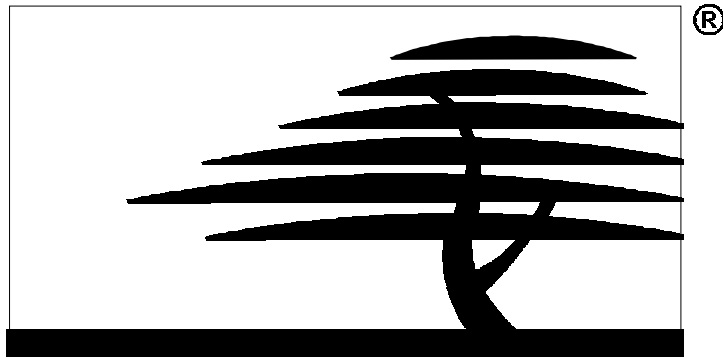




SCO

SCO OpenServer™ Release 5 Network Commands

Job Aid

Course Release: 2.0

For Software: SCO OpenServer™
Release 5

©1983-1999 The Santa Cruz Operation, Inc. All rights reserved.

©1984-1990 AT&T.

This publication is protected under copyright laws and international treaties. ©1992-1994 AT&T Global Information Solutions Company; ©1987-1989 Computer Associates, Inc.; ©1987 Convergent Technologies, Inc.; ©1989 Digital Equipment Corporation; ©1987-1989 Hewlett-Packard Company; ©1994-1996 Netscape Communications Corporation; ©1993-1994 Platinum Technologies; ©1988 Massachusetts Institute of Technology; ©1985-1989 Metagraphics Software Corporation; ©1980-1989 Microsoft Corporation; ©1989 Open Software Foundation, Inc.; ©1993-1994 Programmed Logic Corporation; ©1994-1997 Sun Microsystems, Inc.; ©1988 UNIX Systems Laboratories, Inc.; ©1997 VERITAS Software Corporation.

Information in this document is subject to change without notice and does not represent a commitment on the part of The Santa Cruz Operation, Inc.

Business/21, dbXtra, DiSCover, Internet Way of Computing, IWoC, Multiscreen, ODT, Open Desktop, Optimized For Internet Computing and its logo, Panner, SCO, SCO ACE, SCO CIFS Bridge, SCODoctor, SCO Doctor for Networks, SCO Doctor Lite, SCO Global Access, the SCO logos, SCO MPX, SCO MultiView, SCO Nihongo OpenServer, SCO OK, the SCO OK logo, SCO OpenServer, SCO Open Server, SCO Portfolio, SCO POS System, SCO Premier Motif, SCO TermLite, SCO ToolWare, SCOTopia, SCO Vision 97, SCO VisionFS, SCO Visual Tcl, Skunkware, Tarantella, The Business Choice, The Santa Cruz Operation, UnixWare, Universal Server, VP/ix, and Zones are trademarks or registered trademarks of The Santa Cruz Operation, Inc. in the USA and other countries. APC, SoftCare, and SoftTech are service marks of The Santa Cruz Operation, Inc. Deskterm, Deskworks, IXI, IXI Desktop, the IXI logo, IXI Panorama, Wintif, and X.desktop are trademarks or registered trademarks of IXI Limited, a subsidiary of The Santa Cruz Operation, Inc. X.tra is a service mark of IXI Limited. Codon, Devkit, Vision, Esprit, Kodon, PC-Connect, SQL-Retriever, SuperVision, Super.Vision, TermVision, Term.Vision, Vision Builder, Visionware, Visionware Direction, the Visionware logo, Visionware SQL-Retriever, Visionware Super.Vision, the XV logo, XVision, and X-Visionware are trademarks or registered trademarks of Visionware Limited, a subsidiary of The Santa Cruz Operation, Inc. X/Open and UNIX are registered trademarks and the X Device is a trademark of The Open Group in the United States and other countries. Cheyenne and ARCserve are registered trademarks of Cheyenne Software, Inc. Netscape, Netscape Navigator, Netscape Communications Server, Netscape Commerce Server, Netscape Proxy Server, Netscape FastTrack Server, Netscape Enterprise Server, Netscape SuiteSpot, Netscape Catalog Server, Netscape News Server, Netscape Mail Server, and Netscape Navigator Gold are trademarks of Netscape Communications Corporation. NFS was developed by Legent Corporation (formerly Lachman Associates, Inc.) based on LACHMAN SYSTEM V NFS. LACHMAN is a trademark of Legent Corporation. NFS is a trademark of Sun Microsystems, Inc. TCP/IP was developed by Legent Corporation (formerly Lachman Associates, Inc.) based on LACHMAN SYSTEM V STREAMS TCP, a joint development of Lachman Associates and Convergent Technologies. MPX was developed by Corollary, Inc. VP/ix is a product developed and licensed by Phoenix Technologies, Ltd/INTERACTIVE Systems Corporation. XRemote is a registered trademark of Network Computing Devices, Inc. Oracle is a registered trademark of Oracle Corporation, Redwood City, California. Java is a trademark of Sun Microsystems, Inc. Adobe is a trademark of Adobe Systems Incorporated and is registered in the U.S. Patent and Trademark Office. Reliant is a registered trademark of Siemens Pyramid Information Systems, Inc. (formerly Pyramid Technology Corporation).

All other brand and product names are or may be trademarks of, and are used to identify products or services of, their respective owners. The Santa Cruz Operation, Inc. reserves the right to change or modify any of the product or service specifications or features described herein without notice.

This document is for information only. SCO makes no express or implied representations or warranties in this document.

The Santa Cruz Operation, Inc. and SCO Skunkware are not related to, affiliated with or licensed by the famous Lockheed Martin Skunk Works®, the creator of the F-117 Stealth Fighter, SR-71, U-2, Venturestar, Darkstar, and other pioneering air and spacecraft.

The SCO software that accompanies this publication is commercial computer software and, together with any related documentation, is subject to the restrictions on US Government use as set forth below. If this procurement is for a DOD agency, the following DFAR Restricted Rights Legend applies:

RESTRICTED RIGHTS LEGEND: When licensed to a U.S., State, or Local Government, all Software produced by SCO is commercial computer software as defined in FAR 12.212, and has been developed exclusively at private expense. All technical data, or SCO commercial computer software/documentation is subject to the provisions of FAR 12.211 - "Technical Data", and FAR 12.212 - "Computer Software" respectively, or clauses providing SCO equivalent protections in DFARS or other agency specific regulations. Manufacturer: The Santa Cruz Operation, Inc., 400 Encinal Street, Santa Cruz, California 95060.

The copyrighted software that accompanies this manual is licensed to the End User only for use in strict accordance with the two End User License Agreements. All such Agreements should be read carefully before commencing use of the software. Document Version: UnixWare 7

30 January 1998

Table of Contents

Lesson 1: Network Commands	1
TCP/IP-standard Commands	5
telnet (TC)	5
Invoking telnet (TC) with arguments	7
Invoking telnet (TC) without arguments	9
telnet (TC) commands	9
Exiting a telnet (TC) session	9
ftp (TC)	11
Invoking ftp (TC)	11
Anonymous ftp	13
ftp (TC) commands	13
Variations on the get and put commands	17
The r -commands and Trusted Access	19
The rlogin (TC) command	19
Invoking rlogin (TC)	21
Exiting a rlogin (TC) session	23
Configuring Trusted Access	25
Configuring trusted host access	27
Configuring trusted user access	29
The trusted access sequence	31
Trusted access on the local host	31
Trusted access on the remote host	33
The rcp (TC) command	35
Invoking rcp (TC)	35
rcp (TC) command-line flags	41
The rcmd (TC) command	41
Invoking rcmd (TC)	43
I/O redirection with rcmd (TC)	47
Summary	51
Relevant Commands and Files	53

1

Network Commands



Lesson 1

Network Commands

Introduction

Every implementation of TCP/IP includes two commands, **telnet**(TC) and **ftp**(TC). These commands allow you to connect to a remote system and transfer files. Three additional commands, **rlogin**(TC), **rnp**(TC), and **rcmd**(TC) are included with most UNIX[®] implementations of TCP/IP. They are used to execute commands on remote systems.

This job aid is designed to provide you with information needed to use **telnet**, **ftp**, **rlogin**, **rnp**, and **rcmd** effectively.

It also describes how to set up trusted access for use with the *r-commands*—the network commands that begin with an **r** (**rcmd**, **rlogin**, and **rnp**). Trusted access must be configured before you can use the **rnp** and **rcmd** commands; it can be useful for the **rlogin** command as well.

Objectives

This job aid covers basic networking commands. By the end of this lesson, you will be able to:

- Use the **telnet** and **ftp** commands
- Set up user trusted access
- Use the **r-commands**



1.1 TCP/IP-standard Commands

All TCP/IP implementations include these two commands:

telnet	Allows users to log into a remote system
ftp	Allows users to transfer files to and from a remote system

1.1.1 telnet(TC)

Although every TCP/IP implementation has a **telnet** command, typically only multiuser systems, like SCO OpenServer, accept incoming **telnet** connections. For example, you can **telnet** from a DOS system, but most implementations of TCP/IP for DOS do not allow you to **telnet** into the system. SCO OpenServer Release 5 is configured to accept **telnet** connections by default.

The **telnet** command in SCO OpenServer Release 5 does not perform *terminal emulation*. After, you establish a connection with a remote host, **telnet** passes the characters you type on your keyboard to the remote host without modification. Similarly, it displays any characters it receives from the remote host on your screen.

For example, if you log into an SCO OpenServer system using an ANSI terminal and **telnet** to another remote UNIX system, you must tell the remote UNIX system that you are using an ANSI terminal. Similarly, if using a Wyse-60 terminal, you have to tell the remote UNIX system you are using a Wyse-60 terminal.



1.1.1.1 Invoking telnet(TC) with arguments

The syntax for the **telnet** command is:

```
telnet [ -8 ] [ -E ] [ -L ] [ -a ] [ -c ] [ -d ]  
      [ -e escape_char ] [ -l user ]  
      [ -n tracefile ] [ -r ] [ host [ -port | port ] ]
```

In practice, you will rarely use all of these options. Generally you can invoke **telnet** by entering one of these command lines:

```
telnet hostname  
telnet IP address
```

where:

<i>hostname</i>	The resolvable name of a remote system
<i>IP address</i>	The IP address (in dotted-decimal notation) of a remote system

If the remote host can be accessed, you will see the login prompt for that system. Common reasons for failure at this point include:

Unknown host	You mistyped the name of the host, or your system cannot resolve hostname into an IP address
No route to host	You mistyped the IP address, or your system does not know the route to the specified hostname or IP address



1.1.1.2 Invoking telnet(TC) without arguments

You can also invoke **telnet** without arguments. Enter:

telnet

Without arguments, the **telnet** command displays this prompt:

telnet>

To connect to a remote host, enter either:

```
telnet> open hostname
telnet> open IP address
```

1.1.1.3 telnet(TC) commands

Access the **telnet** prompt (telnet>) by entering the **telnet** escape character. By default, this character is `[CTRL] ]`.

In addition to the **open** command described above, there are other **telnet** commands you can configure to suit your environment. For example, if you need to change the software flow-control characters, you can use the **set start** and **set stop** commands.

For a complete list of **telnet** commands, see the **telnet(TC)** manual page or enter a question mark (?) at the **telnet** prompt:

```
telnet> ?
```

1.1.1.4 Exiting a telnet(TC) session

To leave a **telnet** session:

- Exit the login session with the remote system. This will automatically cause **telnet** to clean up and exit.
- Type the escape character (by default, `[CTRL] ]`). Then enter the **quit** command at the **telnet** prompt:

```
telnet> quit
```



1.1.2 ftp(TC)

The **ftp**(TC) command allows you to transfer files to and from a remote host using the File Transfer Protocol (FTP). Many sites on the Internet have repositories of programs and data that you can retrieve using *anonymous FTP*. Many web browsers (such as, Netscape Navigator™ and NCSA Mosaic™) can perform anonymous FTP transfers for you.

As with **telnet**, every TCP/IP system includes the **ftp** command to transfer files between remote systems. However, not all systems accept incoming **ftp** connections. SCO OpenServer Release 5 can be configured to allow incoming FTP connections.

1.1.2.1 Invoking ftp(TC)

The syntax for the **ftp** command is:

```
ftp [-dgitvn] [host]
```

As with **telnet**(TC), you can invoke **ftp** in one of three ways:

```
ftp hostname  
ftp IP address
```

where:

hostname The resolvable name of a remote system

IP address The IP address (in dotted-decimal notation) of a remote system

In the first two cases, you are asked for a valid username and password on the remote host, provided that you do not encounter any errors (see Section 1.1.1.1 on page 1-7).

When invoked without command-line arguments, the **ftp** prompt (**ftp>**) displays:

```
ftp  
ftp>
```



1.1.2.2 Anonymous ftp

Some network systems are configured to allow anonymous FTP access. This means that instead of supplying a real username and password, you can supply the word **anonymous** or **ftp** as the username, and your e-mail address (for example, *user1@somedom.com*) as the password. In this case, your FTP session can access a designated part of the FTP directory tree.

1.1.2.3 ftp(TC) commands

There are numerous commands supported by **ftp**. For a complete list, see the **ftp(TC)** manual page, or enter a question mark (?) at the **ftp** prompt:

```
ftp> ?
```

The most commonly-used **ftp** commands include:

get Transfers a file from the remote host to the local host. It has two forms:

```
get remote-file-name local-file-name
get remote-file-name
```

If *local-file-name* is omitted, then *remote-file-name* is the name of both files.

put Transfers a file from the local host to the remote host. It has two forms:

```
put local-file-name remote-file-name
put local-file-name
```

If *remote-file-name* is omitted, then *local-file-name* is the name of both files.

You must have write permission for the directory in which you are trying to **put** the file. Some anonymous FTP sites have a directory called *Uploads* or *Incoming* that allows anonymous users to place files on the FTP site.

However, as a security precaution, this directory is often not readable by anyone, so that nobody can see what others have put there.



mget	Allows you to get multiple files using shell-format wildcards characters. Each file created on the local system has the same name as the one retrieved.
mput	Allows you to put multiple files using shell-format wildcards characters. Each file created on the remote system has the same name as the one sent.
cd	Changes the working directory on the remote host
lcd	Changes the working directory on the local host. Note: you cannot change your local host directory with the ! command.
hash	Causes pound signs (#) to display with every 1K bytes that are transferred between hosts. This is useful when transferring large files or when transferring across slow connections because you can see that the transfer that is in progress.
prompt	By default, mget and mput prompt you before each file is transferred. Enter the prompt command to turn off prompting.
mkdir	Creates a new directory on the remote host, provided that you have permission
!	Allows you to execute commands on the local host through a shell escape. Note: you cannot change directory by entering: !cd newdir This is because the ! command starts up a new shell. In this example, a new shell is started, its current directory is changed to <i>newdir</i>, and then that shell exits. The current directory of the FTP session never changed. To change the directory on the local host, use the lcd command.
ascii	Translates files as they are transferred from one host to the other. See the ftp(TC) manual page for the list of conversions performed.
binary	Specifies that no data conversion is done.



1.1.2.4 Variations on the get and put commands

Most modern FTP servers provide additional functionality to make transferring files across the Internet more efficient.

Two commonly-used features are:

- Automatic file compression
- Archiving the contents of a directory into a single file

You can also combine these two features. This is done by specifying certain file name extensions when using the **get** and **put** commands.

To compress a file, add the suffix “.Z” to the file name. To create a **tar**(C) archive, add the suffix “.tar” to the directory name. Tell the FTP server to do both using the suffix “.Z.tar” to the directory name. For example:

- To get a compressed version of a file called *myfile*, enter:

```
ftp> get myfile.Z
```

The *myfile* file is automatically uncompressed on your machine. This also works for the **put** command.

- To get the contents of a directory called *mydir*, enter:

```
ftp> get mydir.tar
```

The *mydir* directory is archived into a **tar** file and transferred to your host, where it is automatically converted back to individual files.

- To get all files in a directory called *mydir2* and compress them before transferring, enter:

```
ftp> get mydir2.tar.Z
```

This *mydir2* directory is placed into a **tar** file and transferred to your host, where it is automatically uncompressed and written as individual files.

Not all FTP servers provide this functionality, however, SCO OpenServer Release 5 can be configured to support these features.



1.2 The r-commands and Trusted Access

In earlier implementations of TCP/IP, the **telnet** and **ftp** commands required you to enter your username and password for the remote host every time you used one of the commands.

To simplify access to remote systems, two commands with similar functionality were created: **rlogin**(TC) and **rcp**(TC). These commands are used in conjunction with *trusted access*. On remote hosts for which trusted access is configured, you do not need to specify your password to log in or to copy files.

In addition, the **rcmd**(TC) command allows you to execute a command on a remote system and have the results displayed on your terminal. Trusted access is also required to use the **rcmd** command.

Because these network commands all begin with the letter **r**, they are referred to as the **r**-commands.

1.2.1 The rlogin(TC) command

The **rlogin** command serves the same function as **telnet**, however you can only use **rlogin** to log in to systems running UNIX operating systems. SCO OpenServer Release 5 is configured to accept **rlogin** connections by default.

The **rlogin** command included in SCO OpenServer Release 5 does not perform any terminal emulation. After you establish connection with a remote host, the **rlogin** command passes the characters you enter on your keyboard to the remote host without modification. Similarly, **rlogin** displays any characters it receives from the remote host on your screen.

For example, if you log into your SCO OpenServer Release 5 system using an ANSI terminal, and **rlogin** to another remote UNIX system, you must tell the remote system that you are using an ANSI terminal. Similarly, if you are using a Wyse-60 terminal, you also have to tell the remote system you are using a Wyse-60 terminal.



1.2.1.1 Invoking rlogin(TC)

The syntax for the **rlogin** command is:

```
rlogin [-L] [-7] [-8] [-ec] [-l username] hostname
```

In practice, all of these options are rarely used. To invoke **rlogin**, enter either of the following:

```
rlogin hostname  
rlogin hostname -lusername
```

where:

<i>hostname</i>	The resolvable name of a remote system
<i>username</i>	The login name to use on the remote host, if different from your current login name

If the remote host is accessible, but trusted access is not configured, you will see the password prompt for that system. To complete the login process, enter the password of your user name (or the one you specified with the **-l** option).

Common reasons for failure at this point include:

Unknown host	You mistyped the name of the name of the host or your system cannot resolve hostname into an IP address
No route to host	Your system does not know the route to reach the specified hostname
Login incorrect	You did not give the proper password for the username or the username does not exist on the remote host

If trusted access is configured, or if the username you specified does not have a password, you are logged in without having to specify a password. The same is true for the other **r**-commands. Note: Your account should always have a password if your machine is on a network to prevent others from accessing your account.



1.2.1.2 Exiting a rlogin(TC) session

To leave an **rlogin** session:

- Exit the UNIX login session. This automatically causes **rlogin** to clean up and exit.
- Enter a two-character escape character sequence “~.” (tilde-dot). This causes **rlogin** to clean up and exit.



1.2.2 Configuring Trusted Access

Trusted access allows you to use the enterprise network in a more convenient and secure manner.

If trusted access is not configured for the **rlogin**(TC) command, it prompts your users for a password. This password is transmitted across the network—perhaps even the Internet. Packets containing these passwords are relatively easy to intercept and identify. If trusted access is set up, no password is necessary for **rlogin**.

If trusted access is not set up, the **rcmd**(TC) and **rcp**(TC) commands do not work.

Individuals can configure trusted access at the user level. The network administrator can configure trusted access at the host level.



1.2.2.1 Configuring trusted host access

Trusted host access indicates that one host trusts the users on another host. When configured, these commands can be used without specifying a password:

- **rlogin**
- **rcp**
- **rcmd**

You must be **root** to configure trusted access at the host level. This configuration consists of creating or editing the */etc/hosts.equiv* file. Each entry in */etc/hosts.equiv* specifies the trusted remote system. illustrates the format of entries in an example */etc/hosts.equiv* file.

If the entries in exist on a machine called *corfu*, they have this meaning:

<i>barbados</i>	All non- root users on <i>barbados</i> with the same user name as those on this machine (<i>corfu</i>) are trusted. In other words, jane on <i>barbados</i> can rlogin , rcp , and rcmd to <i>corfu</i> as jane without specifying Jane's password on <i>corfu</i> .
-----------------	--

<i>tortola bob</i>	Allows bob on <i>tortola</i> to log in as user bob on this machine (<i>corfu</i>).
--------------------	--

WARNING: This entry allows **bob** on *tortola* to log in as any user on *corfu*, except **root**. This can be dangerous!

The **root** account can never be equivalenced by an entry in the */etc/hosts.equiv* file. Refer to *hosts.equiv*(SFF) for further information.

Note: If your network is connected to the Internet, you should always use fully-qualified host names in this file for security reasons.



1.2.2.2 Configuring trusted user access

Trusted user access allows users on one system to indicate which users from another system they trust to use their account, and enter these commands without providing a password:

- **rlogin**
- **rcmd**
- **rcp**

Unlike trusted host access, trusted user access can be created for users on different machines with different login names.

You can configure trusted user access for your users, or they can do it for themselves. The configuration consists of:

1. Creating or modifying a file called *.rhosts* (note the leading dot) in a user's home directory.
2. Setting the ownership so that the file is owned by either **root** or the owner of the home directory.
3. Setting the permissions so that the file is writable only by the owner.

If steps 2 and 3 are not performed, the file is ignored.

The format of entries in the *.rhosts* file is the same as those in the */etc/hosts.equiv* file. However, the semantics are somewhat different. If the entries shown in are put in the home directory of the user **jane** on the machine *corfu*, they have the following meaning:

barbados	User jane on <i>barbados</i> is trusted by the user jane on this machine (<i>corfu</i>). In other words, jane on <i>barbados</i> can rlogin , rcp , and rcmd to <i>corfu</i> as jane without specifying Jane's password on machine <i>corfu</i> .
tortola bob	User bob on <i>tortola</i> is trusted by the user jane on this machine (<i>corfu</i>). In other words, bob on <i>tortola</i> can rlogin , rcp , and rcmd to this machine (<i>corfu</i>) as jane , using the appropriate arguments to those commands, without specifying Jane's password.



1.2.3 The trusted access sequence

Whenever a **rlogin**, **rcmd**, or **rcp** command is executed, the systems involved goes through a set procedure to determine whether trusted access is configured.

1.2.3.1 Trusted access on the local host

These steps describe the trusted access sequence that takes place on the local host:

1. An **r**-command is issued by a user.
2. The local host attempts to resolve the remote host name to obtain its IP address.
3. If the remote host name cannot be resolved into an IP address, this error message displays:

host_name: Host name lookup failure

4. If an address for the remote host can be resolved, the **r**-command connects through the TCP/IP transport to the remote host.



1.2.3.2 Trusted access on the remote host

These steps describe the trusted access sequence that takes place on the remote host:

1. The user's account name is looked up in the password database.
2. If the user's account name is not found in the password database, the **rcmd** and **rcp** commands fail, and the **rlogin** command prompts for a password (that is how the **login** program functions), and then fails.
3. The user's account is checked to see if it has a (encrypted) password.
4. If the user's account does not have a password, the **r**-command executes. Note: This is very dangerous. You should never allow user accounts without passwords on your systems, especially if your network is attached to the Internet.
5. The user's account name is checked to see if the user is **root**.

If the user's account name is not **root**, then */etc/hosts.equiv* is checked for the local host's name.

If */etc/hosts.equiv* contains the name of the local host, the **r**-command executes.

6. If the account name is **root**, or if the */etc/hosts.equiv* check fails, then the *.rhosts* file in the remote user's home directory is checked. If it contains the local host's name, and possibly the local user's name (depending upon how the **r**-command was invoked) the **r**-command executes.
7. If the *.rhosts* file contains the desired entry, then the **r**-command executes.

Failure at this point means that the **rcmd** and **rcp** commands will fail. If the **r**-command invoked was **rlogin**, it prompts for the remote user's password, and only succeeds if the correct password is given.



1.2.4 The rcp(TC) command

The **rcp**(TC) command transfers files from one system to another and is similar to **ftp**(TC). Some differences include:

- The **rcp** command allows you to copy files between any two hosts on the network to which you have access, not just the local host and a single remote host.
- The **rcp** command requires trusted access, unless the target accounts on the remote hosts do not have a password set. The **rcp** command does not ask for a password.
- Because of the overhead in authenticating each **rcp** command, it is faster to use **ftp** if you are copying a large number of files.

1.2.4.1 Invoking rcp(TC)

The syntax for the **rcp** command is:

```
rcp [-p] file1 file2
rcp [-p] [-r] file1 ... directory
```

where filenames (*file1*, *file2*, and *directory*) have one of three forms:

<i>pathname</i>	Refers to a local file on the local host
<i>host:pathname</i>	<i>host</i> is the name of the host where the file resides; <i>pathname</i> is the path to the file on that host. If <i>pathname</i> is not an absolute path, then it is relative to the <i>host</i> users 's home directory.
<i>user@pathname:host</i>	<i>host</i> indicates where the file resides; <i>pathname</i> is the path to the file on that host. If <i>pathname</i> is not an absolute path, then it is relative to the <i>host</i> users 's home directory. Also, the transfer's permissions and trusted access is based on user rather than the account name with which you logged into the local host.



For example, given the following:

- There are three hosts, *atlantis*, *barbados*, and *corfu*.
- You are logged into the host *atlantis* as the user **bubba**.
- All hosts have accounts for the users **bubba**, **forrest**, and **dan**.
- In each user's home directory there is a directory called *stuff* and a file called *treasure*.
- Trusted access is configured as needed.

The following describes how to copy files on these systems:

- To copy the *treasure* file from your host to your account on the host *barbados*, enter:

```
rcp treasure barbados:treasure
```

- To copy the *treasure* file from your account on the host *barbados* to your current directory on your local host, enter:

```
rcp barbados:treasure treasure
```

- To copy the *treasure* file from your account on the host *barbados* to your account on the host *corfu*, enter:

```
rcp barbados:treasure corfu:treasure
```

- To copy the *treasure* file from your account on the host *barbados* to **dan**'s account on the host *corfu*, enter:

```
rcp barbados:treasure dan@corfu:treasure
```

- To copy the *treasure* file from **forrest**'s account on the host *barbados* to **dan**'s account on the host *corfu*, enter:

```
rcp forrest@barbados:treasure dan@corfu:treasure
```

- To copy the *stuff* directory from **forrest**'s account on the host *barbados* to **dan**'s account on the host *corfu*, enter:

```
rcp -r forrest@barbados:stuff dan@corfu:stuff
```



Note: The directories and files on the target machines do not have to exist. They will be created as necessary.

Common reasons for failure of the **rcp** command include:

Unknown host	You mistyped the name of the name of the host, or your system cannot resolve host name into an IP address
No route to host	Your system does not know the route to the specified host name
Permission denied	You do not have trusted access configured between your host and one or more of the remote hosts

If the user name you specified does not have a password, trusted access is not required. The same is true for the other **r**-commands.

Note: Your account should always have a password if the machine you are using is on a network.



1.2.4.2 rcp(TC) command-line flags

Two commonly-used **rcp** flags are:

- p** Maintains modification times and permissions (that is, the mode) of the source files. This is useful when relocating directories from one system to another.
- r** Copies the entire contents of a directory from one host to another

1.2.5 The rcmd(TC) command

Using the **rcmd**(TC) command is equivalent to using **rlogin**(TC) to access a remote system to run only one command. However, there is one significant difference. When you use the **rlogin** command, the remote system sets up a device called a pseudo-tty on which all of your commands run. Pseudo-ttys allow commands such as **vi**(C), **more**(C), and **pg**(C) to control your terminal remotely. Without a pseudo-tty, these commands will not work properly across the network.

The **rcmd** command does not create a pseudo-tty on the remote system. This means **rcmd** cannot run any command that needs to control your terminal.



1.2.5.1 Invoking rcmd(TC)

The syntax for the **rcmd** command is:

```
rcmd host [-l user] [-n] [command]
```

where:

<i>host</i>	The remote host on which you want <i>command</i> to run
<i>user</i>	The account on the host to use instead of the account name you used to login to your local host

For example:

- There are two hosts, *atlantis* and *corfu*
- You are logged into the host *atlantis* as the user **bubba**
- Both hosts have accounts for the users **bubba**, **forrest**, and **dan**
- Trusted access has been configured as needed

The following describes how to execute commands on these systems:

- To run the **ls(C)** command on the host *corfu*, enter:

```
rcmd corfu ls
```

- To run the **ls** command on the host *corfu* as the user **dan**, enter:

```
rcmd corfu -ldan ls
```



Common reasons for failure of the **rcp** command include:

Unknown host	You mistyped the name of the host, or your system cannot resolve host into an IP address
No route to host	Your system does not know the route to the specified host
Permission denied	You do not have trusted access configured between your host and one or more of the remote hosts

If the user name you specified does not have a password, trusted access is not required. The same is true for the other **r**-commands.

Note: Your account should always have a password if the machine you are using is on a network.



1.2.5.2 I/O redirection with **rcmd**(TC)

When an **rcmd**(TC) command is successfully executed on your local host, it passes its standard input (by default your keyboard) to the standard input of the command running on the remote host. Similarly, the standard output and standard error of the command running on the remote host is connected to the standard output and standard error of the **rcmd** running on your local host. This makes it appear as though the standard input, standard output, and standard error for the remote command are directly attached to your keyboard and screen. What happens when you want to perform I/O redirection?

I/O redirection can happen:

- On the local host
- On the remote host

By default, all I/O redirection happens on the local host. For example, the following command runs the **ls**(C) command on the host *corfu*:

```
rcmd corfu ls > myfile
```

However, before the **rcmd** command starts, the shell parses the command line and performs the indicated redirection (the standard output of **rcmd** goes to *myfile*). Because the standard output of the **ls** command on the host *corfu* is connected to the standard output of the **rcmd** command on your local host—and that is redirected to *myfile*—the effect is that the standard output of the **ls** command goes to *myfile* on your local host.

To direct the standard output of the remote command to a file on the remote host, you must quote the redirection symbol using single quotes, double quotes, or the backslash. Quoting the redirection symbol prevents the local shell from interpreting the redirection symbol, leaving it for the remote shell to interpret. For example, either of the following commands run the **ls** command on the host *corfu* and place the standard output in *myfile* on *corfu*:

```
rcmd corfu ls \> myfile  
rcmd corfu ls ">" myfile
```



Using this I/O redirection technique, you can perform a crude remote backup by entering the following:

```
find / -depth -print | cpio -pduvm | rcmd corfu dd ">" /dev/rct0
```

This command runs the **find**(C) and **cpio**(C) commands on the local host. The standard output of the **cpio** command is an archive. This archive output is treated as the standard input to the **rcmd** command. This is sent over to the host *corfu* where it is read by the **dd**(C) command and written to *corfu*'s tape drive (*/dev/rct0*).



1.3 Summary

This job aid presented the syntax for using the **telnet**(TC), **ftp**(TC), **rlogin**(TC), **rnp**(TC), and **rcmd**(TC) commands. It also covered how to configure trusted access for use with the **r**-commands.

In addition to the syntax for each command, this job aid discussed how to use each command including: file transfer and copy with **ftp** and **rnp**; remote login with **rlogin** and **telnet**; and remote command execution with **rcmd**.

It also provided functional examples using each command in a real-world environment. For example; using compression and **tar** options with **ftp**; performing I/O redirection with **rcmd** on both local and remote hosts; and copying files between remote machines.



1.4 Relevant Commands and Files

The following commands and files were presented in this job aid.

Relevant commands

cpio (C)	Copies files (copy in/copy out)
dd (C)	Copies files (disk-to-disk)
find (C)	Locates files in the file tree
ftp (TC)	Transfer files from one TCP/IP host to another.
rcmd (TC)	Executes commands on remote UNIX hosts and displays output on the local host.
rcp (TC)	Used to copy files between UNIX hosts
rlogin (TC)	Used to login to a remote UNIX host
telnet (TC)	Used to login to a remote host running TCP/IP

Relevant files

<i>.rhosts</i>	Used to set up user-level trusted access
<i>/etc/hosts.equiv</i>	Used to set up host-level trusted access



AT75009P000