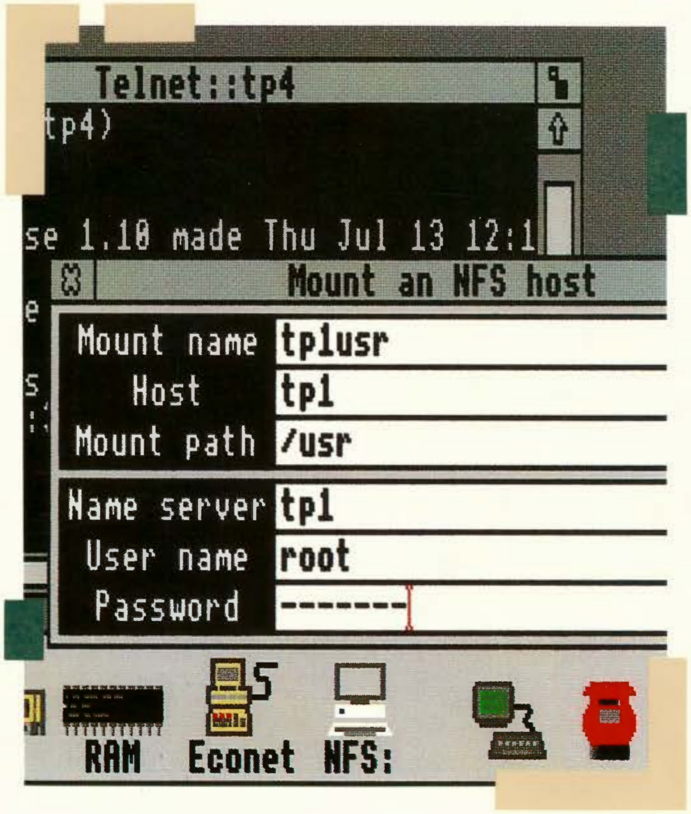
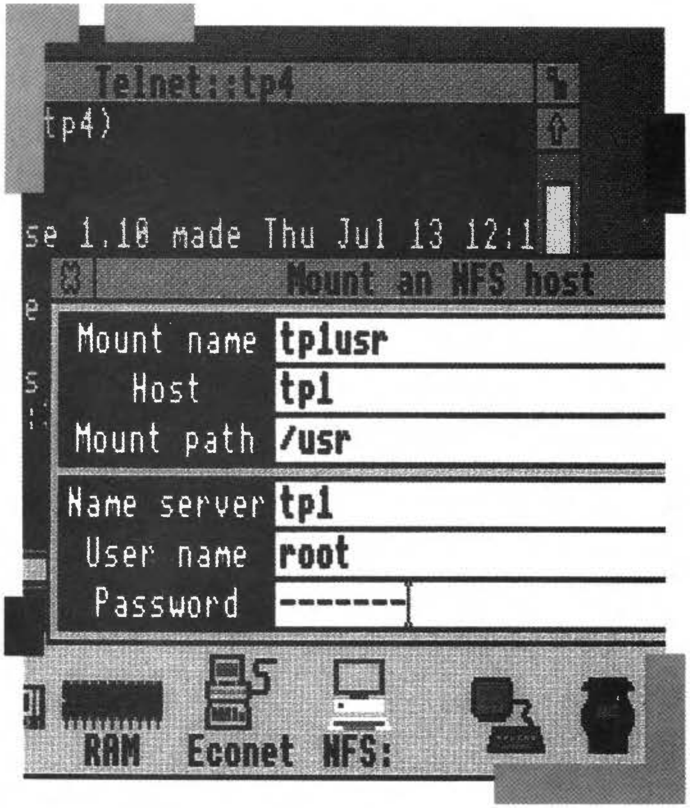
TCP/IP PROTOCOL SUITE USER GUIDE



Mount name tp1usr
Host tp1
Mount path /usr
Name server tp1
User name root
Password -------

Acorn

TCP/IP PROTOCOL SUITE
USER GUIDE



Mount name	tp1usr
Host	tp1
Mount path	/usr
Name server	tp1
User name	root
Password	-------

Acorn

All correspondence should be addressed to:

Customer Support and Services
Acorn Computers Limited
Fulbourn Road
Cherry Hinton
Cambridge CB1 4JN

Information can also be obtained from the Acorn Support Information Database (SID). This is a direct dial viewdata system available to registered SID users. Initially, access SID on (0223) 243642: this will allow you to inspect the system and use a response form for registration.

This software and associated documentation is based in part on the Fourth Berkeley Software Distribution under license from the Regents of the University of California. We acknowledge the Regents of the University of California and the Electrical Engineering and Computer Science Department at the Berkeley Campus of the University of California and other contributors to that Software Distribution.

The NFS software is based in part on SUN RPC software distributed by Sun Microsystems, Inc. We acknowledge Sun Microsystems, Inc for the part that software plays in this product.

Published May 1990
Release 1
Published by Acorn Computers Limited
Part number 0484,001 Issue 1

Acorn Computers Limited: End-User Licence Conditions

1. Definitions

The following expressions have the meanings given here:

'Acorn' means Acorn Computers Limited, being either owner of all intellectual property rights in the Software, or having the right to grant licences of the Software.

'Developer' means any third party software developer who retains copyright in the Software.

'Documentation' means the printed user documentation supplied with the Software inside the pack.

'Software' means the programs contained in object-code form on the disc(s) supplied with these conditions:

2. Licence

Acorn grants you a personal non-transferable non-exclusive licence (or sub-licence), as follows:

(1) You may copy the Software for backup purposes, to support its use on one stand-alone Acorn computer system. (Separate provision for site licences is made on form APP157 available from your Acorn Authorised Dealer.)

(2) You must ensure that the copyright notices contained in the Software are reproduced and included in any copy of the Software.

(3) You may not:

- (i) copy only part of the Software; or
- (ii) make the Software or the Documentation available to any third party by way of gift or loan or hire;
- (iii) incorporate any part of the Software into other programs developed or used by you; or
- (iv) copy the Documentation.

3. Term

This licence remains in effect unless you terminate it:

(1) by destroying the Software and all copies, and the Documentation, or

(2) by failing to comply with the Conditions.

4. Limited warranty and disclaimer of liability

(1) Acorn warrants the disc(s) upon which the Software is supplied to be free from defects in materials and workmanship under normal use for a period of ninety (90) days from the date of purchase, as evidenced by a copy of your receipt. Your Acorn Authorised Dealer will replace a defective disc if returned within ninety (90) days of purchase.

(2) The Software is supplied 'as is'; neither Acorn nor the Developer makes any warranty, whether express or implied, as to the merchantability of the Software or its fitness for any particular purpose.

(3) In no circumstances will Acorn be liable for any damage, loss of profits, goodwill or for any indirect or consequential loss arising out of your use of the Software, or inability to use the Software, even if Acorn has been advised of the possibility of such loss.

5. General

These conditions supersede any prior agreement, oral or written, between you and Acorn relating to the Software.

Contents

About this Guide

Summary of contents

This Guide tells you how to use the RISC OS TCP/IP Protocol Suite. It is split into three major parts:

- *Part 1 – The desktop applications* tells you how to use the desktop applications supplied as part of the TCP/IP Protocol Suite.
- *Part 2 – Internet and NFS* tells you how to use the * Commands supplied by the Internet and NFS modules. Most of this part will mainly be relevant to system administrators and other more experienced users.
- *Part 3 – Appendices* contains two appendices.

  The first introduces you to the fundamentals of UNIX filesystems.

  The second tells you how to install the TCP/IP Protocol Suite on RISC OS computers – but only after some configuration has already been done. If you're the system administrator you should see the *TCP/IP Protocol Suite Installation Guide*, which tells you what you need to do before you can do this.

Command syntax

Special symbols are used when defining the syntax for commands:

- Angle brackets indicate that you must substitute an actual value. For example, <filename> means that you must supply an actual filename.
- Braces indicate that the item enclosed is optional. For example, [K] shows that you may omit the letter 'K'.
- A bar indicates an option. For example 0|1 means that you must supply the value 0 or 1.

Finding out more

For general information on the use of a RISC OS computer and its desktop interface, see the *Welcome Guide* and *User Guide* supplied with it.

For details of how to install the TCP/IP Protocol Suite, see the *TCP/IP Protocol Suite Installation Guide*.

For details of how to use the programming interfaces provided by the TCP/IP Protocol Suite, see the *TCP/IP Protocol Suite Programmer's Guide*, available separately from your Acorn supplier. This includes a disc of useful C library calls.

Reader comments

If you have any comments on this Guide, please complete and return the reader comment form on the last page to the address given there.

Part 1 – The desktop applications

Internet

Introduction

The Internet application holds various modules that provide the services and protocols needed for other TCP/IP Protocol Suite applications to work. All that happens if you run Internet is that it loads these modules, making the services and protocols available to these applications.

Normally you won't ever need to do so, because the other applications automatically run Internet if they need it and it's not yet been run. However, there's no reason why you shouldn't run Internet rather than the applications doing so. Also, it doesn't matter if you try to run Internet after it's first been run (either by you or by an application that needs it.) It can tell it's already been run, and so avoids unnecessarily loading a second copy of itself.

How it works

When RISC OS first 'sees' the Internet application (ie has to display its icon) it remembers where you've stored it. (It uses a system variable to do this). If another application needs the services and protocols in Internet it just finds out where Internet is stored and automatically runs it.

Of course if RISC OS hasn't 'seen' Internet before you try to run an application that needs its services, that application won't know where Internet is and so won't be able to run it. But so long as you (or your system administrator) have followed our installation instructions, you won't get this problem.

Finding out more...

The chapter in Part 2 entitled *Internet* tells you:

- what modules and programs are available within the Internet application
- what * Commands they provide you with
- how to use those * Commands.

The NFS Filer

Introduction

The NFS Filer is an application that gives you access to files on another computer – typically one that runs UNIX. The NFS Filer displays the files in a directory display, just like other RISC OS filing systems. You can then treat the files in exactly the same way as any other files in RISC OS – you use the same methods you've already learned to copy files, rename them, delete them and so on.

You can also use the NFS Filer to send files for printing to a remote computer – again typically one that runs UNIX – which then forwards them to a printer. You do this using the standard RISC OS printer drivers, so again there's very little new to learn.

There are some restrictions, however:

- The other computer needs to be able to understand the messages that your RISC OS computer is sending it. The NFS Filer software uses the NFS protocol to communicate; the other computer must be able to be an *NFS server*.

  This is a widespread standard among UNIX computers, but is not supported by all manufacturers, and so you may find you can't use the NFS Filer with some of your computers. If you have any doubts, ask your system administrator.

  Even if you can't use NFS to handle UNIX files, you may still be able to use the VT220 application with the Telnet *or* Ftp *protocols, or the* *Telnet *program, or the* *Tftp *program to do what you want. See the corresponding chapters for further details.*

- A UNIX computer that supports NFS has a level of control over which parts of its filing system can be accessed by which users from which machines. You may find that you are not allowed to access the computer you want to.

- Files on UNIX computers have access permissions, which set whether you can read, write and execute (run) a file. Just like any other network (including Econet) or multi-user system, you may find that you are not allowed to read every file.

  For more information about UNIX access permissions, see the appendix entitled UNIX filesystems and access.

If you have problems accessing a computer, and you can't get any help from this chapter or your UNIX documentation, you should see your system administrator.

Using NFS from the desktop

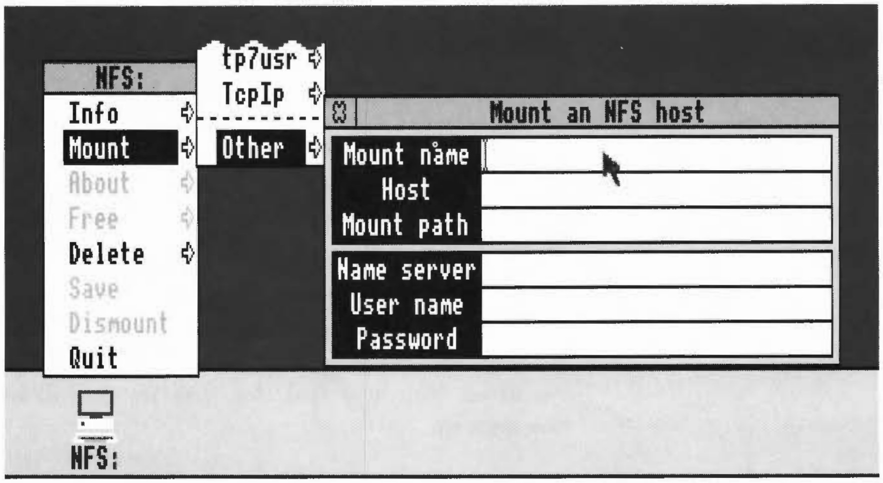
To run the NFS Filer:

- open the directory display that contains the NFS Filer application
- double-click on its icon.

The NFS Filer icon appears on the icon bar.

Mounting an NFS host

Before you can access the files on an NFS server, you need to *mount* it. This is a very similar process to logging on to an Econet file server. Call up the icon bar menu, and choose Other from the Mount submenu. A dialogue box appears:



Now you need to fill in each field:

Mount name

This is a name which is associated with the mount. You can choose any string that you have not already used as a mount name. It is used:

- On the icon bar beneath the NFS Filer icon

  Each time you mount another computer, another NFS Filer icon appears on the icon bar. The name tells you which mount belongs to which icon.

- As the 'disc name' in a full NFS pathname

  Again, this is how you distinguish different mounts you have made.

- As the name of a mount when you save it

  You can save frequently used mounts so you don't have to fill in each field of this dialogue box every time you use the mount. See the section entitled *Saving and deleting mounts* later in this chapter.

Host

This is the host name of the NFS server you wish to mount so you can access files on it.

Mount path

This is the pathname of the directory you wish to mount. It will typically be a UNIX pathname.

- You don't have to mount the root directory of the NFS server; but you should bear in mind that you will not be able to reach any directories that are higher in the directory tree than the one you mounted.
- The NFS server must have *exported* the directory (declared it to be available for use by other computers).

Name server

This is the host name of the NFS server that checks the user name and password you give.

- This need not be the same as the machine you're trying to mount; indeed, your system administrator may well have set up just one machine to be a name server.
- If your system administrator hasn't told you to use a particular machine, you'll normally find it best to use the machine you're trying to mount as the name server.
- Obviously the name server needs to have a record of the user name and password you give. (One way you can check this is to try to log on to the name server, either directly, or using Telnet.) If it doesn't, then the mount will fail.
- The name server must have a utility known as the `pcnfsd` daemon running on it.

User name

This is the user name that you want to use to access the computer. If you don't know a valid user name, you can use the name `nobody`.

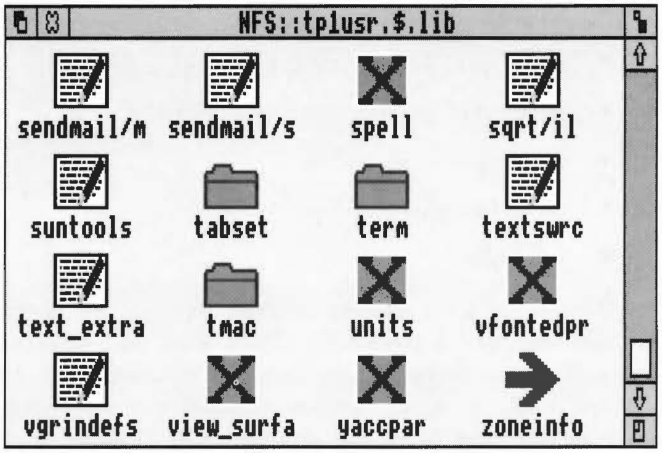
- The user name you use will affect what access you have to files, for the entire duration of the mount. The user `nobody` is an *unprivileged user*, with similar access rights to the user guest – you won't get group or owner access to any of the files. See the appendix entitled *UNIX filesystems and access* for more details.

Password

This must be the password for the user name you have used.

- If you used `nobody` as your user name, then there is no password – just press *Return*.

When you have filled in all the fields the mount takes place, and the root directory of the mount (ie the directory you mounted) is displayed:



You can treat this directory display just like any other. Note, however, that:

- **Long filenames on the UNIX system may cause problems** when used with the desktop Filer.

  By default RISC OS NFS is configured to truncate UNIX filenames to 10 characters, so you won't get this problem – but you can change this: see the chapter in Part 2 entitled *File mapping*.

- **Copying with the Q (Quick) option set may cause 'Not enough free memory' errors**, especially when there is little free memory in both the RMA and the application space.

  If this happens to you try releasing some memory for the NFS Filer to use, or not using the Q option when you copy files.

File mapping

There are a number of differences between the NFS and RISC OS models of a filing system, the more important being:

- length of filenames
- use of special characters in filenames
- numbers of attribute bits stored with files
- meaning of attribute bits
- use of file types
- soft links.

Because of these clashes changes have to be made when mapping RISC OS filenames and attributes to UNIX ones, and vice versa. Generally the changes made when mapping one way are reversed when mapping the other way, so the system is as transparent as possible if only viewed from RISC OS. If you view the files using the remote server though, you'll notice some differences. In particular many filenames will have extensions that are used to store RISC OS file types. By default these take the form '`,xxx`' (where `x` is a hexadecimal digit), but you can reconfigure this to your own sceme. See the chapter in Part 2 entitled *File mapping* if you need more details.

New file types

You may have noticed that in this example directory display there are two file icons that you haven't seen before.

A UNIX Ex file icon

A SoftLink file icon

UNIX Ex files

If a file has a *UNIX Ex* (short for *UNIX executable*) icon, it will typically be a program or a script of commands to run under UNIX. You won't be able to run such files under RISC OS. You can edit scripts, though – but only if the NFS server will let you read and write the files.

For a more detailed description of how the NFS Filer decides what files have type UNIX Ex, see the chapter in Part 2 entitled *File mapping*.

SoftLink files

If a file has a *SoftLink* icon, it represents a UNIX soft link that NFS was unable to resolve. You cannot do anything with one of these icons; if you try to do so, you'll get an error message to remind you.

- Normally the NFS Filer resolves soft links transparently; you'll just see an icon that represents the object (eg a directory) named by the soft link.

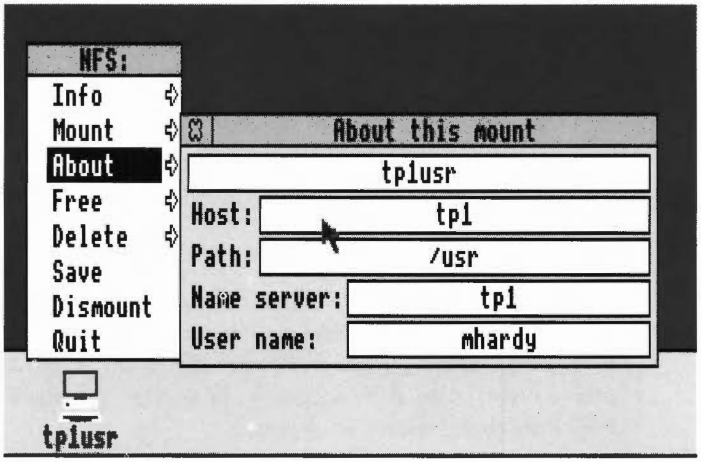
You'll find a more detailed description of how NFS resolves soft links and why it may be unable to resolve one in the chapter in Part 2 entitled *File mapping*.

Other options…

Once you've mounted an NFS server, the other options on the icon bar menu are no longer shaded, and you can use them.

Getting information about a mount

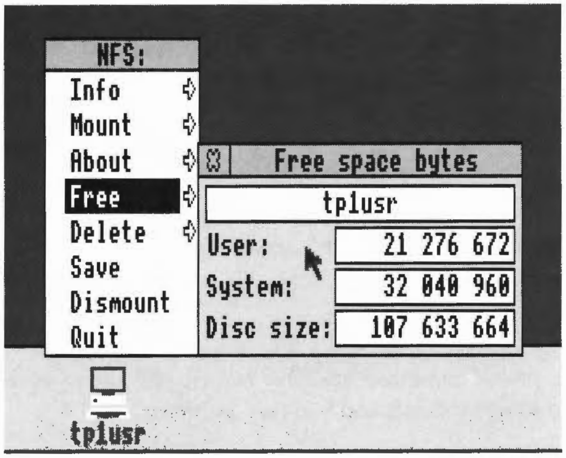
About displays the mount's name, host, path, name server and user name:



tplusr
Host: tpl
Path: /usr
Name server: tpl
User name: mhardy

Finding the free space on a mount

Free opens a dialogue box that shows you the amount of free space on a mount:



Free space bytes

tplusr

User: 21 276 672

System: 32 040 960

Disc size: 107 633 664

- **User** shows the number of bytes free for you to use.
- **System** shows the number of bytes free for the system to use.
- **Disc size** shows the disc size in bytes.

Dismounting an NFS host

Dismount ends a mount. Any directory displays belonging to the mount are closed.

Quitting the NFS Filer

Quit leaves the NFS Filer; you can no longer make mounts from the desktop. However, a RISC OS module (named NFS) that provides access to the files over NFS remains loaded. If you still have anything mounted it remains so, and so you can still access it from the command line. If you restart the NFS Filer these mounts re-appear.

Saving and deleting mounts

You can keep a list of the details of your frequently used mounts. If you want to use one of these saved mounts you no longer have to remember these details to fill in the mount dialogue box. You can also delete mounts from your list.

Saving a mount

To save a mount choose Save from the icon bar menu. The mount and its details get added to your list using its mount name. If you've already saved a mount with the same name it gets overwritten.

Mounting a previously saved mount

Once you've saved a mount its name is added to the Mount submenu of the icon bar menu. From the name you can open a dialogue box which already has the details of the mount filled in, and the caret positioned ready for you to type in its password.

If you want to you can edit the details before making the mount; so you could, for instance, change the user name of a saved mount but leave the other details the same.

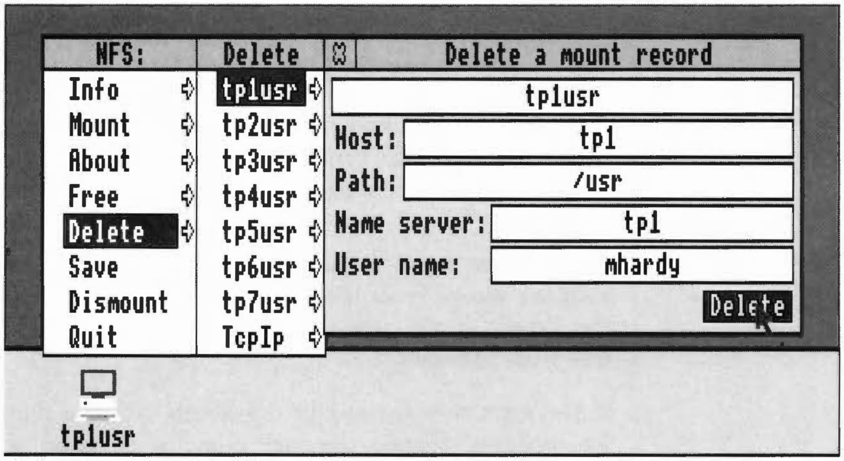
If the mount doesn't need a password you can just click on its name on the Mount submenu; you don't need to open the dialogue box.

Deleting a mount from the saved list

You can delete a mount from your saved list when you no longer need it. You can:

- choose Delete from the mount's icon bar menu if it's currently mounted
- choose the mount by name using the Delete submenu from any NFS Filer icon bar menu.

If you're unsure whether a mount name is the one you want, you can check its details first by calling up a dialogue box from the Delete submenu:



tp1usr
Host: tp1
Path: /usr
Name server: tp1
User name: mhardy

You can then click on the Delete icon to actually delete it from the list of saved mounts.

So in the above illustration, clicking on any of the black regions would delete `tp1usr` from the list of saved mounts.

Using UNIX printers

You can also use the NFS Filer to send files for printing to a remote NFS server which then forwards them to a printer. You do this using a standard RISC OS printer driver, setting it to print to a special 'filing system' provided by the NFS module.

The NFS#Printer filing system

The filing system is called NFS#Printer, and has this syntax:

```
NFS#Printer::<server>[.$][.<printer name>[.<user name>[.<options>]]]
```

- The <server> must be running a utility known as the `pcnfsd` daemon. You need this same utility to use the server as a name server, so:

  Any server you can use as a name server you can also use as a print server, and vice versa.

- You can include the '$' if you wish, but you never need to. It's there just to maintain compatibility with other RISC OS filing systems.

- The <printer name> must be the name of a printer that the server can access. If you don't give a printer name, the server will use the printer named `lp` – the default name for a UNIX printer.
- The <user name> gets passed to the program that does the printing (*lpr* in the case of UNIX), and is used to establish your access rights to the printer. If you leave this out the server uses a blank user name; you can do so if there are no restrictions on printer use at your site.
- The <options> also get passed to the program that does the printing. If you omit them, no options are forwarded.

Setting up your RISC OS printer driver

To set this up you must first choose the RISC OS printer driver that is appropriate to the remote printer you want to use. For example, to print to a PostScript printer you would use PrinterPS. If the printer driver's not already running, start it.

Then from the icon bar menu display the File dialogue box, and fill in the name of the *NFS#Printer* 'file' you want to use. For example:

`NFS#Printer::tp1`	Use printer lp (the default) on server tp1, blank user and options
`NFS#Printer::tp3.ps`	Use printer ps on server tp3, blank user and options
`NFS#Printer::tp1.lp.mhardy`	Use printer lp on server tp1 as user mhardy, blank options

Once you've set up the printer driver you use it just as you always have. Any files you drop onto the printer driver get converted to the correct format for the remote printer (eg PostScript), and then forwarded to it.

Finding out more...

The chapter in Part 2 entitled *File mapping* tells you:

- how filenames, file types, contents, attributes and dates are mapped from RISC OS to UNIX and vice versa
- how RISC OS NFS resolves soft links.

- what * Commands the NFS module provides you with
- how to use those * Commands.

The NFS MailMan

Introduction

The TCP/IP software comes with a new version of the RISC OS Mailman program, that has extensions to allow you to send and receive mail over NFS. How you use it depends on what interfaces you have fitted to your computer:

- If you have an Econet interface you can switch between sending and receiving mail via the Econet (just as the standard MailMan does), or sending and receiving mail via NFS.
- If you only have an Ethernet interface you can send and receive mail over NFS, but not over Econet.

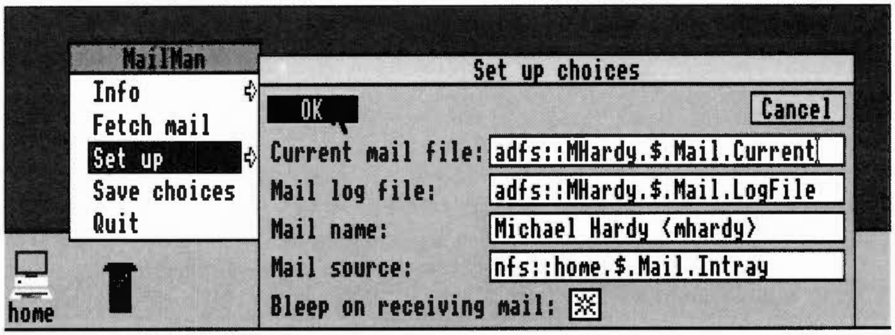
To use the Econet mail system you need to have a home directory on an Econet file server with certain files and subdirectories set up. Likewise, to use the NFS mail system, you need to have a home directory on an NFS server with certain files and subdirectories set up. If your system administrator has not already done this for you, and told you what your home directory is, ask him to do so.

Starting MailMan

You start the NFS MailMan in just the same way as the standard MailMan: double-click on the application's icon. The MailMan icon appears on the icon bar.

source

MailMan. To do so, choose *Set up* from MailMan's icon bar menu. A dialogue box appears:



Current mail file: adfs::MHardy.$.Mail.Current
Mail log file: adfs::MHardy.$.Mail.LogFile
Mail name: Michael Hardy <mhardy>
Mail source: nfs::home.$.Mail.Intray
Bleep on receiving mail: ☒

Change the field to the value you want (see below), and close the dialogue box. To make the change permanent, choose *Save choices* from the icon bar menu.

Setting up MailMan to use Econet

To send and receive mail over Econet:

- You need to be logged on to your normal Econet file server, using your own user name. In this example we're using the file server `TechPubs`.
- Your *Mail source* needs to be set to the subdirectory `Mail.Intray` of your home directory, eg:

  ```
  net#TechPubs:&.Mail.Intray
  ```

Using MailMan with Econet

The NFS MailMan works just like the standard MailMan when you're using Econet. See the *User Guide* supplied with your computer for details.

Setting up MailMan to use NFS

To send and receive mail over NFS:

- Your *Mail source* needs to be set to the subdirectory `Mail.Intray` of your home directory.
- Your *Mail source* needs to be visible to MailMan via an NFS mount. You can use any *mount name* you like.

For example if your home directory were `/usr/staff/mhardy`:

- If you'd mounted your home directory using the mount name `home`, your *Mail source* would have to be `nfs::home.$.Mail.Intray`
- If you'd mounted `/usr` using the mount name `usr`, your *Mail source* would instead have to be `nfs::usr.$.staff.mhardy.Mail.Intray`

Sending mail using NFS

Sending mail over NFS is very similar to using Econet. The only change is in how you use the *To:* and CC: fields; the general syntax is:

<user>`@`*<mail address>*

The mail gets moved straight to a queue on the UNIX server. The server periodically empties this queue, looks at who the mail's addressed to, and uses its own mail system to deliver the mail over the network. Consequently the <user> and <mail address> must be in the format that your UNIX server's mail system uses. This will vary from site to site, depending on how the mail program you use has been configured. See your system administrator for details.

No more than ten messages can be in the queue at once. If you find the queue keeps filling up, ask your system administrator to make the server empty it more often.

Receiving mail using NFS

Receiving mail over NFS works just like the standard MailMan. See the *User Guide* supplied with your computer for details.

Using UNIX mail with Telnet

You may prefer to send and receive mail over the network by using Telnet to connect as a terminal (you can use either the VT220 and Telnet applications, or the *Telnet command), and then using the UNIX *mail* program. If you decide to do this make sure your system administrator knows, so that he can stop your UNIX server from trying to interact with MailMan instead.

Finding out more...

For more information on using MailMan, see the *User Guide* supplied with your computer.

There's no further information about MailMan in Part 2 of this Guide.

VT220

Introduction

The VT220 application provides a full emulation of a DEC VT220 terminal that runs in a window within the RISC OS desktop. It will also emulate the earlier VT52 and VT100 terminals. The window will behave just as a VT220 would; if you normally use another computer (such as an Acorn RISC iX one) the results you get may not quite be those you're used to.

With the software supplied in the TCP/IP Protocol Suite, you can use VT220 in three different ways:

- as a terminal to a remote computer, using the Telnet protocol
- as a terminal to a remote computer, using the Ftp protocol
- as a conventional terminal connected to a host computer using a serial cable.

Telnet

The main way we expect you'll want to use VT220 is as a conventional terminal, using the Telnet protocol to communicate over a TCP/IP network with a remote computer – typically one that runs UNIX.

- The other computer needs to be able to understand the messages that your RISC OS computer is sending it – it too must 'understand' the Telnet protocol.

  This is a widespread standard among UNIX computers, but is not supported by all manufacturers, and so you may find you can't use Telnet with some of your computers. If you have any doubts, ask your system administrator.

Ftp

You can use the Ftp protocol (a file transfer protocol) instead of Telnet. This is more restricting; there is only a limited set of commands you can use, all aimed at the job of transferring files. Just as with Telnet, the remote computer must understand the protocol you are using. Because this is a simple protocol, it's one that's particularly widely supported.

Serial connections

You can also use VT220 as a conventional terminal, directly connected via a serial line to any computer that will support VT220, VT100 or VT52 terminals.

1 Mbyte machines

VT220 is a large application. The amount of memory it needs depends on the use to which you're going to put it. If you've got a 1 Mbyte machine, you'll find that:

- VT220 with Telnet will just fit in memory
- VT220 with Ftp is too large to fit in memory
- VT220 with Serial will easily fit in memory.

Alternatives to Ftp

However, there are alternatives to using Ftp. You can transfer files from the command line using the *Tftp command. (For a full description see the chapter in Part 2 entitled *Internet.*) You can, of course, also use the NFS Filer to transfer files. Provided the host machine you want to access supports NFS, you'll find this vastly preferable to using the *Tftp command.

Alternatives to Telnet

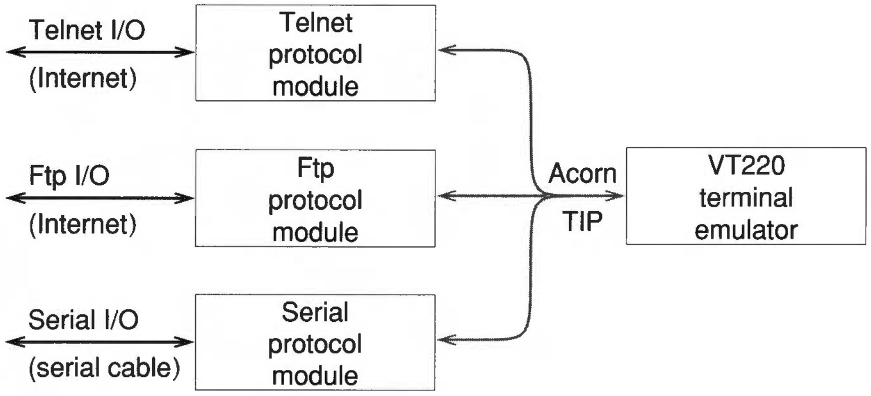
You can also use the Telnet protocol from the command line with the *Telnet command. (For a full description see the chapter in Part 2 entitled *Internet.*)

Running several emulations at once

With the VT220 application you can have several terminal emulations running in their own windows at the same time. There's no reason why the different emulations can't use different protocols to communicate with different computers.

Protocol modules

The reason for this is the way that VT220 works:



Various *protocol modules* convert the different protocols used for input and output to an internal protocol called the *Acorn Terminal Interface Protocol*, or *Acorn TIP*. VT220 only understands Acorn TIP, and so needs the protocol modules to be of any use. The Protocols application contains Telnet, Ftp and Serial protocol modules; it is supplied as a part of the TCP/IP Protocol Suite. For further details see the chapters entitled *Protocols*, *Telnet*, *Ftp* and *Serial*.

Using other protocol modules and emulators

In the future, you may be able to buy other protocol modules and/or terminal emulators, either from Acorn or other third parties. Because Acorn TIP provides a standard interface between protocol modules and terminal emulators, you'll be able to use them with the TCP/IP ones. So:

- if you get any more protocol modules, you'll be able to use them with VT220
- if you get any more terminal emulators, you'll be able to use them with the Telnet, Ftp and Serial protocol modules.

Starting up

To run VT220:

- open the directory display that contains the application
- double-click on its icon.

The VT220 icon is installed on the icon bar.

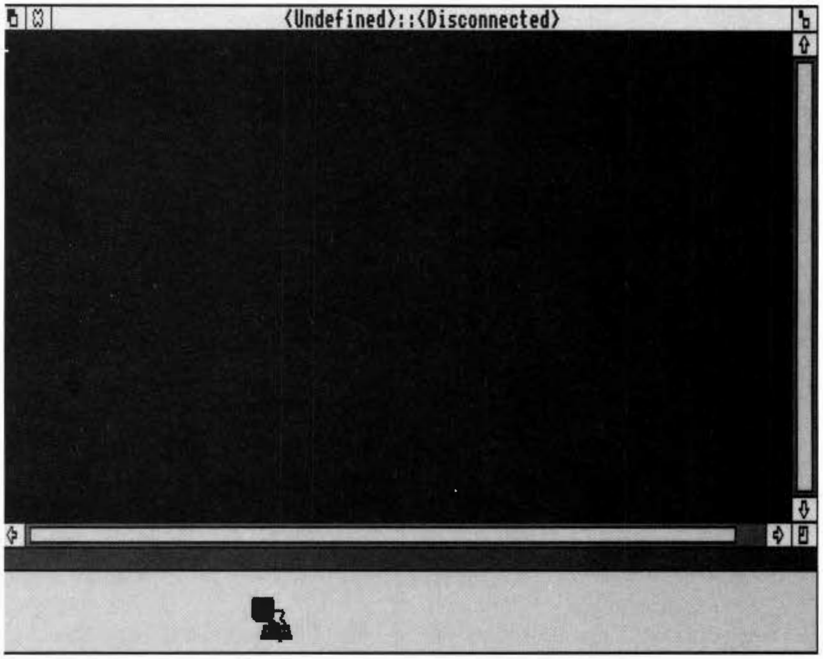
See also *Starting up with VTSetup and VTScript files* below.

Opening a terminal window

To use VT220 you need to open a terminal window:

- click on the VT220 icon, or
- choose Open from the icon bar menu.

Initially the terminal window might look something like this:



Opening a connection

Once you've selected a protocol for the window, if it doesn't need any more information to open a connection (eg the Serial protocol) then it automatically does so.

You only need to explicitly open a connection if the protocol you've chosen needs more information (eg the host name for a Telnet network connection). Choose Open from the Connection submenu; the protocol module you're using will then prompt you for the extra information. For instance, both Telnet and Ftp will prompt you for the host name of the machine to which you want to connect.

To abort a connection attempt while it is still in progress, choose Close from the Connection submenu.

Closing a connection

To close a connection, simply choose Close from the Connection submenu. The protocol remains selected.

Selecting text and using the selection

You can select any text in a terminal window just as you do in Edit: either by using Select to drag through the text, or by using Select and Adjust to set the start and end of the block of text.

Once you've selected some text, you can use the Select submenu of the main menu to save the text, to 'type' (paste) it into a terminal window, to get a file with that name, or to clear the selection of text. In more detail:

Save

This option pops up a standard Save dialogue box, which you then use to save the selected text to a file. The **Text only** option (see below) sets the format used. You cannot use this option to save the text to another terminal emulator; you must instead use the Type option (see below).

Clear

Clears the selection – ie makes it deselected.

Type

This option types the selected characters into the terminal window with the input focus, just as if they had been typed from the keyboard. If no terminal window has the input focus, you'll get an error message.

The terminal window's title

When you first open a terminal window, its title will usually be `<Undefined>::<Disconnected>`. In general, a terminal window's title has the following form:

Protocol`::`*Hostname* `P=`*Script name*

- if no protocol is selected for the window, *Protocol* is `<Undefined>`
- if no connection is open for the window, *Hostname* is `<Disconnected>`
- if a connection is open over a communication protocol which does not support host names (eg the Serial protocol), *Hostname* is blank
- if the terminal emulator is currently attempting to open a connection for the window, *Hostname* is `<Connecting>`
- if no script is currently running in the window, the `P=...` part of the window title is omitted.

Choosing a communication protocol

Before you can use a terminal window for anything you have to choose which protocol you will be using for that window – and hence which protocol module you need. Use the Protocols submenu of the main VT220 menu.

- For a protocol to be on the menu, you need to have started its protocol module. If you don't see the protocol you want to use, click on Load to open the protocol module directory display in the Protocols application; then double-click on the protocol module.
- If the Load option's greyed out it's because RISC OS hasn't 'seen' the Protocols application (ie hasn't had to display its icon), and so it doesn't know where the protocol modules are. Open the directory display containing the Protocols application so its icon is displayed. Then try to load the protocol module again.

The Protocol submenu

When you open a new terminal window there is a *Protocol* entry on its menu, which is greyed out. Once you choose a protocol this entry gets replaced by the protocol's name, which in turn leads to a submenu. This gives you access to any extra features supplied by the protocol module. **You should see the chapters entitled *Telnet, Ftp* and *Serial* for details of the respective submenus.**

Get file

This option assumes the selected text is a filename and tries to get the file. Of the protocol modules supplied with VT220, only Ftp supports this option; for any other protocol this menu entry will always be greyed out.

Text only

Toggles text only mode on and off.

When text only mode is on, selected text gets saved as plain ASCII characters, ignoring any special attributes. When text only mode is off, VT control codes and character attributes get added to the selected text when it's saved, so that if you then send the file to a VT terminal you'll get exactly the same image as in the terminal window.

New file types

VT220 uses two new file types:

A VTSetup file icon

A VTScript file icon

VTSetup files

You can change VT220's behaviour in many ways: how it displays received text, how the keyboard behaves, and so on. You can use a VTSetup file to store how you've set up a terminal window, and to then re-use that setup for other sessions.

VTScript files

VT220 supports a command script language, which you can use to control things automatically, such as selecting a communication protocol or logging on to a remote host. You should use the VTScript file type for any files of these commands. For full details of VT220's command script language see the chapter in Part 2 entitled *VT220*.

While a script's running you can still use the keyboard to type things in, and the terminal remains fully operational.

See also the later section entitled *Controlling scripts*.

Changing the default setup

You can change the default setup used by VT220 by replacing the `!Setup` file within the VT220 application.

1. Open the `!VT220` application directory by holding down the *Shift* key while you double-click on the `!VT220` icon.
2. Rename either `!SetupCol` (the default colour setup; `!Setup` is initially a copy) or `!SetupB+W` (the default monochrome setup) as `!Setup`.
3. Alternatively, create a VTSetup file that contains the new default setup, name it `!Setup`, and drag it to the VT220 application directory .

Starting up with VTSetup and VTScript files

You can also start up VT220 and open a terminal window by double-clicking on a VTSetup or a VTScript file, or by dragging the file to the VT220 icon on the icon bar:

- a VTSetup file loads the saved setup into the new terminal window
- a VTScript file is loaded and run as soon as the window has opened.

Loading VTSetup and VTScript files

Likewise, if you drag a VTSetup or a VTScript file into an existing window:

- a VTSetup file loads a saved setup into the terminal window
- a VTScript file is loaded and run immediately.

Controlling scripts

Once you've loaded a command script into the terminal emulator you can control it using the Script submenu of the main menu:

Re-running a script

To re-run a loaded script, choose Run from the Script submenu.

Aborting execution of a script

To abort the execution of a script in the middle, choose Stop from the Script submenu. The script remains loaded.

Saving a loaded script

To save a script you've already loaded, choose Save from the Script submenu. This is useful to save a script to an editor or to another terminal window; it will be exactly the same as that which you loaded.

Spooling terminal sessions

The Spool submenu allows you to spool parts or all of your terminal session to a text file:

Open Opens the spool file you specify and starts spooling to it.

Pause Pauses spooling to the file.

Continue Restarts spooling to the file.

The text gets appended to what was already spooled.

Close Finishes spooling to the file and closes it.

Loading text and terminal sessions

You can load any Text file into VT220 – just drag the file into an existing window. The characters in the file get sent to VT220 just as if you'd typed them at the keyboard.

- You may first have to change carriage returns to line feeds (or vice versa) to fit in with how the remote host expects lines to end. If you're using Edit you can easily do this by choosing CR↔LF from the Edit submenu
- If you're dragging the file into VT220 so you can view a previously spooled session, you'll want to have VT220 in local mode. That way the remote host won't try to execute all the text as commands – including the text it (or some other remote host) spooled earlier to the file.

The keyboard

A VT220's keyboard has different keys to those on a RISC OS computer. So that all a VT220's keys are available on the emulator, the following keyboard mapping is done:

RISC OS key pressed	**VT key code sent**
f5	Break
f6 – f12	f6 – f12
Shift-f6 – Shift-f12	Shift-f6 – Shift-f12
Ctrl-f3 – Ctrl-f10	f13 – f20
Ctrl-Shift-f3 – Ctrl-Shift-f10	Shift-f13 – Shift-f20
Ctrl-f1	Ctrl-f5 *(Send answerback string)*
keypad NumLock	PF1 *(Gold)*
keypad /	PF2
keypad	PF3
keypad #	PF4
keypad +	Keypad , (when in application mode)
keypad Enter	Enter
Page Up	Prev Screen
Page Down	Next Screen
Insert	Insert Here
Delete	Remove
Home	Find
Copy	Select
Left Alt key	Compose Character

Terminal setup – the main submenu

A VT220 terminal has a setup screen you can use to alter its configuration options. The emulator's Setup submenu provides you with the same facilities.

Display, Keyboard, General and Printer

These entries all lead to extensive submenus which we describe below. First, though, we describe the other facilities provided at the top level of the Setup submenu.

Local

Toggles local mode on and off.

When local mode is on, characters typed at the keyboard are sent to the screen but **not** to the remote host. When it is off (the default), characters are sent to the remote host; they may also be sent to the screen, depending on the setting of local echo mode – see above.

Default

Use this option to restore the setup configuration to a default state. In the text below we indicate what those default values are.

Scrollback

Toggles scrollback mode on and off.

When scrollback mode is on, the terminal window buffers 100 lines of text, even when they scroll off the top of the window. You can then use the vertical scroll bar of the window to scroll back through these lines, view them and select text from them. The actual terminal screen is always represented by the bottom 25 lines.

If you select this mode, it uses an additional 60K of memory. By default it isn't selected.

Clear screen

Use this to clear the terminal's screen.

Save

This entry leads to a Save dialogue box, which you can use to save the current setup options to a file. Simply drag the file icon to a directory viewer.

The file's default name is `Setup`; it has a new type called VTSetup (type number &FDE).

To reload your saved setup, either drag the setup file into an open terminal window, or double-click on it to open a new terminal window.

Lock

This option locks the setup choices so you can't change them – nor can you unlock them again.

This is especially useful to lock a setup before you save it to a VTSetup file. If you then give the file to other users, they can use the setup but they can't change it.

Display submenu

The Display submenu controls how the emulator's display works:

Smooth Jump

Chooses smooth scrolling (one pixel at a time – which can be quite slow), or jump scrolling (one line at a time – the default).

Wrap

Toggles wrap mode on and off.

If wrap mode is on (the default), the cursor wraps from the right edge of the window to the start of the next line, scrolling the window if necessary. If wrap mode is off, the cursor stays at the end of a line when it reaches there.

Reverse

Toggles reverse display on and off.

When reverse display is off (the default) characters are light on a dark background. Turning reverse display on inverts this.

Insert

Toggles insert mode on and off.

When insert mode is on (the default), and you type with the cursor in the middle of a line of text, your characters are inserted between the ones that already exist. If insert mode is off, your characters overwrite those that are already there.

80 columns, 132 columns

Set the window's width to 80 (the default) or 132 characters per line.

Cursor submenu

This entry leads to a further submenu from which you can control the style and state of the cursor:

- **No cursor** disables the cursor display
- **Block / Line** chooses either a block cursor or a line cursor (the default)
- **Blink** toggles whether the cursor blinks (the default) or doesn't.

Display controls

Toggles display controls mode on and off.

When display controls mode is on, any received control characters in the ranges 0–&1F and &80–&9F are displayed on the screen. When it's off (the default) they're instead executed by the emulator.

Margin bell

Toggles the margin bell on and off.

When the margin bell's turned on (the default) and the bell's enabled (see *Bell* below), it sounds when there are only five characters until the right margin. Otherwise it doesn't sound when you reach the margin.

Local echo

Toggles local echo mode on and off.

If local echo mode is on, any characters you type get sent to the terminal window as well as to the remote host. If local echo mode is off (the default), characters just get sent to the remote host. Which of the two you use depends on whether the remote host reflects characters itself.

See also the description of local mode above.

Delete

Toggles interpretation of the delete (&7F) character.

When delete mode is on (the default), the delete character is interpreted as the sequence ASCII 08, ASCII 32, ASCII 08 (backspace, space, backspace); this deletes the character to the left of the cursor. When delete mode is off, the delete character is ignored; this is how a VT220 behaves. **This option is an extension to the VT220 setup options.**

to display text in VT220:

- **Foreground** sets the colour used for normal text.
- **Background** sets the colour used for the background.
- **Selection** sets the colour used to display text that you have selected.
- **Bold** sets the colour used to display text that has its bold attribute bit set.
- **Flashing** sets the colour used to display text that has its flashing attribute bit set.

Keyboard submenu

The Keyboard submenu controls how the emulator's keyboard works. These settings only take effect if the corresponding window has the input focus.

Repeat

Toggles keyboard repeat mode on and off.

If repeat mode is on (the default) and you hold a key down for more than 30cs, then it auto-repeats. If repeat mode is off, keys don't auto-repeat.

App keypad

Toggles keypad state.

When this option is on, the keypad keys send VT keypad application codes. When it's off (the default), the keypad keys send numeric ASCII characters.

App cursor

Toggles cursor keys state.

When this option is on, the cursor keys send VT cursor application codes. When it's off (the default), the cursor keys send VT cursor movement codes.

User keys locked

Toggles function keys lock state.

When function keys are locked they can't be redefined by the remote host; when they're unlocked (the default) they can be redefined.

Local Fn keys

Toggles function keys local state.

When function keys are set to local (the default) they work as RISC OS function keys, so you can use them to store text or as 'hot keys'. When they're not local they work as VT220 function keys – but you can only do this when you've got a connection open, or when VT220 is in local mode (see *Local* above).

CR, CR/LF, LF/CR

Makes the *Return* key send ASCII 13 (CR – the default), or ASCII 13 followed by ASCII 10 (CR/LF), or ASCII 10 followed by ASCII 13 (LF/CR).

General setup submenu

The General setup submenu controls general configuration options for the emulator.

Mode submenu

Use this submenu to choose the terminal emulation mode, and the terminal ID:

- **VT52** makes the window emulate a non ANSI VT52 terminal.
- **VT100** makes the window emulate a VT100 terminal; it doesn't support VT200 specific escape sequences.
- **VT200 7bit** makes the window emulate a VT220 terminal, but transmit 7-bit equivalents of 8-bit control codes.
- **VT200 8bit** (the default) makes the terminal emulate a VT220 terminal, transmitting full 8-bit control codes.
- The **Terminal ID** submenu chooses the response the terminal emulator issues if a remote host sends a terminal ID request: either a VT100 ID, or a VT200 ID (the default).

Tabs dialogue box

Use this dialogue box to set and reset the terminal window's tab stops. You can click Select:

- on a tab position to reverse its state
- on **Clear** to clear all tab stops
- on **Every 8** to set a tab stop every 8th position (the default)
- on **OK** to confirm the tab stops, and close the dialogue box.

Bell

Toggles the terminal bell on and off.

When the bell is off the terminal ignores received ASCII 7 characters, and doesn't sound the margin bell even if it is turned on in the Display submenu. When the bell is on (the default), it sounds when the terminal receives an ASCII 7 character, and may also be used as a margin bell.

Lock features

Locks and unlocks the user preference features.

When user preference features are locked the remote host cannot change them; when they're unlocked (the default) it can. The user preference features are:

- Keyboard lock
- Scroll mode (Jump / Smooth)
- Reverse display mode
- Auto repeat mode
- Tab stops.

Answerback dialogue box

Use this dialogue box to set the answerback string sent to the host computer when an INQ character is received (ASCII 5). It can be up to 20 characters long. Clicking on **Conceal** hides the answerback string, but it will still be sent to the remote host whenever an INQ character is received.

Printer setup submenu

The Printer submenu controls the setup of a locally connected printer:

Print mode submenu

This entry leads to a further submenu from which you can set how printing is controlled:

- **Normal** mode (the default) controls printing by escape sequences sent from the remote host.
- **Auto print** mode prints a line whenever the cursor leaves it. You can't use this mode with the RISC OS PostScript printer driver (PrinterPS).
- **Remote print** mode sends all received characters directly to the printer, without displaying them on the screen. You can't use this mode with the RISC OS PostScript printer driver (PrinterPS).

Dump mode submenu

This entry also leads to a submenu, from which you can select how a screen gets dumped to the printer:

- **ASCII** mode (the default) dumps the screen using plain ASCII text. You can't use this mode with the RISC OS PostScript printer driver (PrinterPS).
- **DEC** mode dumps the screen using DEC control codes, which gives better reproduction on DEC printers. You can't use this mode with the RISC OS PostScript printer driver (PrinterPS).
- **Dump** mode dumps the screen as a bitmap image to give exact reproduction. If a full size image is too big to fit on the page then it's scaled down to fit.

Finding out more...

The next chapter entitled *VT220 scripts* tells you:

- the syntax of VT220's command script
- about the NumLock module that VT220 uses to control the state of the *Num Lock* key.

The chapter entitled *Protocols* tells you:

- how the Protocols application works
- how to install extra protocols.

The chapters entitled *Telnet*, *Ftp* and *Serial* tell you:

- how to load and use the various protocols
- what items each protocol provides on its submenu.

The chapter in Part 2 entitled *Internet* tells you:

- how to run the Internet application from the command line
- how to use the *Telnet and *Tftp commands it provides you with.

VT220 scripts

Command scripts

The VT220 terminal emulator supports a command script language which you can use to control things automatically, such as selecting a communication protocol or logging on to a remote host. You can request input from the user, with an option of concealing the input, so you don't have to include passwords in your scripts.

A script file is a text file with type VTScript (&FDF). The file should contain one script command per line, which can be in either upper or lower case. There must be no blank lines, and the last character in the file must be the newline at the end of the last script command.

You can create such a file using any text editor. You'll then have to set the file type to VTScript using the *SetType command:

```
*SetType <full pathname> VTScript         or:
*SetType <full pathname> FDF
```

You can do so either from an Edit task window (opened from Edit's icon bar menu), or from the command line (reached by pressing F12).

Loading and running a script

To load and run a script do any of the following:

- drag a script file into an open terminal window
- double-click on a script file to open a new terminal window
- drag a script file to the VT220 icon to open a new terminal window.

In all cases the script file is loaded into the window and run. Once you've loaded a script into a terminal window you can also run it (again) by using the Script submenu.

While a script's running you can still use the keyboard totype things in, and the terminal remains fully operational.

script

submenu. The script remains loaded.

Saving a loaded script

To save a script you've already loaded, choose Save from the Script submenu. This is useful to save a script to an editor or to another terminal window; it will be exactly the same as that which you loaded.

Errors

If an error is encountered while running a script file, an error message is displayed and execution of the script stops.

Parameters

Many of the commands provided by the script language take parameters: some are strings, and some are numbers. These parameters are first processed by the SWI OS_EvaluateExpression (SWI &2D). There is a full description of how this SWI works in the *RISC OS Programmer's Reference Manual*; below we present a summary.

Strings

Apart from plain text, there are other things you can include in a string:

- To include a non-printing character in a string, you can use a special combination of two characters, the first of which is a vertical bar:

ASCII	**Symbols used**
0	|@
1 - 26	|<letter> eg |A (or |a) = ASCII 1, |M (or |m) = ASCII 13
27	|[or |{
28	|\
29	|] or |}
30	|^ or |~
31	|_ or |'
32 - 126	keyboard character, except for:
"	|"
|	||
<	|<
127	|?
128 - 255	|!<coded symbol> eg ASCII 128 = |!|@ ASCII 129 = |!|A

  Note that '|!' means set the top bit of the following character, even if it is set by another '|' character.

Alternatively enclose the character's code in angle brackets '< >'. The character code is normally in decimal, but you can use a leading '&' to specify it in hexadecimal. So to specify an ASCII DEL character (ASCII code 127) you could use either '<127>' or '<&7F>'.

- To include leading spaces in a string, the string must be in quotation marks, '"', which are not included in the definition.
- To substitute the current value of a system variable or macro in a string, include its name in angle brackets '< >'. For example to substitute the current value of Protocols$Dir (the system variable holding the location of the protocol modules) you would use '<Protocols$Dir>'.
- To include any of the above special characters ('"', '|' and '<') in a string, you must precede them with a vertical bar '|' to remove their special meaning. So you need to use '|"', '||' and '|<'.

Expressions

You can also make expressions using strings, numbers and the operators listed below. These evaluate to return a result that is either a number or a string.

As well as substituting the value of a system variable when it's in angle brackets within a string (eg '<Protocols$Dir>'), any other item which cannot immediately be treated as a string or a number is also looked up as a system variable. For example, in the expression FRED+1, FRED will be looked up as a system variable.

The operators recognised by the expression evaluator are as follows:

Arithmetic operators

+	Add two integers
–	Subtract two integers
*	Multiply two integers
/	Integer part of division
MOD	Remainder of a division

Logical operators

=	Equal	
<>	Not equal	
>=	Greater than or equal	–1 is TRUE
<=	Less than or equal	0 is FALSE
<	Less than	
>	Greater than	

>>>	Logical shift right
<<	Logical shift left
AND	AND
OR	OR
EOR	Exclusive OR
NOT	NOT

String operators

+	Concatenate two strings eg "HI" + "LO" = "HILO"
RIGHT n	Take 'n' characters from the right eg "HELLO" RIGHT 2 = "LO"
LEFT n	Take 'n' characters from the left eg "HELLO" LEFT 3 = "HEL"
LEN	Return the length of a string eg LEN "HELLO" = 5

Conversions

STR	Convert a number into a string eg STR 24 = "24"
VAL	Take the value of a string eg VAL "12" = 12

Type conversions

Where appropriate, type conversions are performed automatically. For example, if an integer is subtracted from a string, then the string is evaluated and an integer result is produced ("2"–1 gives the result 1). The null string "" is converted to 0 by both the implicit and explicit (VAL) conversions.

Similarly, integers will be converted to strings if necessary: the expression 1234 LEFT 2 will yield "12".

Priorities

The operators have the same relative priorities as their equivalents in BBC BASIC, eg * is higher than + which is higher than >, etc.

Registers

The script language supports the following registers:

R0-R9: numeric registers

R9 is special, and increments automatically every time a character is received (or is typed at the keyboard when the terminal is in local mode).

S0-S4: string registers

These registers can contain a string up to 20 characters.

VTstate

This is a special integer register having the following values:

0	no protocol is selected for the terminal
1	a protocol is selected, but no connection is open
2	a connection is pending
3	a connection is open

VTlast

This is a special string register containing the last 20 characters received from the remote host (or typed at the keyboard if the terminal is in local mode).

In expressions you can treat the registers just as if they are system variables. This is because they are temporarily stored in system variables, but **only** while the current script is the foreground task. Consequently:

- You must be careful to clear any registers holding sensitive information as soon as you've finished using them, as they're not secure. This particularly applies if you're storing a password in a register.
- You can't use registers to pass values between scripts running in different windows, because only one of them can be the foreground task at any one time.

Syntax conventions

In the descriptions below, we use these syntax conventions:

<str>	any valid expression evaluating to a string (see above)
<expr>	any valid expression evaluating to an integer (see above)

Arguments surrounded by [ ] are optional.

Arguments separated by | are alternatives.

#

A line beginning with '#' is treated as a comment and ignored.

Close

The Close command stops execution of the script, and closes the related window.

Connect [<str>] [–wait] [–fail <expr>]

The Connect command opens a connection to a remote host from within a command script. <str> is a protocol specific string and is passed to the TIP module with the open request. It consists of the fields used in VT220's Open dialogue box, in order (top to bottom) and comma separated.

- If the –wait flag is not present, execution continues immediately on the next instruction, even if a connection is not open (eg when the protocol module is waiting for the user to provide a host name). Since you cannot assume that the connection has been opened successfully, you should check the VTstate register.
- If –wait is present, execution of the script continues only when the connection is successfully opened. If the connection fails, an error will be reported; execution of the script then stops unless –fail is present, in which case execution continues at the location pointed to by the label <expr>. (See the *Label* command below.)

Disconnect

The Disconnect command closes the connection to the remote host; if no connection is open an error is reported, and execution of the script stops.

Error <str> [–fatal]

The Error command reports an error to the user. <str> is displayed in a Wimp error box.

- If the –fatal flag is present, execution of the script stops.

Goto <expr>

The Goto command makes execution of the script continue from the position pointed to by label <expr>. (See the *Label* command below.)

If <expr> Then ...

The command following Then is only executed if the value of <expr> is non-zero.

Input <expr> <str1> [<str2>...<str4>] [–default] [–conceal]

The Input command requests textual input from the user. You can specify up to four strings; these are used as titles for the input fields of a dialogue box that is opened on the screen. On exit register S<expr> contains the value of the first field, S<expr>+1 the value of the second, and so on.

- If the –default switch is present, the registers used to return the input strings are expected to contain default values on entry, which are displayed in the appropriate input fields.
- If the –conceal flag is present, '–' characters will be displayed as the user types into the last field of the dialogue box. This is particularly useful for entering passwords without displaying them. For example:

  ```
  Input 0 "Username:" "Password:" -conceal
  ```

  reads a user name and password into registers S0 and S1, concealing the password as it is the last field in the dialogue box.
- You must be careful to clear any registers holding sensitive information as soon as you've finished using them, as they're not secure. This particularly applies if you're storing a password in a register.

Label <expr>

The Label command sets label number <expr> to point to the beginning of the next line. <expr> can have any value between 1 and 9. You can then refer to this label number in future statements that need a label. Label numbers are reusable; the most recently encountered one is used, so the following is perfectly valid:

```
#Set R1 to 0
Set 0
#Use label 1 to mark the beginning of loop1
Label 1
#increment r1
Set r1 r1+1
#if less than 5 go back to beginning of loop 1.
if r1<5 goto 1
#Use the same label to mark the beginning of a new loop
Label 1
#decrement r1
Set r1 r1-1
#if greater than 0 goto beginning of loop2
if r1>0 goto 1
```

In general, a jump to a label goes to the last location at which the label number was used in a Label command. If there's no such location, successive Label commands are executed until one is found that matches; execution then continues from that point in the script. If there's no matching label at all then execution of the script stops, but no error is reported.

Label 0 is a special label which always points to the beginning of the current command line.

Load <str>

The Load command loads the file <str> into the window, just as if you'd dragged it to the window. So:

- a Text file gets 'typed' in at the terminal window's prompt
- a VTSetup file loads a saved setup into the terminal window
- a VTScript file is loaded and run immediately.

Prompt <str> [<expr>] [–case]
Prompt –off

The Prompt command makes VT220 wait between each line of a text import (ie the dragging of a text file onto a VT220 window) for the transmitting host to send the prompt <str>.

- If <expr> is present it gives a timeout value in seconds.
- The string comparison is case insensitive, unless the –case flag is present.
- The command Prompt –off stops VT220 waiting between each line.

OSCLI <str>

The OSCLI command passes <str> to the RISC OS command line interpreter.

Protocol <str> [–load]

The Protocol command selects the <str> communication protocol from within the command script. The necessary TIP module must already be loaded; if it is not then an error is reported, except:

- If the –load option is specified and the system variable protocol$dir is defined, the terminal emulator attempts to load the TIP module by running the application <protocol$dir>.!<str>. It then tries again to select the protocol, and if it's still not present reports an error.

Query <str> [<expr>]

The Query command opens a dialogue box on the screen with the text of <str>, and with Cancel, OK and close icons. It then waits for the user to click on one of the icons; if it's OK then register R<expr> becomes non-zero, otherwise the register is set to zero. If <expr> is missing R0 will be used.

Read [<expr1>] [<expr2>] [–conceal]

The Read command requests textual input from the user. The number of strings to read is specified by <expr2> and must be between 1 and 4; if you omit this parameter it defaults to 1. The command opens a dialogue box with the specified number of fields. The title of the first field is set to the value of S<expr1>, the second field to the value of S<expr1>+1, and so on. If <expr1> is missing it is taken to be 0. On exit register S<expr1> contains the value of the first field, S<expr1>+1 the value of the second, and so on.

- If the –conceal flag is present, '–' characters will be displayed as the user types into the last field of the dialogue box. This is particularly useful for entering passwords without displaying them. For example:

  ```
  SetStr 0 "Username:"
  SetStr 1 "Password:"
  Read 0 2 -conceal
  ```

  reads a user name and password into registers S0 and S1, concealing the password as it is the last field in the dialogue box.

- You must be careful to clear any registers holding sensitive information as soon as you've finished using them, as they're not secure. This particularly applies if you're storing a password in a register.

Send <str>
Transmit <str>

The Send and Transmit commands are identical, and simply send <str> to the remote host (or to the screen if the terminal is in local mode). If the terminal is not connected to a remote host and is not in local mode an error is reported and execution of the script is stopped.

Set <expr1> [<expr2>]

The Set command sets the value of register R<expr1> to the value of <expr2>. If <expr2> is missing, the register is set to zero. If <expr1> does not evaluate to a value between 0 and 9, an error is reported.

SetStr <expr> [<str>]

The SetStr command sets the value of register S<expr> to the value of <str>. If <str> is longer than 20 characters, the leftmost 20 characters are used. If <str> is missing, the register is set to the string "<UNDEFINED>". If <expr> does not evaluate to a value between 0 and 4, an error is reported.

Spool <str> | –Pause | –Continue | –Close

The Spool command controls the spooling of parts or all of your terminal session to file:

- <str> specifies a spool file to open and starts spooling to it.
- If the –Pause flag is present it pauses spooling to the file.
- If the –Continue flag is present it restarts spooling to the file. The text gets appended to what was already spooled.
- If the –Close flag is present it finishes spooling to the file and closes it.

Stop

The Stop command stops the execution of the current script. The window in which the script was running remains open, and the script remains loaded.

Title <str>

The Title command sets the title string which is displayed on the window's title bar, after the P=.

Transmit <str>

This is identical to **Send <str>** – see its entry for details.

Wait <str> [<expr>] [–timeout <expr1>] [–case]

The Wait command simply loops until the string <str> is received from the remote host (or typed at the keyboard if the terminal is in local mode).

- If <expr> is present it gives a timeout value in seconds.
- If the –timeout flag is present then on a timeout execution continues from the statement pointed to by label <expr1>. Otherwise execution continues from the next statement in the script.
- The string comparison is case insensitive, unless the –case flag is present.

An example script

Here is an example script. Note in particular the use of the Wait command to ensure that the host is in the right state to continue, and the use of the SetStr command to clear user names and passwords after they've been used:

```
Protocol telnet -load
Connect tp1
Wait "login:" 5
Input 0 "Username:" "Password:" -conceal
Send "<S0>|m"
Wait "Password:" 30
Send "<S1>|m"
SetStr 0
SetStr 1
Wait % 100
Send "tip pss|m"
Wait connected 100
Send |m
Wait Pad> 100
Send "call 23421920100401|m"
Wait > 100
Input 0 "TTTNS ID:" "TTNS Password:" -conceal
Send "ID <S0>|m"
Wait Password: 100
Send "<S1>|m"
Send "|m8|m"
SetStr 0
SetStr 1
Stop
```

The NumLock module

The NumLock module is an internal module used by the VT220 application to control the action of the *Num Lock* key on the keyboard. It provides one * Command – `*NumLock` – which is detailed on the next page.

*NumLock

Set up the NumLock and left Alt keys for use by VT220, or restore them

Syntax

`*NumLock [0|1]`

Parameters

0 or 1

Use

This command sets up the NumLock and left Alt keys for use by VT220, or restores them to their normal state.

`*NumLock 1` saves the state of the Num Lock key and makes it return ASCII 238; it also makes the left Alt key return ASCII 241.

`*Numlock 0` restores the state of the Num Lock key to that saved by the last *NumLock 1, and makes the NumLock and left Alt keys return their normal values.

Example

```
*NumLock 0
```

Protocols

Introduction

The Protocols application holds the protocol modules supplied with the TCP/IP Protocol Suite.

- For a more detailed explanation of protocol modules and their use in conjunction with the VT220 application, see the chapter entitled *VT220*.
- For more information about a given protocol, see the corresponding chapter.

What it does

The Protocols application is really only a glorified directory; if you double-click on its icon it opens a directory display showing the protocol modules.

The Protocols$Dir system variable

There's one other thing it does for you. When RISC OS first 'sees' the Protocols application (ie has to display its icon) it remembers where you've stored it. (It uses a system variable to do this.) Any application that uses protocol modules can then use this variable to find them. For instance VT220 uses this variable to provide the Load option on its Protocols submenu.

- The system variable is named `Protocols$Dir` . It gets set to the pathname of the protocol modules' directory.

Installing extra protocol modules

In the future you may get more protocol modules. Here's how you add them to the same directory as the other ones:

1. Open the `!Protocols` application directory by holding down the *Shift* key while you double-click on its icon.
2. Open the `Protocols` subdirectory that appears by double-clicking on its icon.
3. Drag the new protocol module into the directory display containing all the other protocol modules (such as Telnet, Ftp and Serial).

Telnet

Introduction

The Telnet application is a protocol module. It converts between the Telnet protocol and Acorn TIP (the standard protocol used by RISC OS terminal emulators). Hence you can connect any terminal emulator that uses the Acorn TIP (such as VT220) to any remote computer that supports Telnet.

- Like other protocol modules, the Telnet application is stored within the Protocols application. If you need to know more see the chapter in Part 1 entitled *Protocols*.
- For a general overview of protocol modules see the chapter in Part 1 entitled *VT220*.

The Telnet protocol module supports multiple concurrent Telnet sessions initiated from multiple terminal emulators.

1 Mbyte machines

VT220 with the Telnet protocol module will only just fit in a 1 Mbyte machine. You may, as an alternative, prefer to use the Telnet protocol from the command line with the *Telnet command. (For a full description see the chapter in Part 2 entitled *Internet*.)

Loading Telnet

To use the Telnet protocol module you need to load it. Double-click on the Protocols application to show the available protocols, then double-click on the Telnet icon. You can do so either before or after you load a terminal emulator. Nothing visible happens, but you can now select the Telnet protocol to use with any terminal emulator.

Alternatively, if RISC OS has 'seen' the Protocols application (ie had to display its icon), you can also open the Protocols directory display directly from a terminal emulator's menu. Again, see the *VT220* chapter for more details.

Opening a Telnet connection

Once you've selected the Telnet protocol you can then open a connection from your terminal emulator. Choose Open from the Connection submenu; Telnet provides a dialogue box to prompt you for the host's name. You can either give a host name or an Internet address specified in the dot notation.

Once the connection's open…

When the connection's opened, you should select either 'character at a time' or 'line by line' input mode. The default mode is character at a time, which is more appropriate for LAN (local area network) usage. Line by line mode may be more appropriate if you're connected to a host on a remote network, via a slower WAN (wide area network).

In character at a time mode, the text you type is immediately sent to the remote host for processing.

In line by line mode, all text is echoed locally, and (normally) only completed lines are sent to the remote host. You can use the local echo toggle to turn off and on the local echo. (The main use of this is so you can enter passwords without them being echoed.)

In either mode, if you've selected the Localchars option (the default in line mode – see below), your Quit, Intr and Flush characters (see *Set* below) are trapped locally, and sent as Telnet protocol sequences to the remote side. There are options which cause this action to flush subsequent output to the screen (until the remote host acknowledges the Telnet sequence) and to flush previous keyboard input (in the case of Quit and Intr).

Opening a connection with a script

Alternatively you can write a VT220 command script to load the Telnet protocol module and open a connection (using the command **Open <hostname>**). For full details see the chapter entitled *VT220 scripts*.

A terminal emulator's Telnet submenu

As we've already outlined above, you can alter how the Telnet protocol module is set up to work. Because you may want to have different Telnet connections set up in different ways, the Telnet protocol module doesn't provide its own main menu. Instead it provides a submenu that gets added to the menu for a terminal emulator window.

- The submenu only gets added once you've selected the Telnet protocol for that window.
- The settings you make in the submenu apply only to that window.

This next section details all the commands provided by the submenu:

Info

This entry leads to a dialogue box which displays version information about the Telnet protocol module.

Mode submenu

This entry leads to a further submenu from which you can select the forwarding mode. The remote host is asked for permission to go into the requested mode; if it's capable of doing so, the requested mode is entered:

- **Char** selects character at a time mode, and also turns off Localchars mode (see below).
- **Line** selects line by line mode, and also turns on Localchars mode (see below).

Send submenu

This leads to a further submenu which you can use to send a special character sequence to the remote host:

- **AO** sends the Telnet AO (Abort Output) sequence, which should cause the remote host to flush all its output to your screen.
- **AYT** sends the Telnet AYT (Are You There) sequence, to which the remote host may or may not choose to respond.
- **BREAK** sends the Telnet BRK (Break) sequence, which may have significance to the remote host.
- **EC** sends the Telnet EC (Erase Character) sequence, which should cause the remote host to erase the last character you typed.
- **EL** sends the Telnet EL (Erase Line) sequence, which should cause the remote host to erase the line currently being entered.
- **GA** sends the Telnet GA (Go Ahead) sequence.
- **IP** sends the Telnet IP (Interrupt Process) sequence, which should cause the remote host to abort the currently running process.
- **NOP** sends the Telnet NOP (No OPeration) sequence.
- **SYN** sends the Telnet SYNCH sequence. This sequence causes the remote host to discard all previously typed (but not yet read) input. This sequence is sent as TCP urgent data.

Set submenu

This leads to a further submenu which you can use to set which characters cause Telnet to take certain actions. For details of the different modes mentioned below see the sections entitled *Mode submenu* above and *Toggle submenu* below. For details of Telnet sequences see *Send submenu* above:

- Typing the **Echo** character when Telnet is in line by line mode toggles between local echoing of entered characters (for normal processing), and no echoing of entered characters (for entering, say, a password). The default is Control-E.
- Typing the **Eof** character as the first character on a line when Telnet is in line by line mode sends this character to the remote host. The default is Control-D.
- Typing the **Erase** character when Telnet is in both Localchars and character at a time modes sends a Telnet EC sequence to the remote host. The default is Control-H.
- Typing the **Flush** character when Telnet is in Localchars mode sends a Telnet AO sequence to the remote host. The default is Control-O.
- Typing the **Intr** character when Telnet is in Localchars mode sends a Telnet IP sequence to the remote host. The default is Control-C.
- Typing the **Kill** character when Telnet is in both Localchars and character at a time modes sends a Telnet EL sequence to the remote host. The default is Control-U .
- Typing the **Quit** character when Telnet is in Localchars mode sends a Telnet BRK sequence to the remote host. The default is Control-\.

Toggle submenu

This leads to a further submenu which you can use to toggle how Telnet responds to certain events:

- **Autoflush** is ignored unless Telnet is in Localchars mode (see below). If it is, and Autoflush is on (the default), then when you send an AO, BRK or IP control sequence Telnet won't display any data on your screen until the remote system acknowledges (via a Telnet Timing Mark option) that it has processed the sequence.

- **Autosynch** is also ignored unless Telnet is in Localchars mode (see below). If it is, and Autosynch is on (the default), then when you send a BRK or IP control sequence it is followed by a Telnet SYNCH sequence. This should cause the remote host to begin throwing away all previously typed input until it's read and acted on the Telnet sequence. By default Autosynch is off.
- **CR/LF** mode, when on, makes Telnet add a line feed character after every carriage return character it receives from the remote host. When it's off (the default) line feeds don't get added.

  This mode doesn't affect what you type, just what you receive. It's useful if the remote host sends carriage returns, but never line feeds.
- **Localchars** mode, when on, makes Telnet locally recognise Erase, Flush, Intr, Kill, and Quit characters, and transform them into appropriate Telnet control sequences (EC, AO, IP, EL, and BRK respectively). By default Localchars is on if Telnet is in line by line mode, and off if it's in character at a time mode.

Finding out more...

The chapter in Part 2 entitled *Internet* tells you:

- how to run the Internet application from the command line
- how to use the *Telnet command it provides you with.

Ftp

Introduction

The Ftp application is a protocol module. It converts between the Ftp protocol and Acorn TIP (the standard protocol used by RISC OS terminal emulators). Hence you can connect any terminal emulator that uses the Acorn TIP (such as VT220) to any remote computer that supports Ftp.

- Like other protocol modules, the Ftp application is stored within the Protocols application. If you need to know more see the chapter in Part 1 entitled *Protocols*.
- For a general overview of protocol modules see the chapter in Part 1 entitled *VT220*.

1 Mbyte machines

VT220 with the Ftp protocol module will not fit in a 1 Mbyte machine, so if you have such a machine you can't use the Ftp application. However, you can still transfer files from the command line using the *Tftp command. (For a full description see the chapter in Part 2 entitled *Internet*.)

You can, of course, also use the NFS Filer to transfer files. Provided the host machine you want to access supports NFS, you'll find this vastly preferable to using the *Tftp command.

Using Ftp

You can use the Ftp protocol module to logon to a Ftp server controlling a remote filestore, via a TCP connection. During the logon session you may interactively manipulate the remote filestore and transfer files between your local filesystem and the remote filestore using a terminal emulator window (such as VT220). The terminal emulator serves both as a window into the remote filestore, and to display protocol control and diagnostic information.

The Ftp protocol module supports multiple concurrent Ftp sessions initiated from multiple terminal emulators.

File mapping

Ftp maps RISC OS filenames and file types in the same way as RISC OS NFS does – both when you send files and when you fetch them.

If you use the NFS Filer to display files you've transferred to the remote filestore by Ftp, you'll generally not see any difference from files' RISC OS names and types. If you view the files using the remote filestore though, you'll notice some differences. In particular many filenames will have extensions that are used to store RISC OS file types. By default these take the form ',xxx' (where x is a hexadecimal digit), but you can reconfigure this to your own sceme. See the chapter entitled *File mapping* in Part 2 if you need more details.

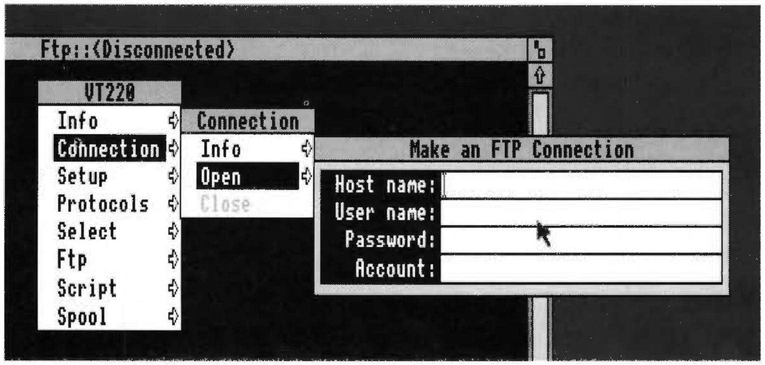
Loading Ftp

To use the Ftp protocol module you need to load it. Double-click on the Protocols application to show the available protocols, then double-click on the Ftp icon. You can do so either before or after you load a terminal emulator. Nothing visible happens, but you can now select the Ftp protocol to use with any terminal emulator.

Alternatively, if RISC OS has 'seen' the Protocols application (ie had to display its icon), you can also open the Protocols directory display directly from a terminal emulator's menu. Again, see the *VT220* chapter for more details.

Opening an Ftp connection

Once you've selected the Ftp protocol you can then open a connection from your terminal emulator. Choose Open from the Connection submenu; Ftp provides a dialogue box to prompt you for information to enable a TCP connection to be made to a remote Ftp server, and then to logon to the server:



Host name

This is the remote host specification. It may be either a host name or an Internet address specified in the dot notation.

User name

This is the user name required to logon to the remote Ftp server. You may skip this field if no user name is required by the remote server.

Password

This is the password required to logon to the remote Ftp server. Most remote Ftp servers will require that you have a non-null password set; in the rare cases where they don't you may skip this field.

Account

This is the account field required to access resources via the remote Ftp server. Again, you may skip this field.

Once you've logged on to the remote Ftp server, you may perform a sequence of file transfer or remote filestore manipulation commands.

Opening a connection with a script

Alternatively you can write a VT220 command script to load the Ftp protocol module and to then open a connection using the command:

Open <hostname>,<username>[,<password>[,<account>]]

If you omit a field it's assumed to be blank. For full details see the chapter entitled *VT220 scripts*.

Sending a file

You can send a single local file to the remote filestore by dragging a file icon onto the terminal emulator window. A dialogue box prompts you for further information:

Remote filename

This specifies the remote filename. It must be a string which uniquely identifies the destination on the remote filestore, in a form intelligible to the remote Ftp server. The filename may be a full pathname or else may be relative to the session current working directory (see below).

Append

Click on this to indicate that the local file is to be appended to the remote file if it's already present. Otherwise an existing remote file will be overwritten.

Sending multiple files

You can transfer multiple files by selecting them from within a RISC OS directory viewer, and then dragging the selection into the terminal emulator window. The files are sent in consecutive file transfer operations. No dialogue box is displayed for each transfer, which means that:

- The remote filename used is the same as the RISC OS one.
- The file gets written to the current working directory in the remote filestore.
- If the destination file already exists you'll overwrite it, unless you've selected the Ftp *Store unique* option (see below).

Since you can't change the filenames, you must ensure that all file and directory names in the local directory tree are also valid in the remote file store.

Sending a directory

You can also transfer a RISC OS directory by dragging a directory icon into the terminal emulator window. A corresponding remote directory tree is constructed (if necessary) by a sequence of Ftp *Make directory* operations, and is traversed by a sequence of Ftp *Change directory* operations. Files within the directory hierarchy are transfered in individual *Send file* operations. Again no dialogue box is displayed for each transfer, and so the above criteria for sending multiple files also apply here.

If any individual file transfer fails or is aborted, the remainder of the tree transfer is also aborted. This means that your current working directory on the remote filestore after a failure or abort may not be the same as it was at the start of the tree transfer.

Fetching files…

You can transfer files from the remote filestore to RISC OS by one of two mechanisms.

…by using Fetch file

The first is to choose Fetch file from Ftp's menu (see below). You'll get a Save dialogue box just like any other one in RISC OS.

…by selecting the filenames

Alternatively you may select a filename or list of filenames from text displayed on the terminal emulator window, typically just after you've done a *List names* or *List directory* operation. Then choose Get file from the terminal emulator's Select submenu.

If you've selected a list of names the files are transferred as a sequence of single fetch file operations, as described above. When you use this method you must take care that the sequence of characters selected on the terminal screen exactly matches one or more valid filenames in the current working directory on the remote filestore. Even superfluous white space may be interpreted by the remote server as being part of a filename.

A terminal emulator's Ftp submenu

The Ftp protocol module doesn't provide its own main menu. Instead it provides a submenu that gets added to the menu for a terminal emulator window.

- The submenu only gets added once you've selected the Ftp protocol for that window.
- The choices you make in the submenu apply only to that window.

Some dialogue boxes require you to type in a remote file or directory name. This should always be supplied in a form intelligible to the remote Ftp server, and may be relative to your session's current working directory.

This next section details all the commands provided by the submenu:

Info

This entry leads to a dialogue box which displays version information about the Ftp protocol module.

Verbose

Displays a record of each subsequent file transfer operation while it is in progress.

Trace

Toggles Display messages mode.

When Display messages mode is on, all control messages sent to the remote server and all control responses received from the remote Ftp server are displayed to you. When it's off (the default) they're not displayed.

Ascii

Sets the type for subsequent file transfers to be Ascii (the default).

Binary

Sets the type for subsequent file transfers to be Binary.

Store unique

Toggles storing of files on the remote machine under unique filenames.

This prevents multiple file transfers from overwriting existing files (see *Sending multiple files* above). The remote Ftp server must support the Ftp protocol STOU command for successful completion. This option is off by default.

SendPort	Toggles the use of Port commands. When Port commands are on (the default), Ftp will attempt to use a Port command when establishing a connection for each data transfer. The use of Port commands can prevent delays when performing multiple file transfers.
Fetch file	Fetches a file from the remote filestore. A dialogue box prompts you for the name of the file you want to fetch.
Full list	Gives a long listing of the contents of a remote directory. A dialogue box prompts you for the name of the remote directory you want to list. If you don't specify a directory, Ftp uses the current working directory on the remote machine. The resultant listing gives full information about each file.
Name list	Gives a short listing of the contents of a remote directory. A dialogue box prompts you for the name of the remote directory you want to list. If you don't specify a directory, Ftp uses the current working directory on the remote machine. The resultant listing typically gives only the names of the files in the directory.
Change dir	Changes the working directory on the remote filestore. A dialogue box prompts you for the new directory name.
ChDir parent	Changes directory up one level in the hierarchy (to the "parent" directory).
Current dir	Displays the name of the current working directory on the remote filestore.
Make dir	Makes a directory on the remote filestore. A dialogue box prompts you for the name of the remote directory you want to create.

Remove dir

Deletes a directory on the remote filestore.

A dialogue box prompts you for the name of the remote directory you want to delete.

Remove file

Deletes a file on the remote filestore.

A dialogue box prompts you for the name of the remote file to be deleted.

Rename file

Renames a file on the remote filestore.

A dialogue box prompts you for further information:

- **From:** supply the old filename.
- **To:** supply the new filename.

Quote

Sends an Ftp command verbatim.

A dialogue box prompts you for the command string to send.

The Inet$FtpOptions system variable

You can use the system variable `Inet$FtpOptions` to establish initial settings for some of the options available from Ftp's menu. A single character represents each of these options:

v	set Verbose option
t	set Trace option
b	set Binary transfer type instead of Ascii
p	unset SendPort option
u	set Store Unique option.

You should set the system variable from the command line or (if you want it to apply each time you start Ftp) from a boot file. For example to make Ftp start with the Verbose and Store Unique options set, you would use this command:

```
*Set Inet$FtpOptions vu
```

If the system variable is unset when Ftp starts then it uses its default values outlined in the rest of the chapter.

Finding out more...

The chapter in Part 2 entitled *Internet* tells you:

- how to run the Internet application from the command line
- how to use the *Tftp command it provides you with.

Serial

Introduction

The Serial application is a protocol module. It converts between data on the serial port and Acorn TIP (the standard protocol used by RISC OS terminal emulators). Hence you can connect any terminal emulator that uses the Acorn TIP (such as VT220) to any remote computer that has a serial port.

- Like other protocol modules, the Serial application is stored within the Protocols application. If you need to know more see the chapter in Part 1 entitled *Protocols*.
- For a general overview of protocol modules see the chapter in Part 1 entitled *VT220*.

The serial protocol module only supports a single session from one terminal emulator, because your computer only has a single serial port to connect to a remote computer.

Loading Serial

To use the Serial protocol module you need to load it. Double-click on the Protocols application to show the available protocols, then double-click on the Serial icon. You can do so either before or after you load a terminal emulator. Nothing visible happens, but you can now select the Serial protocol to use with any terminal emulator.

Alternatively, if RISC OS has 'seen' the Protocols application (ie had to display its icon), you can also open the Protocols directory display directly from a terminal emulator's menu. Again, see the *VT220* chapter for more details.

Opening a Serial connection

The Serial protocol doesn't need any information to open a connection, so it does so automatically as soon as you've selected it. With the VT220 application, you do so using the Protocols submenu.

with a script

protocol module and hence to automatically open a connection. For full details see the chapter entitled *VT220 scripts*.

A terminal emulator's Serial submenu

The Serial protocol module doesn't provide its own main menu. Instead it provides a submenu that gets added to the menu for a terminal emulator window.

- The submenu only gets added once you've selected the Serial protocol for that window.
- The choices you make in the submenu apply only to that window.

This next section details all the commands provided by the submenu:

Word length submenu

Use this submenu to choose the word length and parity used for serial communication.

Stop bits submenu

Use this submenu to choose the number of stop bits used for serial communication.

Rx speed submenu

Use this submenu to choose the receive speed for serial communication.

Tx speed submenu

Use this submenu to choose the transmit speed for serial communication.

There is also a 'Tx=Rx' option, which makes the transmit speed the same as the receive speed, so the transmit speed changes every time you change the receive speed. By default this is on, and so the default transmit speed is the same as the default (ie current) receive speed.

Flow control submenu

Use this submenu to set the flow control used for serial communication to either Xon/Xoff or RTS/CTS.

Defaults

With the exception of the Tx speed (see above), all the settings default to the current state of the computer. You can configure he word length and stop bits using `*Configure Data`, and the Rx speed using `*Configure Baud`. Flow control defaults to RTS/CTS on a reset. However, there are various commands which can alter the initial state of the computer, and some applications may use these commands without your realising (particularly any that use the serial port).

The Serial$Options system variable

You can use the system variable Serial$Options to establish initial settings for the options available from Serial's menu. The syntax is a series of flag value pairs:

```
[-TX <baud rate>] [-RX <baud rate>] [-WL 7|8] [-P N|E|O|M|S] [-SB 1|2] [-XON|-CTS]
```

Flag	Sets	Values	Meaning
-TX	Tx speed	<number>	baud rate
-RX	Rx speed	<number>	baud rate
-WL	Word length	7	7 bits/word
		8	8 bits/word
-P	Parity	N	none
		E	even
		O	odd
		M	mark
		S	space
-SB	Stop bits	1	1 stop bit
		2	2 stop bits
-XON	Flow control	—	use Xon/Xoff
-CTS	Flow control	—	use CTS/RTS

If the system variable is unset when Serial starts then it uses the default values outlined above; similarly if you omit a flag value pair, Serial uses the default.

You should set the system variable from the command line or (if you want it to apply each time you start Serial) from a boot file. For example to make Serial start with a 9600 baud Tx speed, a 1200 baud Rx speed, 8 bit words, no parity, 1 stop bit and CTS/RTS flow control, you would use this command:

```
*Set Serial$Options -TX 9600 -RX 1200 -WL 8 -P N -SB 1 -CTS
```

Finding out more...

There's no further information about the Serial protocol in Part 2 of this Guide.

Part 2 – Internet and NFS

Internet

Introduction

In the earlier chapter on *Internet* in Part 1 of this manual we saw that the Internet application provides the services and protocols needed for other TCP/IP Protocol Suite applications to work. It also provides a variety of modules and * Commands that duplicate UNIX commands. If you're a more experienced Internet user you may well find these useful; this chapter gives you more information about them.

Running the Internet application

To use any of these commands you need to run the Internet application. There are three ways you can do this:

- double-clicking on the Internet icon from the desktop
- typing a command at the command line
- including a command in a boot file.

In both the latter cases you must use a command of the form:

```
*Run <Internet pathname>.!Internet.!Run
```

where *<Internet pathname>* is the rest of the pathname to the Internet application. You **must** start the command with `*Run`; if you don't RISC OS won't know which filing system holds the Internet application.

Getting a command line

If you're using the desktop, there are three ways of getting a command line so you can enter * Commands:

- Open an Edit task window using Edit's icon bar menu.
- Press *F12* to temporarily leave the desktop.
- Press *Ctrl-Shift-F12* to permanently leave the desktop.

* Commands

stored as the `!RunImage` file within the application. It provides three * Commands:

`*InetConfig`	Increase Internet module storage allocation
`*InetGateway`	Toggle IP packet forwarding
`*InetInfo`	Display Internet module internal statistics.

Running the Internet application loads the Internet module; you can then use the above * Commands.

The absolute programs

There are also a set of commands that are provided as Absolute files (ie each file is a program that performs one command) in the `bin` subdirectory of the Internet application. These are:

`*ARP`	Address resolution display and control
`*IfConfig`	Configure network interface parameters
`*IfRConfig`	Configure network interface parameters from a remote server
*Ping	Send ICMP ECHO_REQUEST packets to network hosts
`*Route`	Manually manipulate the routing tables
`*Telnet`	User interface to the Telnet protocol
`*Tftp`	Trivial file transfer program.

Running the Internet application adds the `bin` subdirectory to the system variable `Run$Path`. RISC OS then knows where to find the above * Commands, so you can use them directly from the command line.

Although the above are actually programs, in the section that follows we've treated them as * Commands because you're most likely to use them in just the same way.

The RouteD * Commands

Finally, the Internet application also contains the `RouteD` module in its `rm` subdirectory. It provides three * Commands:

`*RouteD`	Start the RouteD module
`*RouteDTraceOff`	Turn off tracing by the RouteD module
`*RouteDTraceOn`	Turn on tracing by the RouteD module.

To use these * Commands you need to load the RouteD module. By default the Internet application doesn't do so when you run it – but you can configure it so it does do so. See your *Installation Guide* for details, or ask your system administrator.

*ARP

Address resolution display and control

Syntax

```
*ARP <host>
*ARP -a
*ARP -d <host>
*ARP -s <host> <phys_addr> [temp] [pub]
*ARP -f <filename>
```

Parameters

`<host>`	An Internet host specified either by name (which must be present in the host name data base `<InetDBase$Path>hosts`) or by address (using the standard Internet dot notation).
`<phys_addr>`	The physical address of `<host>` given in Ethernet format (ie six hexadecimal bytes separated by colons).
`<filename>`	The full pathname of a file containing multiple entries to be set in the ARP table.

Use

The ARP program displays and modifies the Internet-to-Physical-address translation tables located in the Internet module and used by the address resolution protocol ARP.

With no flags, the program displays the current ARP entry for `<host>`.

The -a flag makes the program display all the ARP entries currently in its table.

The -d flag makes the program delete an entry for `<host>`.

The -s flag makes the program create an ARP entry for the host called `<host>` with the physical address `<phys_addr>`.

- The entry will be permanent unless the word `temp` is given in the command.
- The entry will be 'published' if the word `pub` is given. This system will then act as an ARP or Reverse ARP server, responding to requests for `<host>`'s physical address even though the host address is not its own.

The -f flag causes the file `<filename>` to be read and multiple entries to be set in the ARP tables. Entries in the file should be of the form:

```
<host> <phys_addr> [temp] [pub]
```

with argument meanings as given above.

If you don't know the physical address of an interface, you can use the *InetInfo command to find it.

- For your reference, the physical address of an Econet interface takes the form:

  `00.00.00.00.`<station number>`.`<net number>

Examples

```
*ARP tp1
*ARP -a
*ARP -d 01.01.01.01
*ARP -s tp1 01.01.01.01.01.01 temp
*ARP -f adfs::HardDisc.$.Internet.ARP_Table
```

Related commands

None

*IfConfig

Configure network interface parameters

Syntax

```
*IfConfig [-e] "<interface>" [<host> [<parameters>]]
```

Parameters

`<interface>`	The name (either `et` for Ethernet, or `ec` for Econet) and unit number (starting from 0 for the first one fitted) of an interface – viz `et0` (Ethernet unit 0), `ec0` (Econet unit 0).
`<host>`	An Internet host specified either by name (which must be present in the host name data base `<InetDBase$Path>hosts`) or by address (using the standard Internet dot notation).

Use

The IfConfig program is used to assign an Internet address to a network interface supported by the Internet module, and/or to configure network interface parameters. It must be used at Internet module startup time to define the network address of each interface present on a machine; it may also be used at a later time to redefine an interface's address or other operating parameters.

You can set the following `<parameters>` with the IfConfig program:

`up`	Mark an interface 'up'. This may be used to enable an interface after going `*IfConfig ... down`. It happens automatically when setting the first address on an interface. If the interface was reset when previously marked down, the hardware will be re-initialised.
`down`	Mark an interface 'down'. When an interface is marked 'down', the system will not attempt to transmit messages through that interface. If possible, the interface will be reset to disable reception as well. This action does not automatically disable routes using the interface.

`arp`	Enable the use of the Address Resolution Protocol in mapping between network level addresses and link level addresses (default), for mapping between Internet addresses and Ethernet addresses.
`-arp`	Disable the use of the Address Resolution Protocol.
`metric <n>`	Set the routing metric of the interface to `<n>` (default 0). The routing metric is used by the routing protocol (RouteD module). Higher metrics have the effect of making a route less favourable; metrics are counted as additional hops to the destination network or host.
`netmask <mask>`	Specify how much of the address to reserve for subdividing networks into sub-networks. The mask includes the network part of the local address and the subnet part, which is taken from the host field of the address. The mask can be specified as a single hexadecimal number with a leading 0x, as an Internet address using the standard dot notation, or as a pseudo-network name listed in the network table <Inet$DBase>networks. The mask contains 1's for the bit positions in the 32-bit address which are to be used for the network and subnet parts, and 0's for the host part. The mask should contain at least the standard network portion, and the subnet field should be contiguous with the network portion.
`broadcast`	Specify the address to use to represent broadcasts to the network. The default broadcast address is the address with a host part of all 1's.

If you don't use any of the optional parameters, the IfConfig program displays the current configuration for a network interface.

The -e option makes IfConfig place any fatal error report string in the system variable Inet$Error instead of writing it to the standard output.

Example

```
*IfConfig -e "ec0" tp1 up
```

Related commands

*IfRConfig

Configure network interface parameters from a remote server

Syntax

```
*IfRConfig [-e] "<interface>" [revarp] [netmask]
```

Parameters

`<interface>`	The name (either `et` for Ethernet, or `ec` for Econet) and unit number (starting from 0 for the first one fitted) of an interface – viz `et0` (Ethernet unit 0), `ec0` (Econet unit 0).

Use

The IfRConfig program is used to assign an Internet address to a physical network interface. It does so using a Reverse ARP broadcast/response transaction with a remote server host, and/or (optionally) a netmask via an ICMP MASKREQ broadcast/response transaction. If it does not find a remote server, it generates an error.

The optional parameter `revarp` requests an interface address; `netmask` requests a netmask.

If you don't use any of the optional parameters, the IfRConfig program displays the current configuration for a network interface.

The -e option makes IfRConfig place any fatal error report string in the system variable Inet$Error instead of writing it to the standard output.

Example

```
*IfRConfig "et0" revarp
```

Related commands

*IfConfig

*InetConfig

Increase Internet module storage allocation

Syntax

`*InetConfig <mbufs>`

Parameters

`<mbufs>`	Number of 'mbufs' (each 128 bytes long) to claim from RMA RAM at startup time.

Use

*InetConfig may be used to increase the amount of RMA RAM obtained by the Internet module at startup time for internal data buffering. You might wish to do so, for example, if the host machine is to be dedicated to a gateway function. Data is buffered in objects called "mbufs", each of which is 128 bytes in size. The amount of memory initially obtained by Internet is 32 kbytes, providing a pool of 256 mbufs.

Example

`*InetConfig 512`	doubles the pool size at startup to 512 mbufs by obtaining an extra 32 kbytes of RMA RAM.

Related commands

None

*InetGateway

Toggle IP packet forwarding

Syntax

```
*InetGateway 1|0
```

Use

*InetGateway may be used to enable (*InetGateway 1) or to disable (*InetGateway 0) IP layer packet forwarding (ie gateway operation) if multiple network interfaces are present.

The default state is off.

Example

```
*InetGateway 1
```

Related commands

None

*InetInfo

Display Internet module internal statistics

Syntax

```
*InetInfo [r][p][i]
```

Use

*InetInfo displays detailed information about Internet module activity. By default it only gives details of physical interface activity, including the physical addresses of all interfaces fitted to the computer. You can also request details of internal resource usage and protocol activity using the options:

r	display only internal resource information
p	display only protocol information
i	display only interface information (the default)

Most of the information displayed is runic in nature. It is presented mainly as an aid to trouble-shooting, should you require it.

Example

```
*InetInfo r
```

Related commands

None

*Ping

Send ICMP ECHO_REQUEST packets to network hosts

Syntax

`*Ping [-r] [-v] <host> [<packetsize>] [<count>]`

Parameters

`<host>`	An Internet host specified either by name (which must be present in the host name data base `<InetDBase$Path>hosts`) or by address (using the standard Internet dot notation).
`<packetsize>`	Size of packet to send (default is 64 bytes).
`<count>`	Number of packets to send (default is 1).

Use

An Internet can be a large and complex aggregation of network hardware, connected together by gateways. Tracking a single-point hardware or software failure can be difficult. The Ping program utilises the ICMP protocol's mandatory ECHO_REQUEST datagram to elicit an ICMP ECHO_RESPONSE from a host or gateway. ECHO_REQUEST datagrams (or *pings*) have an IP and ICMP header, and then an arbitrary number of padding bytes used to fill out the packet.

The optional `-r` parameter makes the Ping program bypass the normal routing tables and send directly to a host on an attached network. (If the host is not on a directly attached network an error is returned.) You can use this option to 'ping' a local host through an interface that has no route through it.

The optional -v parameter causes verbose output: any ICMP packets other than ECHO RESPONSE that are received are listed.

When using Ping for fault isolation you should first run it on your local host, to verify that your local network interface is up and running. Then you should 'ping' hosts and gateways that are further and further away.

If you don't give a `<count>` the Ping program sends just one datagram and expects a single ECHO_RESPONSE to be returned. If you give a `<count>` Ping sends one datagram per second up to that number, and prints one line of output for every ECHO_RESPONSE returned. No output is produced if there is no response.

Packet loss statistics are computed and a brief summary is displayed when all responses have been received, or when the program times out, or when you terminate the program.

RISC OS Ping messages are not time-stamped.

Example

```
*Ping -v tp1 1024 20
```

Related commands

None

*Route

Manually manipulate the routing tables

Syntax

```
*route [-e] [-f] add [net|host] <destination> <gateway> <metric>
*route [-e] [-f] delete [net|host] <destination> <gateway>
```

Parameters

`<destination>`	An Internet host or network specified either by name (which must be present in the respective host or network name data base `<InetDBase$Path>`...) or by address (using the standard Internet dot notation).
`<gateway>`	The next-hop gateway to which packets should be addressed.
`<metric>`	A count giving the number of hops to the destination.

Use

You can use the Route program to manually manipulate the Internet module's network routing tables.

The `<metric>` must be zero if the destination is on a directly-attached network, and non-zero if the route utilises one or more gateways. If you're adding a route with `<metric>` 0, the `<gateway>` given is the address of this host on the common network, indicating the interface to be used for transmission.

Routes to a particular host are distinguished from those to a network by interpreting the Internet address associated with `<destination>`. The optional keywords `net` and `host` force the `<destination>` to be interpreted as a network or a host, respectively. Otherwise, if the `<destination>` has a local address part of 0 or if it's the symbolic name of a network, then the route's presumed to be to a network; else the route's presumed to be to a host. All symbolic names specified for a destination or gateway are looked up first as a host name; if this fails, the name is then looked up as a network name.

The -e option makes Route place any fatal error report string in the system variable Inet$Error instead of writing it to the standard output.

The -f option makes Route delete all gateway entries in the Internet module's routing tables. If this is used in conjunction with an add command, the tables are flushed first.

Examples

```
*route -f add net 1 0.0.0.1 1
*route -e delete host tp1 tp4
```

Related commands

*RouteD

*RouteD

Start the RouteD module

Syntax

```
*RouteD
```

Use

RouteD may be invoked to manage the network routing tables within the Internet module.

When RouteD is started, it finds those directly connected interfaces configured into the system and marked 'up'. If multiple interfaces are present, it is assumed that the host will forward packets between networks. RouteD then broadcasts a request packet on each interface and subsequently listens continuously for request and response packets from other hosts. Notification of the arrival of these packets is performed in the background, via RISC OS events.

When a request packet is received, RouteD formulates a reply based on the information maintained in its internal tables. The response packet generated contains a list of known routes, each marked with a hop count metric (a count of 16, or greater, is considered 'infinite'). The metric associated with each route returned provides a metric relative to the sender.

Response packets received by RouteD are used to update the routing tables if one of the following conditions is satisfied:

- No routing table entry exists for the destination network or host, and the metric indicates the destination is reachable (ie the hop count is not infinite).
- The source host of the packet is the same as the router in the existing routing table entry. That is, updated information is being received from the very internetwork router through which packets for the destination are being routed.
- The existing entry in the routing table has not been updated for some time (defined to be 90 seconds) and the route is at least as cost-effective as the current route.
- The new route describes a shorter route to the destination than the one currently stored in the routing tables. (The metric of the new route is compared against the one stored in the table to decide this.)

When an update is applied, RouteD records the change in its internal tables and updates the Internet module routing table. The change is reflected in the next response packet sent.

In addition to processing incoming packets, RouteD also periodically checks the routing table entries. If an entry has not been updated for 3 minutes, the entry's metric is set to infinity and marked for deletion. Deletions are delayed an additional 60 seconds to insure the invalidation is propagated throughout the local internet.

Hosts acting as internetwork routers gratuitously broadcast their routing tables every 30 seconds to all directly connected networks. The normal routing tables are bypassed when sending gratuitous responses. The reception of responses on each network is used to determine that the network and interface are functioning correctly. If no response is received on an interface, another route may be chosen to route around the interface, or the route may be dropped if no alternative is available.

The Inet$RouteDOptions system variable

Routed supports several option flags, recorded in <Inet$RouteDOptions>:

g	This flag is used on internetwork routers to offer a route to the 'default' destination. This is typically used on a gateway to the Internet, or on a gateway that uses another routing protocol whose routes are not reported to other local routers.
s	Supplying this option forces RouteD to supply routing information whether it is acting as an internetwork router or not. This is the default if multiple network interfaces are present.
q	This is the opposite of the s option.

You should set this system variable with a line having the syntax:

```
*Set Inet$RouteDOptions [g][s|q]
```

Passive and active gateways

In addition to the facilities described above, RouteD supports the notion of 'distant' passive and active gateways. When RouteD is started up, it reads the file <InetDBase$Path>gateways (if present) to find information about such gateways. Gateways specified in this manner should be marked passive if they are not expected to exchange routing information, while gateways marked active should be willing to exchange routing information (ie they should have a RouteD process running on the machine). Passive gateways are maintained in the routing tables forever and information regarding their existence is included in any routing information transmitted. Active gateways are treated equally to network interfaces. Routing information is distributed to the gateway and if no routing information is received for a period of time, the associated route is deleted. External gateways are also passive, but are not placed in the routing table nor are they included in routing updates. The function of external entries is to inform RouteD that another routing process will install such a route, and that alternate routes to that destination should not be installed. Such entries are only required when both routers may learn of routes to the same destination.

The format of the gateways file

The <Inet$DBase>gateways file is comprised of a series of lines, each in the following format:

```
net|host <destination> gateway <gateway> metric <metric> passive|active|external
```

- The `net` or `host` keyword indicates if the route is to a network or specific host.
- `The <destination>` parameter gives an Internet host or network specified either by name (which must be present in the respective host or network name data base `<InetDBase$Path>`...) or by address (using the standard Internet dot notation).
- The `<gateway>` parameter gives the name or address of the next-hop gateway to which packets should be forwarded.
- The `<metric>` parameter is a count giving the number of hops to the destination.
- One of the keywords `passive`, `active` or `external` indicates if the gateway should be treated as passive or active (as described above), or whether the gateway is external to the scope of the RouteD protocol.

Example

```
*RouteD
```

Related commands

*Route, *RouteDTraceOn, *RouteDTraceOff

*RouteDTraceOff

Turn off tracing by the RouteD module

Syntax

```
*RoutedTraceOff
```

Use

*RoutedTraceOff turns off tracing by the RouteD module.

Example

```
*RouteDTraceOff
```

Related commands

*RouteD, *RouteDTraceOn

*RouteDTraceOn

Turn on tracing by the RouteD module

Syntax

```
*RoutedTraceOn [<logging level>] [<filename>]
```

Parameters

`<trace level>`	Trace level (1-4) of trace required, where 1 is the lowest level and 4 the highest.
`<filename>`	Name of file to output trace to.

Use

*RoutedTraceOn turns on tracing by the RouteD module. The output may be written either on the standard output or into the given logging file (which remains open until *RoutedTraceOff is invoked).

Most of the information displayed is runic in nature. It is presented mainly as an aid to trouble-shooting, should you require it.

Example

```
*RouteDTraceOn 2 adfs::HardDisc.$.RDTraceOut
```

Related commands

*RouteD, *RouteDTraceOff

*Telnet

User interface to the Telnet protocol

Syntax

```
*Telnet [<host> [<port>]]
```

Parameters

`<host>`	An Internet host specified either by name (which must be present in the host name data base `<InetDBase$Path>hosts`) or by address (using the standard Internet dot notation).
`<port>`	A valid port number.

Use

The Telnet program is used to communicate with another host using the Telnet protocol. If you invoke Telnet without arguments, it enters command mode, indicated by its prompt `telnet>` . In this mode, it accepts and executes the commands listed below. If you invoke it with arguments, it performs an open command (see below) with those arguments.

When the connection's opened, you should select either 'character at a time' or 'line by line' input mode. The default mode is character at a time, which is more appropriate for LAN (local area network) usage. Line by line mode may be more appropriate if you're connected to a host on a remote network, via a slower WAN (wide area network).

In character at a time mode, the text you type is immediately sent to the remote host for processing.

In line by line mode, all text is echoed locally, and (normally) only completed lines are sent to the remote host. You can use the local echo character (initially '^E') to turn off and on the local echo. (The main use of this is so you can enter passwords without them being echoed.)

In either mode, if you've selected the Localchars option (the default in line mode – see below), your Quit, Intr and Flush characters (see *set* below) are trapped locally, and sent as Telnet protocol sequences to the remote side. There are options which cause this action to flush subsequent output to the screen (until the remote host acknowledges the Telnet sequence) and to flush previous keyboard input (in the case of Quit and Intr).

While connected to a remote host you can enter Telnet command mode by typing the Telnet 'escape character' (initially *Control-]*). When you're in command mode your RISC OS computer will behave just as it normally does; so you can, for instance, use the *Copy* key to copy characters off the screen, or the function keys to 'type' stored strings.

Commands

You can use the following commands. You need only type enough of each command to uniquely identify it. (This is also true for arguments to the mode, set, toggle, and display commands.)

open <host> [<port>]

Open a connection to the named host. If you don't give a port number, Telnet will attempt to contact a Telnet server at the default port. The host specification may be either a host name, or an Internet address specified using the standard dot notation.

close

Close a Telnet session and return to command mode.

quit

Close any open Telnet session and exit Telnet.

? [<command>]

Get help. With no arguments, Telnet prints a help summary. If a command is specified, Telnet will print the help information for just that command.

status

Show the current status of Telnet. This includes the peer you're connected to, as well as the current mode.

display [<arguments...>]

Displays all, or some, of the set and toggle values (see below). You can use any valid *set* or *toggle* argument with this command.

mode <type>

Selects the forwarding mode to be <type>. The remote host is asked for permission to go into the requested mode; if it's capable of doing so, the requested mode is entered:

- **character** selects character at a time mode, and also turns off localchars mode (see below).
- **line** selects line by line mode, and also turns on localchars mode (see below).
- **?** displays help information for the *mode* command.

send <arguments...>

Sends one or more special character sequences to the remote host. These are the arguments you can specify:

- **ao** sends the Telnet AO (Abort Output) sequence, which should cause the remote host to flush all its output to your screen.
- **ayt** sends the Telnet AYT (Are You There) sequence, to which the remote host may or may not choose to respond.
- **brk** sends the Telnet BRK (Break) sequence, which may have significance to the remote host.
- **ec** sends the Telnet EC (Erase Character) sequence, which should cause the remote host to erase the last character you typed.
- **el** sends the Telnet EL (Erase Line) sequence, which should cause the remote host to erase the line currently being entered.
- **escape** sends the current Telnet escape character (initially '^]').
- **ga** sends the Telnet GA (Go Ahead) sequence.
- **ip** sends the Telnet IP (Interrupt Process) sequence, which should cause the remote host to abort the currently running process.
- **nop** sends the Telnet NOP (No OPeration) sequence.
- **synch** sends the Telnet SYNCH sequence. This sequence causes the remote host to discard all previously typed (but not yet read) input. This sequence is sent as TCP urgent data.
- **?** displays help information for the *send* command.

Sets which characters cause Telnet to take certain actions. For details of the different modes mentioned below see the *mode* command above and the *toggle* command below. For details of Telnet sequences see the *send* command above. Valid arguments are in **bold type**:

- Typing the **echo** character when Telnet is in line by line mode toggles between local echoing of entered characters (for normal processing), and no echoing of entered characters (for entering, say, a password). The default is Control-E.
- Typing the **escape** character enters Telnet command mode. The default is Control-].
- Typing the **eof** character as the first character on a line when Telnet is in line by line mode sends this character to the remote host. The default is Control-D.
- Typing the **erase** character when Telnet is in both Localchars and character at a time modes sends a Telnet EC sequence to the remote host. The default is Control-H.
- Typing the **flushoutput** character when Telnet is in Localchars mode sends a Telnet AO sequence to the remote host. The default is Control-O.
- Typing the **interrupt** character when Telnet is in Localchars mode sends a Telnet IP sequence to the remote host. The default is Control-C.
- Typing the **kill** character when Telnet is in both Localchars and character at a time modes sends a Telnet EL sequence to the remote host. The default is Control-U .
- Typing the **quit** character when Telnet is in Localchars mode sends a Telnet BRK sequence to the remote host. The default is Control-\.
- **?** displays help information for the *set* command.

toggle <arguments...>

Toggle how Telnet responds to certain events. You can see the current state of these modes using the *display* command. You can specify one or more of the arguments in **bold type** below:

- **autoflush** is ignored unless Telnet is in localchars mode (see below). If it is, and autoflush is on (the default), then when you send an AO, BRK or IP control sequence Telnet won't display any data on your screen until the remote system acknowledges (via a Telnet Timing Mark option) that it has processed the sequence.
- **autosynch** is also ignored unless Telnet is in localchars mode (see below). If it is, and autosynch is on (the default), then when you send a BRK or IP control sequence it is followed by a Telnet SYNCH sequence. This should cause the remote host to begin throwing away all previously typed input until it's read and acted on the Telnet sequence. By default autosynch is off.
- **crmod** mode, when on, makes Telnet add a line feed character after every carriage return character it receives from the remote host. When it's off (the default) line feeds don't get added.

  This mode doesn't affect what you type, just what you receive. It's useful if the remote host sends carriage returns, but never line feeds.
- **localchars** mode, when on, makes Telnet locally recognise erase, flushoutput, interrupt, kill and quit characters, and transform them into appropriate Telnet control sequences (EC, AO, IP, EL and BRK respectively). By default localchars is on if Telnet is in line by line mode, and off if it's in character at a time mode.
- **netdata** mode, when on, displays all network data in hexadecimal format. By default netdata is off.
- **?** displays help information for the *toggle* command.

Termcap entries

The file `$.unixhost.termcap` on the *Applications* distribution disc contains entries for a UNIX `/etc/termcap` file that cater for different RISC OS screen modes. The Telnet program selects an appropriate terminal type from them. You should add these entries to the Telnet host's termcap file if you can.

Example

```
*Telnet tp1
```

Related commands

None

*Tftp

Trivial file transfer program

Syntax

`*Tftp [<host>]`

Parameters

`<host>`	An Internet host specified either by name (which must be present in the host name data base `<InetDBase$Path>hosts`) or by address (using the standard Internet dot notation).

Use

The Tftp program is the user interface to the Internet TFTP (Trivial File Transfer Protocol), which allows you to transfer files to and from a remote machine. You can specify the remote host on the command line, in which case Tftp uses it as the default host for future transfers (see the *connect* command below).

Because there is no user-login or validation within the TFTP protocol, the remote site will probably have some sort of file-access restrictions in place. The exact methods are specific to each site.

Commands

Once Tftp is running, it issues the prompt `tftp>` and recognizes the following commands:

connect <host> [<port>]

Sets the host (and optionally port) for transfers. Note that the TFTP protocol, unlike the FTP protocol, does not maintain connections betweeen transfers; thus, the *connect* command does not actually create a connection, but merely remembers what host is to be used for transfers. You do not have to use the *connect* command; you can instead specify the remote host as part of the *get* or *put* commands.

mode <transfer-mode>

Sets the mode for transfers; transfer-mode may be one of **ascii** or **binary**. The default is ascii.

put <localname> <remotename>

Puts a file to the specified remote file. The destination can be in one of two forms: a filename on the remote host (if you've already specified the host), or a string of the form <host>:<filename> to specify both a host and filename at the same time. If you use the latter form, the hostname you specify becomes the default for future transfers.

get <remotename> <localname>

Gets a file from the specified sources. The source can be in one of two forms: a filename on the remote host, (if you've already specified the host), or a string of the form <host>:<filename> to specify both a host and filename at the same time. If the latter form is used, the last hostname specified becomes the default for future transfers.

quit

Exits Tftp.

verbose

Toggles verbose mode.

trace

Toggles packet tracing.

status

Shows Tftp's current status.

rexmt <retransmission-timeout>

Sets the per-packet retransmission timeout, in seconds.

timeout <total-transmission-timeout>

Sets the total transmission timeout, in seconds.

ascii

Shorthand for 'mode ascii'.

binary

Shorthand for ‘mode binary’.

? [<command-names...>]

Prints help information.

Example

```
*Tftp tp1
```

Related commands

None

NFS

Using NFS from the command line

This chapter tells you how to use NFS from the command line.

To do so you need to:

- run the Internet application's `!Run` file
- load the NFS module.

There are three ways you can do this:

- double-clicking on the NFS Filer icon from the desktop
- typing two commands at the command line
- including two commands in a boot file.

In the former case you also load the NFS Filer application that provides the desktop interface to NFS. If you only want to use the * Command interface to NFS, choose Quit from the NFS Filer's icon bar menu. The NFS module will remain loaded; you can still use * Commands, but can no longer use the desktop interface. If you subsequently restart the NFS Filer its desktop interface will reflect any changes you've made at the command line – including mounts and dismounts.

In both the latter cases you must use commands of the form:

```
*Run <Internet pathname>.!Internet.!Run
*RMLoad <NFS pathname>.!NFSFiler.rm.NFS
```

where *<Internet pathname>* and *<NFS pathname>* are the rest of the pathnames to the Internet and NFS Filer applications. (If you've put Internet and the NFS Filer in the same directory the *<pathnames>* will be the same.) You **must** start the first command with `*Run`; if you don't RISC OS won't know which filing system holds the Internet application.

you can enter * Commands:

- Open an Edit task window using Edit's icon bar menu.
- Press *F12* to temporarily leave the desktop.
- Press *Ctrl-Shift-F12* to permanently leave the desktop.

Standard * Commands provided by NFS

A full set of * Commands is available for you to use with the NFS filing system. You can use all the * Commands described in the *RISC OS User Guide* chapter entitled *Filing system commands*. Furthermore, these commands operate as do their equivalents in other filing systems:

- *Back
- *Bye
- *Free
- *NoDir
- *NoLib
- *NoUrd
- *URD

For details of these commands, see the chapter entitled *ADFS-specific commands* in your *RISC OS User Guide*.

Other * Commands provided by NFS

The NFS filing system also provides other * Commands which we've documented in the following pages because:

- some have the same name as * Commands used in other filing systems, but are different in function and/or in the parameters they take (*Dismount, *Logon, *Mount)
- others are new * Commands unique to the NFS filing system (*CacheTime, *NFS, *TimeOffset).

*CacheTime

Sets the time for which cached information is assumed to be valid

Syntax

```
*CacheTime <time>
```

Parameters

`<time>`	in 1/100ths of a second.

Use

This sets the time for which any information cached in the future is assumed to be valid. It doesn't affect any information already cached, which will remain valid for the <time> that was set when it was originally cached.

The units are 1/100ths of a second. The default <time> is 60000 – which corresponds to ten minutes.

Example

```
*CacheTime 100000
```

Related commands

None

*Dismount

Dismounts NFS mount points

Syntax

```
*Dismount [-Host <hostname>|<mountname>]
```

Parameters

`<hostname>`	The name of an NFS host having mount points.
`<mountname>`	The name of an NFS mount point.

Use

Dismounts NFS mount points:

- if a <mountname> is given, then just that mount point gets dismounted
- if a <hostname> is given, then all mount points on that host get dismounted
- if neither a <mountname> nor a <hostname> are given, then all mount points are dismounted.

Before a mount point is dismounted, any open NFS files on it are closed.

Examples

`*Dismount tp1usr`	Dismounts the mount point `tp1usr`, closing all open NFS files on it.
`*Dismount -Host tp1`	Dismounts all mount points on the host `tp1`, closing all open NFS files on the host.
`*Dismount`	Dismounts all mount points in use, closing all open NFS files.

Related commands

*Mount

*Logon

Sets the name server to use and/or authenticates a user/password pair

Syntax

`*Logon [-Host <hostname>] [<username>[[CR]<password>]]`

Parameters

`<hostname>`	Chooses this host as the name server for this and subsequent name requests.
`<username>`	The user whose details are to be obtained from the name server.
<password>	The user's password. If not given you will be prompted for this.

Use

Sets the name server to use and/or authenticates a user/password pair, depending on the parameters passed:

- if a <hostname> is given, then it sets the name server to use for this and subsequent authentication using `*Logon`
- if a <username> and <password> are given, they are passed to the current name server for authentication; they are also used to access any mount points you create in the future
- if no parameters are given, then the current user and name server are displayed.

Examples

`*Logon -Host tp1`	Sets `tp1` as the current name server.
`*Logon mhardy`	Prompts for the password, which is reflected as dashes; authenticates the pair using the current name server (ie `tp1`); if the password is valid, sets the current user to `mhardy`.
`*Logon`	Displays the current user (`mhardy`) and the current name server (`tp1`).

Related commands

*Mount

*Mount

Lists or mounts NFS mount points

Syntax

`*Mount [-Host <hostname>] [<mountname> <mount path>]`

Parameters

`<hostname>`	Chooses this host as the NFS server for this and subsequent mount requests.
`<mountname>`	Specifies the name this mount point shall be referred to in the RISC OS world.
`<mount path>`	Specifies the directory on the NFS server which shall act as $ on NFS::<mountname>.

Use

Lists or mounts NFS mount points, depending on the parameters passed:

- if a <hostname> is given, then it sets the host as the NFS server to be used for this and any subsequent mount requests; it also lists all mount points on the host, giving the mount name, mount path and user name for each mount point
- if a <mountname> and <mountpath> are given, then the <mountpath> is mounted, using the last user name that was successfully authenticated with `*Logon`
- if no parameters are given, then all current mount points are listed, showing the mount name, mount path and user name for each mount point.

The files accessed via a mount point are always accessed using the details of the user which was current at the time the *Mount command was executed.

An NFS mount does **not** set the URD, library etc.

Examples

`*Mount -Host tp1 tp1usr /usr`	Mounts the directory `/usr` on the host `tp1`, giving it the mount name `tp1usr`.
`*Mount -Host tp1`	Lists all mount points on `tp1`.
`*Mount`	Lists all mount points.

Related commands

*Logon, *Dismount

*NFS

Selects NFS as the current filing system

Syntax

```
*NFS
```

Use

*NFS selects NFS as the filing system for subsequent operations. Remember that it is not necessary to switch filing systems if you use the full pathnames of objects. For example, you can refer to ADFS objects when NFS is the current filing system.

Example

```
*NFS
```

Related commands

*ADFS, *Net, *RAM, *SCSI

*TimeOffset

Sets the time offset in minutes from GMT to local time

Syntax

```
*TimeOffset <number>
```

Parameters

`<number>`	minutes offset from GMT to local time.

Use

This sets the time offset in minutes from GMT to local time. RISC OS NFS adds this offset when it interprets the server's date stamps, which it assumes to be in GMT. The offset can be negative or positive.

Example

```
*TimeOffset 60
```

Related commands

None

File mapping

Introduction

There are a number of differences between the UNIX and RISC OS models of a filing system, the more important being:

- length of filenames
- use of special characters in filenames
- numbers of attribute bits stored with files
- meaning of attribute bits
- use of file types
- soft links.

Because of these clashes changes have to be made when mapping RISC OS file names and attributes to UNIX ones, and vice versa. Generally the changes made when mapping one way are reversed when mapping the other way, so the system is as transparent as possible if only viewed from RISC OS. If you view the files using the remote filestore though, you'll notice some differences.

This chapter outlines how the mapping of file names takes place, and what differences you'll notice between the RISC OS view of a file and the UNIX view. Consecutive sections look first at RISC OS NFS, then at the Ftp protocol module.

NFS – its file mapping from RISC OS to UNIX

This section describes how RISC OS NFS maps files from RISC OS to UNIX.

Filenames

Character translation

The first change RISC OS NFS makes to a filename is to translate the character '/' (the UNIX directory separator) to '.', for example:

RISC OS name	**UNIX name**
fred/c	fred.c
/profile	.profile

File type extensions

RISC OS NFS then adds a filename extension to store the RISC OS file type.

You can set the extension used for any given file type to one of your choice. To do so you must edit the `extensions` file, held within the Internet application. See the section at the end of this chapter entitled *Editing the extensions file*.

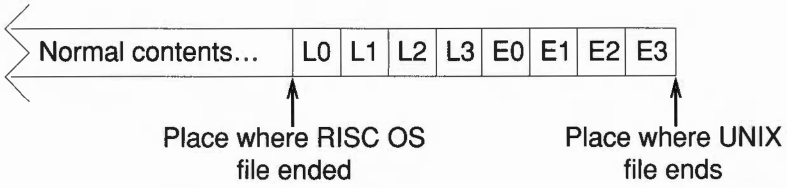
If you haven't set up a filename extension for a given file type, then a default extension gets used instead. The default mapping of a RISC OS file called `Fred` is as follows:

RISC OS type	**UNIX name**	
Text	Fred	
UNIX Ex	Fred	
Draw (&aff)	Fred,aff	
Obey (&feb)	Fred,feb	and other file types similarly…
dead †	Fred,xxx	
untyped	Fred,lxa	
directory	Fred	

† A dead file is one that has been created but the contents of which are being updated. For example when NetFS copies a file to a file server it reserves space by creating a dead file before writing to it.

File contents

The contents of files are unchanged when transferring to UNIX, save for untyped files. These have their load and execute addresses appended to the file, making it 8 bytes longer:

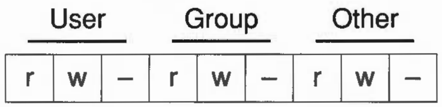


L0 is the least significant byte of the load address, L3 the most significant. Bytes E0 to E3 are the execute address.

Access attributes

When creating a new file or directory

You can use the system variable `NFS$CreateAccess` to control how RISC OS NFS sets the read/write access attributes for user, group and other when creating a file or directory on UNIX. This variable uses six of its lowest nine bits:



You should set the variable in a boot file; see your *RISC OS User Guide* if you need help on this. You can set it in octal by using a leading '`0`' (you'll find this familiar if you've ever used the UNIX `chmod` command with numbers), or in hexadecimal by using a leading '`0x`', or in decimal by just using a number. So the following would all set the variable to specify user read/write access, group read only access, and no access to others:

```
*Set NFS$CreateAccess 0640        (using octal)
*Set NFS$CreateAccess 0x1A0       (using hexadecimal)
*Set NFS$CreateAccess 416         (using decimal)
```

If this variable exists then files and directories are created with the read/write access it specifies. Files of type UNIX Ex also have their execute attributes set to be the same as the corresponding read bits in the variable.

If this variable does not exist then files are created with user read/write access, and with user execute permission if the files' type is UNIX Ex. Directories are created with user read, write and execute permission.

When mapped from RISC OS

When RISC OS NFS sets the access to a UNIX file using RISC OS attributes they are mapped as follows:

RISC OS bit	UNIX bit
owner read	user read
	user execute is also set if owner read is set and the file's type is UNIX Ex
owner write	user write
public read	group read and other read
	group execute and other execute are also both set if public read is set and the file's type is UNIX Ex.
public write	group write and other write
locked	(discarded)

Similarly, when RISC OS NFS sets the access to a UNIX directory using RISC OS attributes they are mapped as follows:

RISC OS bit	UNIX bit
owner read	ignored – ie user read is left unchanged
owner write	ignored – ie user write is left unchanged
	user execute is always set
public read	group read and other read
public write	group write and other read
locked	NOT group execute and NOT other execute

Dates

UNIX date stamps any files just as usual if you use RISC OS NFS to create or amend them.

Finding an object

When RISC OS NFS is finding an object it searches in this order, using the first match it makes:

1 It searches for an exact name match.

2 It searches for an exact name match after any RISC OS specific extension has first been removed.

3 It searches for a name match ignoring case, after any RISC OS specific extension has first been removed.

Ftp – its file mapping from RISC OS to UNIX

This section describes how the Ftp protocol module maps files from RISC OS to UNIX.

Filenames

The Ftp protocol module treats filenames and adds file type extensions in exactly the same way as RISC OS NFS. It shares the same `extensions` file.

File contents

The Ftp protocol module does not change the contents of any files, including untyped files.

Access attributes

The access attributes of any file or directory created as a result of an Ftp transfer depend on the UNIX host and its implementation of `ftpd` – the program that handles UNIX's end of the file transfer. They'll be exactly the same attributes as those you get from Ftp transfers from any other machine.

Dates

Similarly, the date stamps of any file or directory created as a result of an Ftp transfer are those normally set by UNIX.

Finding an object

When the Ftp protocol module is finding an object it always searches for an exact name match.

NFS – its file mapping from UNIX to RISC OS

This section describes how RISC OS NFS maps files from UNIX to RISC OS.

Filenames

File type extensions

The first change RISC OS NFS makes is to remove any filename extension used to store the RISC OS file type.

It starts by looking through the `extensions` file to see if the filename has an extension that matches one you specified; if so, the extension gets removed. You can in fact set up a different mapping for each direction of file transfer, so you can map many UNIX file extensions to single RISC OS file types. See the section at the end of this chapter entitled *Editing the extensions file*.

If RISC OS NFS can't find a matching filename extension in the `extensions` file it then tries to remove any of its own default extensions; so the following all appear as `Fred` under RISC OS:

UNIX name	Notes
Fred	
Fred,*hhh*	*hhh* is 3 lower-case hex digits
Fred,xxx	
Fred,lxa	

Truncation

The next thing RISC OS NFS does to a filename is to truncate it to the length set by the system variable `NFS$TruncateLength`. By default this is set to the value 10 – the same length as the maximum that the desktop Filers can handle. It only gets read once, when the NFS module is loaded.

If you want a different truncate length use the `*Set` command, say in a boot file:

```
*Set NFS$TruncateLength 12
```

If you're using NFS from the command line you may want to override filename truncation. To do so set the variable to a large number, eg 1000000.

Character translation

The final change RISC OS NFS makes to a filename is to translate the character '.' (the Acorn directory separator) to '/', for example:

UNIX name	RISC OS name
fred.c	fred/c
.profile	/profile

File contents

RISC OS NFS makes the last 8 bytes of any file with a '`,lxa`' extension invisible; this is to hide the load and execute addresses it presumes itself to have appended.

- If you generate a file in UNIX with a '`,lxa`' extension which is less than 8 bytes long, you will get unpredictable behaviour if you try to manipulate it from RISC OS.

Access attributes

When the access attributes of a UNIX file or directory get translated by RISC OS NFS they are mapped as follows:

UNIX Bit	RISC OS bit
user read	owner read
user write	owner write
user execute	(discarded)
group read	(discarded)
group write	(discarded)
group execute	(discarded)
other read	public read
other write	public write
other execute	(discarded for files) NOT locked for directories

Dates

RISC OS NFS always uses the UNIX last modified date stamp to map to a RISC OS date stamp. It assumes the UNIX date stamp to be in GMT, and uses the value set by `*TimeOffset` to convert this to local time. For details of `*TimeOffset` see the end of the chapter earlier in this part entitled *NFS*.

File types

RISC OS NFS resolves file types by looking for any filename extension used to store the RISC OS file type. It does so at the same time time as it resolves filenames – see also the earlier section entitled *Filenames*.

It starts by looking through the `extensions` file to see if the filename has an extension that matches one you specified; if so, it sets the file to the corresponding file type. See the section at the end of this chapter entitled *Editing the extensions file*.

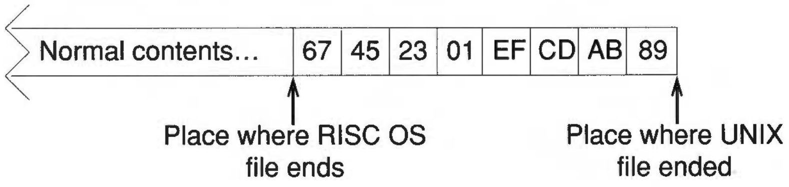
If RISC OS NFS can't find a matching filename extension in the `extensions` file it then sets the file type using its default file extensions. So, again taking the example of a file that will be displayed as `Fred`:

UNIX name	**Notes**	**RISC OS type**
Fred	UNIX directory	Directory
Fred	no execute bit is set	Text
Fred	any execute bit is set	UNIX Ex
Fred,*hhh*	*hhh* is 3 lower case hex digits	&*hhh*
Fred,xxx		dead
Fred,lxa		none, undated

See also *Soft links* below.

Load and execute addresses

If a UNIX file has the extension '`,lxa`' then RISC OS NFS assumes it to be a RISC OS untyped file that it created on UNIX. It uses the last 8 bytes of the file to give the load and execute addresses. So if they were:



the load address would be 01234567, and the execute address would be 89ABCDEF.

Soft links

RISC OS NFS resolves soft links up to eight times – that is, whilst following a soft link, it only allows eight soft links to be traversed. If this traversal reaches an existing object other than a soft link:

- the object's UNIX attributes and contents get used
- the soft link's UNIX name gets used to determine the RISC OS file type.

In other words soft links behave transparently except that, where there is more than one soft link to a file, its type may differ depending on which soft link you use to view it.

RISC OS NFS can't traverse a soft link that leaves a mount. If a UNIX link name starts with the character '/' then RISC OS NFS treats it as the root of its mount. Consequently absolute soft links will only work if you've mounted the UNIX root directory '/' and if the soft link does not leave the root filing system. For example, if you had mounted `/usr` then this UNIX soft link in the `/usr` directory would be traversed:

```
lrwxrwxrwx  1 root     wheel        11 Feb 23 17:19 man -> ./share/man
```

whereas this one wouldn't be:

```
lrwxrwxrwx  1 root     wheel        11 Feb 23 17:19 man -> /usr/share/man
```

We advise that when you make soft links on UNIX you always make relative links (ie start them with '`.`' or '`..`') rather than absolute ones.

If a soft link does not resolve to an existing non-soft-link object within eight expansions it's displayed as a file with type 'SoftLink' (&FDC). You can't do anything from RISC OS with one of these dead soft links.

Other object types

Block and character special files and named sockets are displayed as UNIX Ex files. Fiddle with these from RISC OS at your peril!

Ftp – its file mapping from UNIX to RISC OS

This section describes how the Ftp protocol module maps files from UNIX to RISC OS.

Filenames

The Ftp protocol module converts filenames and removes file type extensions in exactly the same way as RISC OS NFS. It shares the same `extensions` file.

File contents

The Ftp protocol module does not change the contents of any files, including untyped files.

Access attributes

The Ftp protocol module creates RISC OS files and directories with owner read and write access.

Dates

When the Ftp protocol module creates a file or directory, it date stamps it with the current RISC OS time and date.

Load and execute addresses

When the Ftp protocol module creates an untyped file, it will have an undefined load and execute address.

Soft links

The Ftp protocol module traverses soft links.

Other object types

It is very unlikely you will be able to access block and character special files and names sockets over Ftp. If you can, you fiddle with them at your peril!

Editing the extensions file

The `extensions` file is held in the `files` subdirectory of the Internet application, and configures the mapping of RISC OS file types to UNIX filename extensions. To add your own filename extensions for specific RISC OS file types you need to edit this file:

1. Load Edit onto the icon bar – if it's not already loaded.
2. Open the directory display containing the `!Internet` application – if it's not already open.
3. Open the `!Internet` application directory by holding down the *Shift* key while you double-click on the `!Internet` icon.
4. Open the `files` directory.
5. Load it into Edit by dragging its icon to the Edit icon on the icon bar.
6. Add to the file your own mappings of file type to UNIX file extension.
   - There are two sets of mappings: one for files coming from RISC OS (starting immediately beneath the 'From extensions:' line), another for files returning to RISC OS (starting immediately beneath the 'To extensions:' line).
   - The general syntax is:

     <RISC OS file type> <new extension> [<anything>]

     The RISC OS file type can be the name of a file type, or its file type number in hexadecimal. So to give Data files (type &FFD) the extension '`.dat`' you could use either of these lines:

     ```
     Data .dat
     ffd .dat
     ```
   - If you add a third field ('<anything>' above) then the extension becomes 'sticky'.

     When moving to UNIX the extension is only added if it's not already present. So if the line were to read:

     ```
     ffd .dat sticky
     ```

     the Data file `output` would be renamed `output.dat`, whereas the Data file `output.dat` would not be renamed.

     When returning from UNIX the extension doesn't get removed; otherwise it's handled the same as ever, so the file type gets set using this extension.

We expect you'll want the 'To extensions:' part to duplicate the entries in the 'From extensions' part, so any extension that gets added when a file is transferred to UNIX gets removed again if the file returns to RISC OS. However, there may be a lot of UNIX extensions that you wish to convert to a single RISC OS file type. For example, you may have several UNIX applications each of which generates text files with different extensions – say '.txt', '.doc' and '-asc'. To do so, just add extra entries to the 'To extensions', thus:

```
Text .txt
Text .doc
Text -asc
```

7 When you've added all the extensions you want to, save the edited `extensions` file, overwriting the old one.

Appendices

Appendix A – UNIX filesystems and access

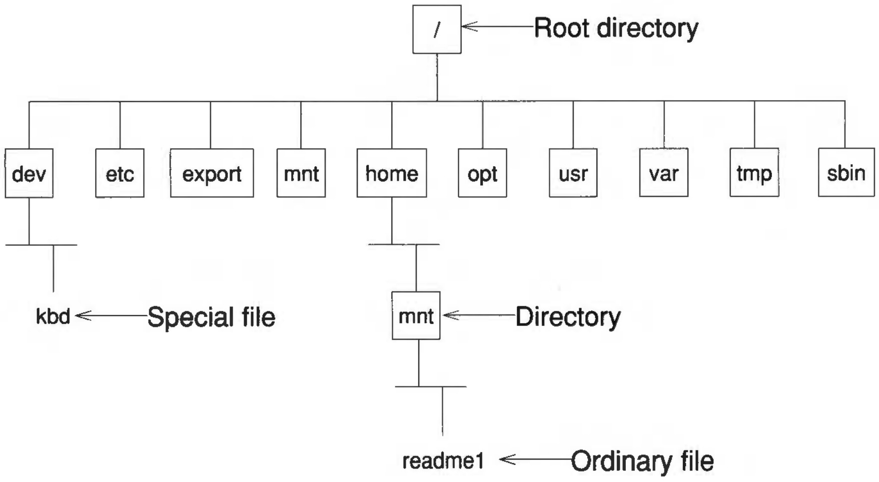
Introduction

This appendix gives a simplified overview of a UNIX filesystem. It describes its structure, and how it controls access to individual files, giving you enough information to start using it. It assumes you're already familiar with RISC OS filesystems, and describes UNIX filesystems largely by comparison with them.

Eventually you may find you need more detail than we provide here; when that time comes, you should see the documentation supplied with your UNIX computer.

The structure of a UNIX filesystem

A UNIX filesystem is based upon a hierarchical tree structure, similar to that used by RISC OS. For example, the diagram below shows the structure of the uppermost levels of a typical UNIX filesystem:



Types of UNIX files

UNIX has three different types of file that together make up the complete filesystem:

- An *ordinary file* is used to store such things as data or programs – just as files are in RISC OS
- A *directory* is used to organise the structure of the filesystem – again, just as they are used in RISC OS. A directory may contain files as well as other directories called *subdirectories*.

  In the filesystem diagram above, all directories are placed in boxes to distinguish them from files.
- A *special file* is the file used to represent a physical device on the filesystem, such as a keyboard. There is at least one file corresponding to each physical device on the computer. They are normally found in the `/dev` directory. A special file is treated just like any other UNIX file.

UNIX pathnames

UNIX uses a *pathname* to describe a file, just as RISC OS does. There are differences in the syntax, though:

- UNIX uses the '/' character to represent the root directory, whereas RISC OS uses the '$' character.
- UNIX uses the '/' character as its directory separator, whereas RISC OS uses the '.' character.
- UNIX filesystems are case sensitive (it matters whether you use upper or lower case), whereas RISC OS is case insensitive (it doesn't matter whether you use upper or lower case).

So, for example, the ordinary file in the filesystem diagram above has the pathname:

```
/home/guest/readme1
```

- the initial '/' refers to the root directory
- *home* is the name of a directory in the root directory
- the second '/' is being used as a directory separator (as also is the final '/')
- *guest* is the name of a subdirectory of the *home* directory
- *readme1* is the name of the ordinary file itself.

Users and Groups

When you log on to a UNIX computer you give the user name you're going to use for the session, just as you do with the Econet in RISC OS. But in UNIX when you give your user name, you're also automatically made a member of a *group*.

Groups are set up by the system administrator: perhaps most importantly, they set up the file that specifies what group you're put in when you log on. Each group typically gathers together users who work on common material: for example, users in a University Chemistry Department may belong to the group `chemistry`. The main purpose of a group is to share permissions and privileges across such a group of people who need common access to each others' files, as outlined below.

UNIX file attributes

UNIX uses file attributes to control who is allowed to access individual files. The system is similar to but more sophisticated than that used by the RISC OS NetFS. The main differences are as follows:

- UNIX uses the same access system for all its files (including directories and special files), whereas RISC OS treats directories somewhat differently.
- UNIX uses read, write and *execute* permission for each file, whereas RISC OS just uses read and write permission for each file.
- UNIX applies these types of permission to three categories of users, whereas RISC OS uses just two.

The three separate categories of user permission that UNIX has are *user*, *group* and *other*.

- *User* permission refers to the individual user who owns the file; it is similar to the *owner* in RISC OS.
- *Group* permission refers to the *group* of users to which the file 'belongs'; there is no equivalent concept in RISC OS. (When a file gets created, it 'belongs' to the group its owner is using at the time.)
- *Other* permission refers to all other users; it is similar to the *public* in RISC OS.

How UNIX uses permissions

As an example of how UNIX permissions work, let's look at how UNIX would check things when you try to execute (ie run) a file:

1 If you are the owner of the file, and if *user execute permission* is set, UNIX executes the file; if not:
2 If the file 'belongs' to the group you're in, and if *group execute permission* is set, UNIX executes the file; if not:
3 If *other execute permission* is set, UNIX executes the file; if not:
4 UNIX generates an error message

UNIX similarly checks the read permissions when you try to read a file, and the write permissions when you try to write to a file.

Directories

As we said earlier, directories are treated just like any other UNIX file; but it may not immediately be obvious what it means to read, write or execute a directory. Bearing in mind that a directory holds details of all the files and subdirectories it contains:

- You need read permission to read a directory – that is, to list its contents.
- You need write permission to write to the directory – that is, to create or remove files.
- You need execute permission to search a directory – that is, to access files in it or to make it the current directory.

UNIX filing commands – an introduction

UNIX provides a full set of commands to manipulate the filesystem, just like any other operating system. Below, we list some of the most common ones. You can use these commands (and any other UNIX command) using the Telnet protocol, either via the VT220 application and the Telnet protocol module, or via the *Telnet command.

Command	Use
cd	Change directory
chgrp	Change group ownership of files
chmod	Change permission of files
chown	Change ownership of files
cp	Copy files
find	Find files
ln	Link to a file
ls	List contents of a directory
mkdir	Make a directory
mv	Move or rename files
rm	Remove files
rmdir	Remove directories
touch	Datestamp/create a file

For full details of these commands see the corresponding UNIX manual pages, and any extra documentation supplied with your UNIX computer.

Other UNIX security features

There are other standard features in UNIX (and NFS) that allow system administrators to control access to entire filesystems, or to just a part of a filesystem. Furthermore, many manufacturers choose to implement extra security features of their own. We don't cover system-wide access restrictions in this appendix; instead your system administrator will be able to tell you how things are configured at your site.

Appendix B – Installing the software on RISC OS

Introduction

This chapter tells you how to install the TCP/IP Protocol Suite on a RISC OS computer.

This is only a part of the process of installing the TCP/IP Protocol Suite on a site. Your system administrator must already have made some changes to the software before you can follow these instructions.

If you're the system administrator you should see the separate *TCP/IP Protocol Suite Installation Guide*, which gives full details of that process. (It also has a copy of this appendix, slightly altered to fit the different context.)

Installing the software

This section assumes you are using the desktop, and are familiar with simple use of it. If you have any problems refer to the *User Guide* supplied with your RISC OS computer.

Installing the TCP/IP Protocol Suite

First you've got to install the TCP/IP Protocol Suite.

1 Find out from your system administrator:

- the *principal host name* they've assigned to your RISC OS computer
- if you need to change the values to which two variables named `INet$EcoIPAddr` and `INet$EtherIPAddr` are set, and if so what values you should change them to
- where to get the master copy of the software they have prepared for your part of the network.

2. Decide where you're going to install the software.

   You can install the copy on any media you like, such as a hard disc, floppy discs or a remote file server. All that is important is that you preserve the directory structure that is on your copy – things that are in the same directory must stay together.

   - If you're installing the software on a hard disc or an Econet file server, we suggest you make a directory called (say) `TCP_IP` to hold the software. This directory can be anywhere you like – it needn't be in the root directory.
   - You don't have to do this, and may prefer to put the software in a directory that already contains other applications.

3. Copy the master for your part of the network to the location you've just chosen.

   - If you're copying from one pair of floppy discs to another, you'll need to use the 'Backup' option from the floppy drive's icon bar menu.
   - Otherwise you just need to drag the files from the master copy to the location you've chosen.

   You've now finished with the master copy.

4. Load Edit onto the icon bar – if it's not already loaded.
5. Open the directory display containing the `!Internet` application that you just installed – if it's not already open.
6. Open the `!Internet` application directory by holding down the *Shift* key while you double-click on its icon. (`!Internet` is on the *Network* distribution disc.)
7. Edit the `!Configure` file:

   - Load it into Edit by dragging its icon to the Edit icon on the icon bar.
   - Change the word *yourhost* in the line:

     ```
     set INet$HostName yourhost
     ```

     to the computer's principal host name.
   - If your system administrator has said you need to, change the line that sets the `INet$EcoIPAddr` system variable. Follow their instructions.

- If your system administrator has said you need to, change the line that sets the `INet$EtherIPAddr` system variable. Follow their instructions.
- Save the edited `!Configure` file, overwriting the old version.

Updating the !System directory

You must also ensure that some system resources on the RISC OS computer are sufficiently up to date.

8 Open the `System` directory. (The `System` directory is on the *Network* distribution disc.)

9 Open its `Modules` subdirectory.

10 Open the computer's `!System` directory (as supplied on the *RISC OS Applications Disc 1*) by holding down the *Shift* key while you double-click on its icon.

11 Open its `Modules` subdirectory.

12 Choose Full Info from the Filer's Display submenu so you can see how old the modules are in both `Modules` directories.

13 Copy each of the modules supplied with the TCP/IP Protocol Suite to the computer's `!System` directory if either of these is true:

- the same module is already in `!System`, but it's older than the TCP/IP Protocol Suite version of it
- there isn't a copy of the module in `!System`.

14 Once you've copied any necessary modules, you can delete the `System` directory from the installed software.

You've now finished installing the TCP/IP Protocol Suite on RISC OS.

Index

Reader's Comment Form

TCP/IP Protocol Suite User Guide

We would greatly appreciate your comments about this Guide, which will be taken into account for the next issue:

Did you find the information you wanted?

Do you like the way the information is presented?

General comments:

If there is not enough room for your comments, please continue overleaf

How would you classify your experience with computers?

☐	☐	☐	☐
First-time user	**Used computers before**	**Experienced user**	**Programmer**

Cut out (or photocopy) and post to:

Dept RC, Technical Publications
Acorn Computers Limited
645 Newmarket Road
Cambridge CB5 8PB.

Your name and address:

This information will only be used to get in touch with you in case we wish to explore your comments further.

Acorn

ANI 60