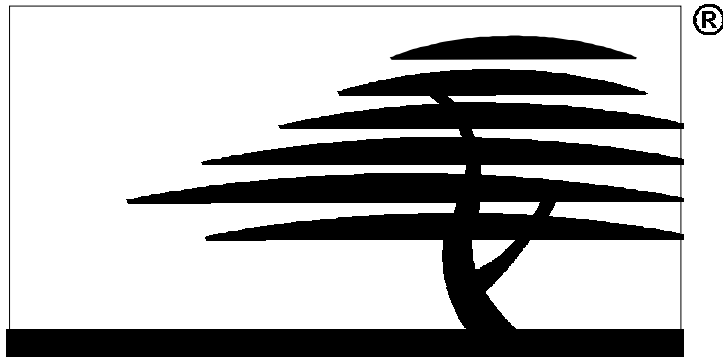




SCO

UnixWare® 7 Network Commands

Job Aid

Course Release: 1.0

For Software: UnixWare® 7

© 1987–1998 The Santa Cruz Operation, Inc. All rights reserved.

This publication is protected under copyright laws and international treaties.

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.

SCO, The Santa Cruz Operation, the SCO logos, VP/ix, SCO OpenServer, UnixWare, dbXtra, Panner, SCO Global Access, SCO MPX, SCO Doctor, SCO Doctor for Networks, SCO Doctor Lite, SCO Premier Motif, SCO Visual Tcl, SCO VisionFS, SCO ToolWare, SCO TermLite, SCO Vision 97, SCO CIFS Bridge, SCO WebTop, The Internet Way of Computing, and IWoC are trademarks or registered trademarks of The Santa Cruz Operation, Inc. in the USA and other countries. IXI, X.desktop, Deskworks, Wintif, IXI Panorama, Deskterm, and IXI Desktop are trademarks or registered trademarks of IXI Limited, a subsidiary of The Santa Cruz Operation, Inc. Visionware, Devkit.Vision, Esprit, Kodon, Super.Vision, Visionware Super.Vision, Term.Vision, Visionware Direction, the Visionware logo, Visionware SQL-Retriever, and XVision are registered trademarks of Visionware Limited, a subsidiary of The Santa Cruz Operation, Inc. SuperVision, TermVision, X-Visionware, SQL-Retriever, and Vision Builder are trademarks of Visionware Limited. UNIX is a registered 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. 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 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: Use, duplication, or disclosure by the US Government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of Rights in Technical Data and Computer Software Clause at DFARS 252.227-7013. Contractor/Manufacturer is The Santa Cruz Operation, Inc., 400 Encinal Street, Santa Cruz, CA 95060.

If this procurement is for a civilian government agency, this FAR Restricted Rights Legend applies:

RESTRICTED RIGHTS LEGEND: This computer software is submitted with restricted rights under Government Contract No. _____ (and Subcontract No. _____, if appropriate). It may not be used, reproduced, or disclosed by the Government except as provided in paragraph (g)(3)(i) of FAR Clause 52.227-14 alt III or as otherwise expressly stated in the contract. Contractor/Manufacturer is The Santa Cruz Operation, Inc., 400 Encinal Street, Santa Cruz, CA 95060.

The copyrighted software that accompanies this publication is licensed to the End User only for use in strict accordance with the End User License Agreement, which should be read carefully before commencing use of the software.

1

Network Commands



Lesson 1

Network Commands

Introduction

Two commands (**telnet**(1tcp) and **ftp**(1tcp)) are present in every implementation of TCP/IP. They allow you to log into a remote system and to transfer files to and from remote systems. Three additional commands (**rlogin**(1tcp), **rcp**(1tcp), and **rsh**(1tcp)) come with almost every UNIX[®] implementation of TCP/IP and perform similar functions to the **telnet** and **ftp** commands—they allow you to execute commands on remote systems without logging in.

This job aid is designed to provide you with the information you need to use the **telnet**, **ftp**, **rlogin**, **rcp**, and **rsh** commands effectively in your job. This will include command function and syntax.

This ***Job Aid*** also includes a discussion of how to set up trusted access for the ***r-commands***—the network commands that begin with an *r* (**rsh**, **rlogin**, and **rcp**). This is something you will need to set up before you can use the **rcp** and **rsh** commands; it can be useful for the **rlogin** command as well.

Objectives

This ***Job Aid*** covers the 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**
- Compare and contrast the **r-commands** with the **telnet** and **ftp** commands



1.1 TCP/IP-standard Commands

A technical specification describes what a vendor must include in a software package before it can be called TCP/IP. The specification says (among other things) that a vendor must include two commands:

telnet Allows users to log into a remote system as though their terminal was a terminal on the remote system

ftp Allows users to transfer files to and from a remote system

1.1.1 telnet(1tcp)

The **telnet(1tcp)** command allows you to log into a remote system as though your terminal was a terminal on the remote system.

Although every TCP/IP implementation has to have a **telnet** command, typically only multiuser systems, like UnixWare 7, accept **telnet** connections. For example, you can **telnet out** of a DOS system, but most implementations of TCP/IP for DOS do not allow you to **telnet into** the system. UnixWare 7 will accept **telnet** connections if the **telnetd(1Mtcp)** daemon is configured.

The **telnet** command in UnixWare 7 does not perform any *terminal emulation*. In other words, once you have established 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. Therefore, if you log into your UnixWare 7 system using an ANSI terminal, for example, and you **telnet** to another remote UNIX system, you must tell the remote UNIX system that you are using an ANSI terminal. Similarly, if you are using a Wyse-60 terminal, you would have to tell the remote UNIX system you are using a Wyse-60 terminal.



1.1.1.1 Invoking telnet(1tcp) with arguments

The syntax for the **telnet** command is given in the **telnet(1tcp)** manual page and is shown here:

```
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 will invoke **telnet** by entering either of the following:

```
telnet hostname  
telnet IP-address
```

where:

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

If the remote host is reachable and has a daemon to connect to, 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 <i>hostname</i> into an IP address.
No route to host	You mistyped the <i>IP-address</i> , or your system does not know the correct route to take to get to the specified <i>hostname</i> or <i>IP-address</i> .



1.1.1.2 Invoking telnet(1tcp) without arguments

Another way to invoke **telnet** is to enter the command without any arguments:

telnet

Without arguments, the **telnet** command provides you with a prompt:

telnet>

To connect with a remote host enter either of the following:

open *hostname*

open *IP-address*

where *hostname* and *IP-address* are just as they were described in Section 1.1.1.1 on page 1-7. There is no advantage to using this method over the previous method.

1.1.1.3 telnet(1tcp) commands

You can get to a **telnet** prompt (telnet>) by typing the **telnet** escape character. By default, this character is CTRL]. The escape character can be changed with the **-e** command-line flag.

In addition to the **open** command described above, there are other commands that you can give **telnet** to configure your environment. For example, if you are a remote logging into a mainframe, block-oriented terminal environment, you may need to use the **mode** command. As another example, if you need to change the software flow-control characters, you may need to use the **set start** and **set stop** commands.

You can get a complete list of **telnet** commands from either the **telnet(1tcp)** manual page or by entering a question mark (?) at the **telnet** prompt:

telnet> ?



1.1.1.4 Exiting a telnet(1tcp) session

To leave a **telnet** session, you can:

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

```
telnet> quit
```

1.1.1.5 telnet(1tcp) command-line flags

Although you will generally not need to use any of the command-line flags, they are available if you need them. See the **telnet(1tcp)** manual page for an explanation of the flags. What follows are reasons why you would use some of the flags:

- | | |
|------------------|--|
| -8 and -L | If you want to be able to use the IBM 8-bit character graphics, you need to set the -8 flag. If you only want 8-bit output and 7-bit input, you must use the -L flag instead. |
| -E | Use this flag to prevent people from accessing the telnet prompt (telnet>). This can be useful if telnet is being invoked from a script. |
| port | By default, the telnet command attempts to connect to the telnetd daemon on the specified host. However, telnet can also be used to connect to and converse with other daemons. This is most often done to test whether or not a daemon is configured properly. |

For example, to test the functionality of the SMTP mail daemon, enter:

```
telnet localhost 25
```

This is because 25 is the port number where the SMTP server listens for connections. A list of all system daemons and their port numbers can be found in the */etc/services* file.



1.1.2 ftp(1tcp)

The **ftp(1tcp)** command allows you to transfer files to and from a remote host using the File Transfer Protocol (FTP). Many sites on the Internet have set up repositories of programs and useful data that you can retrieve via *anonymous ftp*. Many web browsers (e.g., Netscape Navigator™ and NCSA Mosaic™) will even perform anonymous **ftp** transfers for you.

As with **telnet**, every TCP/IP system must have a command to FTP *out*, but not every system must allow you to FTP *into* the system. UnixWare 7 accepts FTP connections if the **ftpd** daemon is properly configured.

1.1.2.1 Invoking ftp(1tcp)

The syntax for the **ftp** command is given in the **ftp(1tcp)** manual page and is shown here:

```
ftp [-digitvn] [host]
```

As with the **telnet(1tcp)** command, you will generally invoke **ftp** in one of three ways:

```
ftp hostname  
ftp IP-address  
ftp
```

where:

hostname The resolvable name of a remote system you want to FTP to

IP-address The IP address (in dotted-decimal notation) of a remote system you want to FTP to

In the first two cases, you will be asked for a valid username and password on the remote host, provided that you do not encounter any errors. These errors are the same as those described for **telnet** (see Section 1.1.1.1 on page 1-7).

Without any command line arguments, you will receive the FTP prompt (**ftp>**).



1.1.2.2 Anonymous ftp

Some network administrators have configured *anonymous ftp*. This means that instead of supplying a real username and password, you can supply the word **anonymous** or **ftp** as the username, and your email address (e.g., *user1@somedom.com*) as the password. In this case, your FTP session will not have access to the entire file tree, but only a sub-tree of it.

1.1.2.3 ftp(1tcp) commands

You can issue any FTP command at the **ftp** prompt; these commands are described in the **ftp(1tcp)** manual page. To see a list of available commands, enter a question mark (?) at the FTP prompt:

```
ftp> ?
```

The available commands vary. The commands available on UnixWare 7 are listed in the following table:

Table 1: FTP Commands

!	delete	mdir	sendport	send
\$	debug	mget	put	site
account	dir	mkdir	pwd	size
append	disconnect	mls	quit	status
ascii	form	mode	quote	struct
bell	get	modtime	recv	system
binary	glob	mput	reget	sunique
bye	hash	newer	rstatus	tenex
case	help	nmap	rhel	trace
cd	idle	nlist	rename	type
cdup	image	ntrans	reset	user
chmod	lcd	open	restart	umask
clear	ls	prompt	rmdir	verbose
close	macdef	protect	runique	?
cr	mdelete	proxy	safe	



You will typically only use a few of these commands, including:

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

get *remote-name local-name*
get *remote-name*

where *remote-name* is the name of the remote file, and *local-name* is the name the file will have on the local system. If *local-name* is omitted, then *remote-name* is the name of both the remote file and local file.

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

put *local-name remote-name*
put *local-name*

where *local-name* is the name of the local file, and *remote-name* is the name the file will have on the remote system. If *remote-name* is omitted, then *local-name* is the name of both the local file and remote file.

You must have write permission in the directory in which you are trying to **put** the file. Some anonymous FTP sites have a directory called *Uploads* 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 with one command using wildcards that use the same form as shell patterns. Each file created will have the same name as the one retrieved.

mput Allows you to **put** multiple files with one command using wildcards that use the same form as shell patterns. Each file created will have 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 be displayed with every 1K bytes that are transferred between hosts. This is useful when transferring large files or when transferring across slow connections because it allows you to discern between a transfer that is still ongoing (i.e., pound signs are still being displayed) and one that is hung (i.e., no more pound signs are being displayed).
prompt	By default, when you transfer files with mget and mput , you will be prompted before each and every file transfer. The prompt command toggles this prompting. The first time you issue the prompt command, prompting is turned off. If you issue the prompt command again, prompting will be turned back on.
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 typing: !cd newdir This is because the ! command starts up a new shell. Therefore, in the case of the command above, 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	Causes the file transfer to perform certain translations on the file when it is transferred from one host to the other. See the ftp(1tcp) manual page for the list of conversions performed.
binary	Causes the file transfers to be full image transfers. No conversion of the data will take place.



1.1.2.4 Variations on the get and put commands

There are two issues concerning getting and putting files with **ftp**:

- Transferring uncompressed files across a network is wasteful of both your time and network bandwidth.
- Transferring all the files and subdirectories in a directory can be quite tedious.

Most modern FTP servers provide you with a way to work around one or both of these problems by allowing you to indicate that:

- The file(s) to be transferred should be compressed before transmission.
- All the files in a directory to be transferred should be put into a single **tar** file before transfer.

You can even combine the two approaches. You tell the FTP server to compress a file by getting or putting a given filename as usual, but add the file suffix “**.Z**” as the filename. You tell the FTP server to create a **tar** file by placing the file suffix “**.tar**” to the directory name. You tell the FTP server to do both by placing the file suffix “**.Z.tar**” to the directory name. For example:

- Given a file that you want to get called *myfile*, to cause the file to be compressed before transmission, issue the command:

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

The *myfile* file will be automatically uncompressed on your machine. This works for the **put** command, too.

- Given a directory that you want to get called *mydir*, to cause a **tar** file to be created and the entire contents of the directory to be transferred, issue the command:

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

The *mydir* directory will be placed into a **tar** file and transferred to your host. It will then be automatically converted back to normal files on your local host.



- Given a directory that you want to get called *mydir2*, to cause a **tar** file to be created and the entire contents of the directory to be compressed and transferred to your host, you issue the command

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

This *mydir2* directory will be placed into a **tar** file and transferred to your host. It will then be automatically uncompressed and converted back into normal files on your local host.

The only way to know whether or not this works with a particular FTP server is to try it. If you have access to that server's manual pages, it will be most likely be documented in the manual pages for the **ftpd** daemon or */etc/ftphosts* file. UnixWare 7 can be configured to support these features.

1.1.3 The *.netrc* file

If you do not want to have to enter you login name and password each time you **telnet** or FTP to a remote host, you can place this information in a file in your home directory called *.netrc*.

WARNING! Using a *.netrc* file is very dangerous because your login name and passwords for specific hosts must be placed in this file unencrypted. This violates most computer security principles.

The format of the *.netrc* file is given in the **netrc(4tcp)** manual page, which includes an example.

If you must use a *.netrc* file, you should be sure that it is readable and writable only by yourself. No other users, even those in your group, should have any access to the file, therefore the file's mode should be 600.



1.2 The *r*-commands and Trusted Access

Historically, the difficulty with the **telnet** and **ftp** commands is that you had to give your username and password for the remote host each time you used them. The only alternative was to use a *.netrc* file, which may create a security problem.

To counter these problems, two commands with essentially equivalent functionality were created: **rlogin**(1tcp) and **rcp**(1tcp). These commands allow you to set up and use trusted access. On remote hosts for which trusted access has been configured, you do not need to specify your password to log in or to copy files.

In addition, the **rsh**(1tcp) command has been provided, but has no equivalence to the **telnet** or **ftp** commands. The **rsh** command allows you to execute a command on a remote system and have the results displayed on your terminal. Trusted access also applies to the **rsh** command.

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

1.2.1 The **rlogin**(1tcp) command

The **rlogin** command serves the same function as the **telnet** command, however you may only use **rlogin** to login to systems running the UNIX operating system. UnixWare 7 will accept **rlogin** connections if the **rlogind**(1Mtcp) daemon is configured.

The **rlogin** command in UnixWare 7 does not perform any terminal emulation. In other words, once you have established a connection with a remote host, the **rlogin** command on UnixWare 7 passes the characters you type on your keyboard to the remote host without modification. Similarly, the **rlogin** command on UnixWare 7 displays any characters it receives from the remote host on your screen. Therefore, if you log into your UnixWare 7 system using an ANSI terminal, for example, and you **rlogin** to another remote UNIX system, you must tell the remote UNIX system that you are using an ANSI terminal. Similarly, if you are using a Wyse-60 terminal, you would have to tell the remote UNIX system you are using a Wyse-60 terminal.



1.2.1.1 Invoking rlogin(1tcp)

The syntax for the **rlogin** command is given in the **rlogin(1tcp)** manual page and is shown here:

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

In practice, you will rarely use all of these options. Generally you will invoke **rlogin** by entering either of the following:

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

where:

<i>hostname</i>	The resolvable name of a remote system you want to rlogin to
<i>username</i>	The login name to use on the remote host instead of the login name you are using on the local host

If the remote host is reachable and has a daemon to connect to, but trusted access has not been configured, you will see the password prompt for that system. To complete the login, type 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 <i>hostname</i> into an IP address
No route to host	Your system does not know the right route to take to get to the specified <i>hostname</i>
Login incorrect	You did not give the proper password for the <i>username</i> you used, or the <i>username</i> you used does not exist on the remote host

If trusted access is configured, or if the *username* you specified does not have a password, you will be logged in without having to specify a password. The same is true for the other **r**-commands. NOTE: Because of the **r**-commands, your account should always have a password if your machine is on a network.



1.2.1.2 Exiting a **rlogin(1tcp)** session

To leave a **rlogin** session, you can:

- Exit the login session with the remote system. This will automatically cause **rlogin** to clean up and exit
- Type the two-character escape character sequence, which is by default “~.” (tilde-dot). This will cause **rlogin** to clean up and exit.

1.2.1.3 **rlogin(1tcp)** command-line flags

Although you will generally not need to use any of the command-line flags, they are available if you need them. See the **rlogin(1tcp)** manual page for an explanation of the flags. What follows are reasons why you would use some of the flags:

- | | |
|------------|--|
| -8 | If you want to be able to use the IBM 8-bit character graphics, you need to set the -8 flag. |
| -7 | If you want to be able to use the 7-bit characters, you need to set the -7 flag. |
| -L | Tells the remote side not to do any post-processing of characters coming from the standard output on the remote host. This is useful, for example, if you do not want carriage return characters mapped to carriage-return/line-feed characters. |
| -E | Prevents people from exiting rlogin by using the escape sequence. This can be useful if rlogin is being invoked from a script. |
| -ec | Changes the escape character from a tilde (~) to the character, <i>c</i> , specified after the -e flag |



1.2.2 Configuring Trusted Access

Trusted access allows you to utilize the enterprise network in a way that is both more convenient and more secure.

If trusted access is not configured for the **rlogin**(1tcp) command, it will prompt 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 **rsh**(1tcp) and **rcp**(1tcp) commands will not work.

You can configure trusted access at the user level. Your system or network administrator can configure trusted access at the host level.



1.2.2.1 Configuring trusted host access

Trusted host access allows you to indicate that one host trusts all the users on another host that have identical login names. It trusts them to use the **rlogin**, **rcp**, and **rsh** commands—without specifying a password.

You must be **root** to configure trusted access at the host level. This configuration consists of editing the */etc/hosts.equiv* file, creating it if necessary. Each entry in */etc/hosts.equiv* specifies the remote system that is trusted.

The following shows the format and example entries in the */etc/hosts.equiv* file:

```
barbados
tortola bob
guam
```

If these entries exist on a machine called *corfu*, then they mean the following:

barbados	All 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 rsh to <i>corfu</i> as jane without specifying jane 's password on <i>corfu</i> .
tortola bob	Allows bob on <i>tortola</i> to log in as bob on this machine (<i>corfu</i>), except root .

The **root** account can never be equivalenced by an entry in the */etc/hosts.equiv* file. Refer to **hosts.equiv(4tcp)** 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.



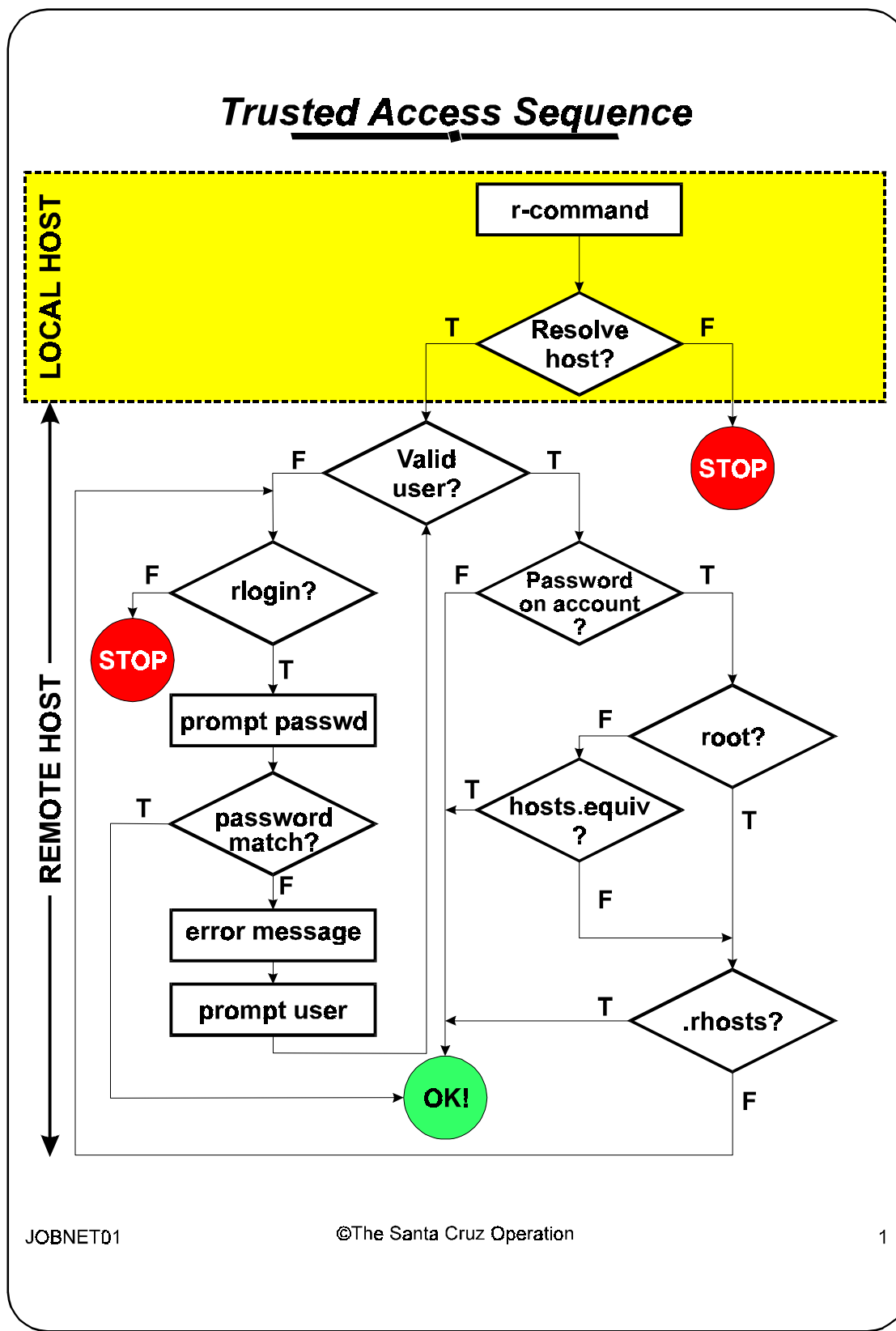
1.2.2.2 Configuring trusted user access

Trusted user access allows users on one system to indicate which users from another system they trust to use their account via the **rlogin**, **rsh**, and **rcp** commands, without providing a password. 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 yourself, or the system or network administrator may do it for you. The configuration consists of creating or modifying a file called *.rhosts* (note the leading dot) in a user's home directory. This file must be owned by either **root** or the owner of the home directory. It must also be writable only by the owner, otherwise it will be ignored.

The format of entries in the *.rhosts* file is the same as those in the */etc/hosts.equiv* file. The semantics are, however, slightly different. Referring back to the previous page, if those entries were put in the home directory of the user **jane** on the machine *corfu*, then they would 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 rsh 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 rsh to this machine (<i>corfu</i>) as jane , using the appropriate arguments to those commands, without specifying jane 's password.



1.2.2.3 The trusted access sequence

Whenever an **rlogin**, **rsh**, or **rcp** command is executed, the systems involved go through a set procedure to determine whether trusted access has been configured. The diagram opposite depicts this procedure.

1.2.2.3.1 Trusted access on the local host

The following steps are taken on the local host:

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

host_name: Host name lookup failure
- Step 4. If an address for the remote host can be resolved, the **r**-command connects through the TCP/IP transport to the remote host.

1.2.2.3.2 Trusted access on the remote host

The following steps are then taken on the remote host:

- Step 1. The user's account name is looked up in the password database.
- Step 2. If the user's account name is not found in the password database, the **rsh** and **rcp** commands will fail, and the **rlogin** command will prompt for a password (that is how the login program functions), and then fail.
- Step 3. The user's account is checked to see if it has a (encrypted) password.
- Step 4. If the user's account does not have a password, the **r**-command executes.
NOTE: This is very dangerous. You should never allow users to not have passwords on your systems, especially if your enterprise network is attached to the Internet.



- Step 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 the */etc/hosts.equiv* file contains the name of the local host, the **r**-command executes.
- Step 6. If the account name check from Step 5 was **root**, or if the account name was not **root** and the */etc/hosts.equiv* check failed, then the *.rhosts* file in the remote user's home directory is checked to see if it contains the local host's name, and possibly the local user's name, depending upon how the **r**-command was invoked.
- Step 7. If the *.rhosts* file contains the desired entry, then the **r**-command executes.



1.2.3 The rcp(1tcp) command

The **rcp**(1tcp) command transfers files from one system to another and is similar to the **ftp**(1tcp) command. Some differences between **rcp** and **ftp** include:

- **rcp** 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 that trusted access be set up, unless the target accounts on the non-local hosts do not have a password set. The **rcp** command will not ask for a password they way **rlogin**(1tcp) and **ftp** do.
- Because of the overhead in authenticating each **rcp** command, if you are copying many files (and not using the **-r** flag), it is faster to use the **ftp** command

1.2.3.1 Invoking rcp(1tcp)

The syntax for the **rcp** command is given in the **rcp** (1tcp) manual page and is shown here:

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

where filenames (*file1*, *file2*, and *directory*) all 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 the <i>pathname</i> is not an absolute path, then it is relative to the home directory of the account on <i>host</i> that has the same name as the account name with which you logged into the local host.
<i>user@pathname: host</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 the <i>pathname</i> is not an absolute path, then it is relative to the home directory of the account <i>user</i> on <i>host</i> . Also, the transfer's permissions and trusted access will based upon <i>user</i> 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 has been 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*, you would 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, you would enter:

```
rcp barbados:treasure treasure
```

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

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



NOTE: The directories and files on the target machines do not have to exist before they can be copied to. 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 <i>hostname</i> into an IP address
No route to host	Your system does not know the right route to take to get to the specified <i>hostname</i>
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: Because of the **r**-commands, your account should always have a password if the machine you are using is on a network.



1.2.3.2 rcp(1tcp) command-line flags

See the **rcp**(1tcp) manual page for an explanation of the flags. What follows are reasons why you would use some of the flags:

- p** Use this flag if you want **rcp** to maintain modification times and permissions (i.e., the mode) of the source files. This can be useful when relocating directories from one system to another.
- r** You should use this flag whenever you want to copy the entire contents of a directory from one host to another, as in the last example in the previous section.

1.2.4 The rsh(1tcp) command

Using the **rsh**(1tcp) command is similar to using **rlogin**(1tcp) to access a remote system and only running 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**(1), **more**(1), and **pg**(1) to control your terminal from across the network. Without a pseudo-tty, these commands will not work properly across the network.

The **rsh** command does not cause a pseudo-tty to be created on the remote system. This means that you cannot run via **rsh** any command that needs to control your terminal. There is a possible work-around, however.

If you are using the X Window System™ environment, you can use **rsh** to start a terminal emulator window on the remote host (e.g., **xterm**(X1)). Most terminal emulators allow you to specify a flag (**-e** for **xterm**) to give the terminal emulator a command to run.



1.2.4.1 Invoking rsh(1tcp)

The syntax for the **rsh** command is given in the **rsh(1tcp)** manual page and is repeated here:

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

where:

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

For example, given the following:

- 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**(1) command on the host *corfu*, enter:

```
rsh corfu ls
```

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

```
rsh corfu -ldan ls
```

- To run the **vi**(1) command on the host *corfu* in a **xterm**(X1) window, enter:

```
rsh corfu xterm -e vi &
```

If the ampersand (&) is not present on the command line, then the window that runs this command is unusable until **vi** stops in the new window created by this command.



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

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

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

NOTE: Because of the **r**-commands, your account should always have a password if the machine you are using is on a network.



1.2.4.2 I/O redirection with rsh(1tcp)

When an **rsh(1tcp)** 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 **rsh** 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 as normal. What happens when you want to perform I/O redirection?

I/O redirection can happen in one of two places:

- 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*:

```
rsh corfu ls > myfile
```

However, before the **rsh** command starts, the shell parses the command line and performs the indicated redirection (the standard output of **rsh** goes to the *myfile* file). Since the standard output of the **ls** command on the host *corfu* is connected to the standard output of the **rsh** 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 get the standard output of the remote command to go 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 causes the local shell not to interpret the redirection symbol, leaving it for the remote shell to interpret. For example, either of the following commands will run the **ls** command on the host *corfu* and place the standard output of the **ls** command in *myfile* on *corfu*:

```
rsh corfu ls \> myfile  
rsh 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 | rsh 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 **rsh** command. This in turn 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**(1tcp), **ftp**(1tcp), **rlogin**(1tcp), **rnp**(1tcp), and **rsh**(1tcp) commands. It has also covered how to configure trusted access for use with the **r**-commands, and discussed the danger of using the *.netrc* file.

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 **rsh**. The flags of each command were also discussed from a functional perspective to augment the manual pages as necessary.

This *Job Aid* also displayed some functional examples of using each command in a real-world environment, e.g., using compression and tar options when doing FTP; using **xterm** with **rsh** to run commands that require terminal control; performing I/O redirection with **rsh** on both local and remote hosts; and performing remote copies between remote machines, not just between a local and a remote host.



1.4 Relevant Commands and Files

The following commands and files were presented in this *Job Aid*.

Relevant commands

cpio (1)	Copy-in/copy-out command. Often used for performing backups.
dd (1)	Disk-to-disk copy command. Often used when performing backups across the network with the rsh command.
find (1)	Locates files in the file tree
ftp (1tcp)	File Transfer Protocol command. Used to transfer files from one TCP/IP host to another.
inetd (1Mtcp)	The super-daemon. Starts other daemons as they are needed.
rsh (1tcp)	Remote command. Used to execute commands on remote UNIX hosts and have the output displayed on the local host.
rcp (1tcp)	Remote copy. Used to copy files between UNIX hosts
rlogin (1tcp)	Remote login. Used to login to a remote UNIX host
rlogind (1Mtcp)	The daemon that performs the same function as getty (1M) for network logins using rlogin .
telnet (1tcp)	Used to login to a remote host that is using TCP/IP
telnetd (1Mtcp)	The daemon that performs the same function as getty (1M) for network logins using telnet



Relevant files

<i>.netrc</i>	Used to obviate the need for people to type their username and password when using telnet or ftp to remote hosts. NOTE: This file can create a significant security hole.
<i>.rhosts</i>	Used to set up user-level trusted access
<i>/etc/hosts.equiv</i>	Used to set up host-level trusted access
<i>/etc/services</i>	Contains the port numbers of all well-known services

