1991
Czechoslovakia	+1	13:00	+2	Mar. 31—Sep. 28, 1991	Rwanda	+2	14:00		
Denmark	+1	13:00	+2	Mar. 31—Sep. 28, 1991	Saint Pierre and Miquelon	-3	09:00	-2	Apr. 7—Oct. 26, 1991
Djibouti	+3	15:00			Saint Vincent and the Grenadines	-4	08:00		
Dominican Republic	-4	08:00			Samoa	+1	01:00		
Ecuador					San Marino	+1	13:00		
Continental	-5	07:00			Sao Tome Island and Principe Is.	GMT	12:00		
Galapagos Is.	-6	06:00			Saudi Arabia	+3	15:00		
Egypt	+2	14:00	+3	May 1—Sep. 30, 1991	Senegal	GMT	12:00		
El Salvador	-6	06:00			Seychelles	+4	16:00		
Equatorial Guinea	+1	13:00			Sierra Leone	GMT	12:00		
Ethiopia	+3	15:00			Singapore	+8	20:00		
Falkland Islands	-4	08:00	-3	Sep. 9, 1990E—Apr. 20, 1991E	Solomon Islands (Excluding Bougainville Is.)	+11	23:00		
Port Stanley	GMT	12:00	+1	Mar. 31—Sep. 28, 1991	Somalia	+3	15:00		
Faroe Island	+12	23:59	+3	Mar. 31—Sep. 28, 1991	South Africa	+2	14:00		
Fiji	+12	23:59	+2	Mar. 31—Sep. 28, 1991	Spain				
Finland	+1	13:00	+2	Mar. 31—Sep. 28, 1991	Canary Is.	GMT	12:00	+1	Mar. 31—Sep. 28, 1991
France	+1	13:00			Continental, Balearic and Mallorca Islands	+1	13:00	+2	Mar. 31—Sep. 28, 1991
French Guiana	-3	09:00			Melilla	+1	13:00		
French Polynesia					Sri Lanka	+5 1/2	17:30		
Gambier Is.	-9	03:00			St. Helena	GMT	12:00		
Marquesas Is.	-9 1/2	02:30			Sudan	+2	14:00		
Society Is., Tubuai Is., Tuamotu Is., Tahiti	-10	02:00			Suriname, Republic Of	-3	09:00		
Gabon	+1	13:00			Swaziland	+2	14:00		
Gambia	GMT	12:00			Sweden	+1	13:00	+2	Mar. 31—Sep. 28, 1991
Germany, Federal Rep. Of	+1	13:00	+2	Mar. 31—Sep. 28, 1991	Switzerland	+1	13:00	+3	Apr. 1—Sep. 29, 1991
Ghana	GMT	12:00			Syria	+2	14:00		
Gibraltar	+1	13:00	+2	Mar. 31—Sep. 28, 1991	Taiwan	+8	20:00		
Greece	+2	14:00	+3	Mar. 31—Sep. 28, 1991	Tanzania	+3	15:00		
Greenland					Thailand	+7	19:00		
Angmagssalik and W. Coast	-3	09:00	-2	Mar. 30—Sep. 28, 1991	Togo	GMT	12:00		
East Greenland	GMT	12:00	+1	Mar. 30—Sep. 28, 1991	Tonga	+13	01:00		
Scoresbyund	-1	11:00	GMT	Mar. 30—Sep. 28, 1991	Trinidad and Tobago	-4	08:00		
Thule	-4	08:00	-3	Mar. 30—Sep. 28, 1991	Tunisia	+1	13:00		
Guadeloupe (incl. St. Barthelemy, Northern St. Martin)	-4	08:00			Turkey	+2	14:00	+3	Mar. 31—Sep. 28, 1991
Guam	+10	22:00							
Guatemala	-6	06:00	-5	Mar. 23—Sep. 6, 1991					
Guinea	GMT	12:00							

Standard Time: Legal time for each country fixed by law and based on the theoretical division of the world's surface into 24 zones each of 15° longitude with certain deviations due to frontiers or local option.
Daylight Saving Time (DST): Modified (advanced) legal time adopted by certain countries for part of year, especially during local summer.

△ — Arizona and parts of Indiana do not observe DST.
‡ — Province of Saskatchewan and certain Canadian cities remain on Standard Time all year.
* — Except Broken Hill, N.S.W., follows South Australia times.
■ — Amapá, Amazonas, Pará, Rondônia, Roraima, States of Brazil do not observe DST.
E — Estimated Date Based On Previous Year.

Country	Standard Time		Daylight Saving Time		Country	Standard Time		Daylight Saving Time	
	Hours from GMT	Time at 1200 hrs GMT	Hours from GMT	Effective Period (first and last day)		Hours from GMT	Time at 1200 hours GMT	Hours from GMT	Effective Period (first and last day)
Guinea-Bissau	GMT	12 00			Turks and Caicos Is.	-5	07 00	-4	Apr. 7—Oct. 26, 1991
Guyana	-3	09 00			Tuvalu	+12	23 59		
Haiti	-5	07 00	-4	Apr. 7—Oct. 26, 1991	Uganda	+3	15 00		
Honduras	-6	06 00			United Arab Emirates (Abu Dhabi, Dubai, Sharjah, Ras Al Khaimah)	+4	16 00		
Hong Kong	+8	20 00			United Kingdom	GMT	12 00	+1	Mar. 31—Oct. 26, 1991
Hungary	+1	13 00	+2	Mar. 31—Sep. 28, 1991	U.S.A.—Eastern Time ^Δ	-5	07 00	-4	Apr. 7—Oct. 26, 1991
Iceland	GMT	12 00			Central Time	-6	06 00	-5	Apr. 7—Oct. 26, 1991
India (incl. Andaman Is.)	+5 1/2	17 30			Mountain Time ^Δ	-7	05 00	-6	Apr. 7—Oct. 26, 1991
Indonesia					Pacific Time	-8	04 00	-7	Apr. 7—Oct. 26, 1991
Central	+8	20 00			Alaska (except Aleutian Islands W. 169.30 Deg.)	-9	03 00	-8	Apr. 7—Oct. 26, 1991
East	+9	21 00			Aleutian Islands 169.30 long. W. to Western tip	-10	02 00	-9	Apr. 7—Oct. 26, 1991
West (Jakarta)	+7	19 00			Arizono ^Δ	-7	05 00		
Iran (The Islamic Rep. of)	+3 1/2	15 30			Hawaiian Islands	-10	02 00		
Iraq	+3	15 00			Indiana (East) ^Δ	-5	07 00		
Ireland, Rep. of	GMT	12 00	+1	Mar. 31—Oct. 28, 1991	U.S. Virgin Islands	-4	08 00		
Israel	+2	14 00	+3	Mar. 24—Aug. 31, 1991	Uruguay	-3	09 00		
Italy	+1	13 00	+2	Mar. 31—Sep. 28, 1991	U.S.S.R.—(Moscow time)	+2	15 00	+3	Mar. 31—Sep. 28, 1991
Jamaica	-5	07 00			Approximately every 15° from 30° 30' E. to 172° 30' W. add 1 hr.	+12	23 59		
Japan	+9	21 00			Vanuatu	+11	23 00	+12	Sep. 29, 1991—Mar. 28, 1992
Johnston Is.	-10	02 00			Venezuela	-4	08 00		
Jordan	+2	14 00	+3	Apr. 19—Sep. 26, 1991	Vietnam, Socialist Rep. of	+7	19 00		
Kampuchea, Dem. (see Cambodia)					Wake Is.	+12	23 59		
Kenya	+3	15 00			Wallis and Futuna Is.	+12	23 59		
Kiribati, Rep. of	+12	23 59			Windward Islands	-4	08 00		
Canton, Enderbury Islands	-11	01 00			Grenada, St. Lucia	+3	15 00	+2	Mar. 31—Sep. 28, 1991
Christmas Is.	-10	02 00			Yemen, Republic of	+1	13 00		
Korea, Democratic People's Rep. of	+9	21 00			Yugoslavia	+1	13 00		
Korea, Republic of	+9	21 00			Zaire	+1	13 00		
Kuwait	+3	15 00			Kinshasa Mbandaka	+1	13 00		
Laos	+7	19 00			Kasai, Kivu, Haut-Zaire, Shaba	+2	14 00		
Lebanon	+2	14 00	+3	May 1—Oct. 15, 1991	Zambia	+2	14 00		
Leeward Islands					Zimbabwe	+2	14 00		
Antigua, Dominica, Montserrat, St. Christopher, St. Kitts, Nevis, Anguilla	-4	08 00							



A

access control utilities

login 20

newgrp 21

passwd 20

su 20

access fields 22

accessing resources 20

accounting file 25 – 27

 clearing 27

 compressing 27

 creating 25

 discarding information 25

 format 25

 interpreting 25

 typical example 26

accounts

 adding 20

 creating without password,
 home directory, shell, or group 21

 deleting 21

 restriction on who can add to system 20

 setting up 19 – 27

adapter cards 32

see also serial adapters

altsysinit file 9

see also system initialization file

archive utilities

cpio 71

pax 71

tar 71

vol 71

archives 71

 compressing data 74

 examples 75 – 76

see also backups

B

backup formats 70 – 71

 archive backups 71

 filesystem backups 72

backup media 72 – 73

 fixed disks 74

 floppies 72

 removable disks 74

 tape 73

backup utilities

see archive utilities

backups

 compressing data 74

 when to back up 70

see also archives

- baud rate
 - specifying 36
- bitmap blocks 88
- blocks
 - bad block, recovering from
 - middle of file 97
 - examining 97
 - extent blocks 91
- booting
 - boot server, selecting 5
 - diskless workstation 5, 82
 - from floppy 4
 - from hard disk 4 – 7, 10
 - from hard disk, processes required 80
 - initial boot 8
 - netboot configuration file 6
 - over a network 5 – 7
 - system doesn't boot 97 – 100
 - without creating a system
 - initialization file 7
- bugs, reporting 106
- build file, constructing 78 – 83
- buildqnx** utility 78

C

- cancelling print jobs 47
- chkfsys** utility 93
 - Rambo mode 96
 - using 94
 - using in a live system 95
 - when to run 95
- clock, starting 13
- communication sessions 34
- compressed data
 - recovering 74
- compression utilities 74
- conferencing system 107
 - see* QUICS

- connecting serial devices 36
 - see also* serial devices
- console driver
 - starting 10, 30
- consoles, starting login 11
- coordinated universal time
 - see* time zones, UTC
- cpio** utility 71
- cron** server
 - selecting a machine as 12
- current process
 - changing group ID of 21
- custom operating system images
 - see* images

D

- data
 - bits 35
 - compression 74
 - slowing down transmission of 35
- date
 - establishing 14
 - see also* time zones
- daylight savings time
 - changing to or from 201 – 202
 - time zone, defining 202
 - see also* time zones
- deleted files, recovering 96
- dequeues 48
 - file where defined 50
 - serving the same queue 49
 - see also* print spooling
- device drivers
 - Dev.con** 30
 - Dev.par** 30
 - Dev.ser** 30 – 31
 - starting 30
 - see also* system initialization file

- Device Manager
 - starting 10, 30
 - devices
 - see* parallel devices
 - see also* serial devices
 - dinit** utility 93
 - directories
 - access fields 22
 - directory entries 89
 - links 90
 - lost+found** directory 102
 - recovering if lost 101 – 103
 - removing without returning
 - used blocks 94
 - root directory 88
 - structure 89
 - structure, checking 93
 - disk images 80
 - see also* images
 - disk partitions 87
 - see also* partitions
 - diskless workstations
 - booting over the network 82
 - disks
 - backing up to or restoring from 72, 74
 - bad blocks, recovering 97
 - browsing 94
 - maintenance utilities 93
 - patching minor problems 94
 - recovering 85 – 103
 - recovery procedures 94 – 96
 - structure 86 – 92
 - DOS
 - attribute bits 65
 - disks, accessing via QNX programs
 - and utilities 61
 - filesystems, setting up 61 – 67
 - invalid filename characters 65
 - invalid filenames 65
 - partition types 64
 - text files, structure of 64
 - Dosfsys** filesystem manager 61
 - actions performed when started 62
 - binary files 64
 - DOS to QNX permission-bit translations 66
 - DOS version support 63
 - DOS/QNX permission mapping 65
 - error codes 66
 - file ownership 66
 - invocation modes 61
 - name adoption 62
 - QNX to DOS attribute-bit translations 66
 - QNX to DOS character and name mapping 65
 - drivers
 - see* device drivers
 - DSR 34
 - DTR lines 34
- ## E
- electronic mail
 - sending to Quantum employees
 - or other users 106
 - see also* QUICS
 - embedded images 83
 - emu87** utility 11
 - environment variables, defining 12
 - /etc/acclog** file
 - see* accounting file
 - executable files
 - executing with privileges of owner or group 22
 - extent blocks 91
- ## F
- fdisk** utility 93
 - file maintenance utilities 93
 - chkfsys** 93 – 95

- dinit** 93
- fdisk** 93
- spatch** 94, 97, 102
- zap** 94, 96
- file permissions 22
- filenames
 - DOS 65
 - longer than 16 characters 90
- files
 - access fields 22
 - accounting file 25 – 27
 - blocks, examining 97
 - deleted 96
 - DOS text files 64
 - executable files 22
 - extent blocks 91
 - links 90
 - maintenance utilities 93
 - multiple extents 91
 - permissions 22
 - recovering 85 – 103
 - recovering if deleted 96
 - recovering if lost 101 – 103
 - removing without returning used blocks 94
 - structure 92
- filesystems
 - backups 72
 - how data is stored on disk 86
 - restoring 94
 - setting up second system 57 – 67
 - see also* DOS filesystems; partitions
- fixed disks, backing up to or restoring from 74
 - see also* backups
- floating point emulator 11
- floppies, backing up to or restoring from 72
 - see also* backups
- floppy driver, starting 12
- free software 106
- freeze** utility 75

G

- global name server
 - selecting a machine as 12
- greenwich mean time
 - see* time zones, UTC
- group IDs 21

H

- hard disk booting
 - see* booting
- hard disks
 - see* disks
- hardware adapters 31
- hardware interrupt generated by each UART 31

I

- images
 - build file, constructing 78 – 83
 - buildqnx** utility 78
 - creating through **buildqnx** or the Makefile 78
 - disk images 80
 - embedded images 83
 - heap size, setting 79
 - image file location 78
 - local memory, allocating 79
 - network images 82
 - processes required for hard disk booting 80
 - processes, selecting 79
- installing
 - on a hard disk 4
- interrupt generated by each UART 31

L

- licensing 15 – 17
 - application licensing 16
 - expanding your license 17
 - license utility 17
 - operating system licensing 15
- links 90
- loader block 87
- login** utility 20
- login, automated 40
- login, starting on consoles 11
- lost+found** directory 102
- lp** utility 47
- lpc** utility 48
- lpq** utility 47
- lprm** utility 47
- lpsrvr** utility 48 – 55

M

- mail
 - see* QUICS, email
- Makefile
 - examples 80
 - location 80
 - macros for changing defaults 80
- melt** utility 75
- modem** utility 40
- modems
 - configuring 40
 - connecting 36 – 37
 - detecting incoming call 35
 - dial-up access 40
 - establishing connection with
 - another modem 35
 - flow control between terminals
 - and modems 35
 - hardware handshaking 37

- high-speed, error-correcting 36
- standard 37
- terminal type, querying user for 42

N

- name server
 - see* global name server
- network
 - sharing devices 46
- network booting 5 – 7
 - netboot configuration file 6
- network card
 - removing, effects of 10
- network images 82
- node numbers
 - changing, effects of 10

O

- operating system images
 - see* images
- output buffers, specifying 31

P

- parallel devices 30
 - multiple parallel ports 30
 - output buffers, specifying 31
 - single parallel port 30
- parallel driver
 - starting 11, 30
- parity bits 35
- partitioning pathname space 58 – 60
- partitions
 - bitmap blocks 88
 - directory structure, checking 93

- key components 87
- loader block 87
- multiple partitions on single drive 59
- root block 87
- root directory 88
- passwd** utility 20
- password database 20
 - access to files 23
 - default password files 25
 - /etc/group** file 24
 - /etc/passwd** file 23
 - /etc/shadow** file 24
 - list of files 23
 - users modifying their own entry 22
- passwords
 - changing 20
 - characters required in 20
 - entering 20
- patching minor disk problems 94
- pathname space
 - partitioning 58 – 60
- pax** utility 12, 71
- permissions 22
- phoning Quantum 108
- print jobs
 - cancelling 47
 - querying 47
 - submitting 47
- print queues, controlling 48
- print spooling 46 – 47
 - configuration file 50
 - configuration file (example) 52, 54
 - configuring spooler 48 – 55
 - flexibility 52
 - flow of data within spooler 49
 - lp** utility 47
 - lpc** utility 48
 - lpq** utility 47
 - lprm** utility 47
 - lpsrvr** utility 48 – 55

- server 46
- printers
 - serial printers, connecting 37
 - see also* serial devices
- privileges
 - giving to another user 20
 - non-privileged users performing privileged functions 22
- processes
 - group ID, changing 21

Q

- Quantum update system
 - see* QUICS
- Quantum, phoning
 - see* QUICS
- querying print jobs 47
- queues 48
 - feeding a single dequeue 49, 52
 - file where defined 50
 - see also* print spooling
- QUICS
 - conferencing system 107
 - email 106
 - free software 106
 - hardware 107
 - logging in 109
 - phoning Quantum 108
 - refresh policy 106
 - reporting bugs 106
 - using 110

R

- Rambo mode (**chkfsys**) 95
- raw disks
 - browsing 94

- realtime clock
 - getting time from 10
 - see also* time zones
- recovering
 - deleted files 96
 - disks 85 – 103
 - files 85 – 103
 - lost file and directories 101 – 103
 - lost+found** directory 102
 - see also* file maintenance utilities
- refresh policy 106
- removable disks
 - backing up to or restoring from 74
 - see also* disks
- resources
 - accessing 20
 - controlling access to 21
 - distributing throughout LAN 46
 - restricting access to a group 22
- root block 87
- root directory
 - contents 88
- RS-232
 - cabling assignments 33
 - flow control between terminal
 - and modem 35
 - serial protocol 34
 - session control 34
 - signal names 33
 - transmitted characters 34
 - see also* serial devices
- S**
- security 20 – 24
- serial adapters
 - multi-port 32
 - single-port 32
 - typical installation 32
- serial devices 31 – 43
 - connecting 36
 - hardware adapters 31
 - multi-port serial adapters 32
 - printers, connecting 37
 - RS-232 serial protocol 33
 - serial ports, configuring 35
 - troubleshooting 42
 - see also* modems; RS-232; terminals
- serial driver
 - starting 11, 30
- serial ports
 - baud rate, specifying 36
 - configuring 35
 - data bits 35
 - parity bits 35
 - stop bits 35
- setgid 22
- setting up user accounts 19 – 27
- setuid 22
- shared resources 46
- sinit** utility 8
 - requirements for running 8
- software refresh policy 106
- software, free 106
- spatch** utility 94, 97, 102
- spool configuration file
 - see* print spooling
- standard time
 - see also* time zones
- standard time zone, defining 202
- stop bits 35
- su** utility 20
- submitting print jobs 47
 - see also* print spooling
- system initialization file 8 – 12
 - altsysinit** 9
 - base-level services 10
 - clock, starting 13
 - console driver, starting 10

- cron** server, selecting machine as 12
- Device Manager, starting 10
- DOS filesystem, starting 12
- environment variables, defining 12
- floating point emulator 11
- floppy driver, starting 12
- global name server,
 - selecting machine as 12
- last command in file 11
- login, starting on consoles 11
- machines booting from hard disk 10
- optional services 12
- parallel driver, starting 11
- realtime clock, getting time from 10
- serial driver, starting 11
- single file affecting
 - group of machines 11
- sysinit** 10
- sysinit.node** 9
- time zone, defining 10

T

tape

- backing up to or restoring from 73

tape utility 73

tar utility 71

termdef utility 42

terminals

- automated login 40
- capabilities 41
- configuring for users 39
- connecting 39
- defining type 42
- ending communication 34
- flow control between modems
 - and terminals 35
- hardwired terminals 42
- launching custom applications 40

- querying for type 42

- terminfo database 41

- see also* serial devices

terminfo database 41

time zones 13 – 14

- coordinated universal time (UTC) 13

- date and time, establishing 14

- daylight savings time,

- changing to or from 201 – 202

- daylight savings zone, defining 202

- defining 10

- establishing 13

- not setting, effects of 13

- standard zone, defining 202

- time, establishing 14

- TZ environment variable 201 – 202

- UTC 13

tinit utility 41

transmission of data, slowing down 35

troubleshooting serial device problems 42

TZ environment variable 201

- contents of 202

U

UARTs 31

- hardware interrupts generated by 31

update system

- see* QUICS

user accounts

- see* accounts

user IDs 21

username

- entering 20

UTC

- see* time zones

utilities, finding information about users 24

V

vol utility 12, 71

Z

zap utility 94 – 96

