# CONVEX Internet Services User's Guide

Order No. DSW-141

First Edition
November 1990

CONVEX Computer Corporation
Richardson, Texas USA

# CONVEX
# Internet Services
# User's Guide

Order No. DSW-141



Printed in the United States of America

# Revision Information for

# CONVEX Internet Services User's Guide

| Edition | Document No. | Description |
|---|---|---|
| First | 710-002530-204 | Released with CONVEX Internet ServicesV9.0, November 1990. Modified to include additional `ftp` and `telnet` commands. This manual was previously entitled *CONVEX Networking Utilities User's Guide*. |

# Table of Contents

## CONVEX Internet Services User's Guide

# Using This Guide

## Purpose and Audience

This document is for users who are unfamiliar with CONVEX Internet Services. It describes in simple terms networking utilities that enable you, among other things, to log in to remote systems, execute commands on different machines, and transfer files from one machine to another.

## Organization

This guide is organized as follows:

- Chapter 1, "Introduction," briefly describes the CONVEX Internet Services product and the networking services it provides.
- Chapter 2, "Accessing Hosts on a Network," describes files used to define hosts on a network and to grant access to users.
- Chapter 3, "Using Berkeley Utilities," describes utilities used to communicate with ConvexOS or UNIX systems that run BSD networking. Topics discussed include how to log in, copy files, execute commands, and check status on remote machines.
- Chapter 4, "Using DARPA Internet Utilities," describes utilities used primarily for communicating with systems that use basic TCP/IP protocols, but do not run BSD networking, You can also use these utilities to communicate with ConvexOS systems.
- Appendix A lists the syntax of each utility discussed in this guide.
- Appendix B provides instructions for reporting software or documentation problems to the CONVEX Technical Assistance Center (TAC).

## Technical Assistance

If you have questions that are not answered in this book, contact the CONVEX Technical Assistance Center (TAC). To contact the TAC, use one of the following phone numbers:

- Within the continental U.S., call 1(800)952-0379.
- From Canada, call 1(800)345-2384.
- All other locations, contact the local CONVEX office.

## Associated Documents

Using this software successfully may require more information than is contained in this manual.

CONVEX Computer Corporation provides the following documents to help you with the ConvexOS operating system:

- *CONVEX UNIX Primer*, DSW-133, provides an introduction for users who have not previously used the *ConvexOS* operating system.
- *ConvexOS Programmer's Reference*, DSW-332, is the standard reference for the ConvexOS operating system. It contains manual pages describing the set of CONVEX Internet Services software.

CONVEX Computer Corporation provides the following documents to help you with CONVEX Internet Services:

- *CONVEX Networking Concepts*, DSW-128, provides an overview of CONVEX networking products.
- *CONVEX Internet Services System Manager's Guide*, DSW-142, provides system managers and operators with information needed to configure and maintain a CONVEX Internet Services TCP/IP network.
- *CONVEX Interprocess Communication (IPC) Programming Guide*, DSW-143, provides application programmers with the necessary information to develop network applications.

## Ordering Documentation

To order the current edition of this or any other CONVEX document, send requests to:

CONVEX Computer Corporation
Customer Service
PO Box 833851
Richardson, TX 75083-3851 USA

Include the order number or the exact title, as listed on the front cover.

# Notational Conventions

This section discusses notational conventions used in this guide.

## Command Syntax

Consider this example:

```
COMMAND input_file [...] {a | b} [output_file]
   ①        ②      ③     ④          ⑤
```

1. **COMMAND** must be typed as it appears.
2. *input_file* indicates a file name that must be supplied by the user.
3. The horizontal ellipsis in brackets indicates that additional input file names may be supplied.
4. Either `a` or `b` must be supplied.
5. *output_file* indicates an optional file name.

## General Conventions

In general, the following conventions are used in this guide:

- **`Bold constant-width font`** identifies user input in examples.
- *Italics*
  - Designate user-supplied variables in a command-line example.
  - Introduce new and important terms.
  - Identify variables in mathematical equations.
  - Indicate titles of documents.
- `Constant-width font` is used to designate input and output, including:
  - Command names and options.
  - System calls.
  - Data structures and types.
  - Directives, program statements, display examples, printout examples, and error messages.
- Horizontal ellipsis (...) shows repetition of the preceding item(s).
- Vertical ellipses show that lines of code have been left out of an example.
- Words and abbreviations that indicate keyboard keys you press are identified in a distinctive bold type. For example, **RETURN** refers to the carriage return key. Words separated by a hyphen indicate two keys that you press simultaneously. For example, **CTRL-X** indicates that you press and hold down the **CTRL** key and then press the **X** key.
- The word "enter" in a phrase such as "enter **`ls`**" means that you type the command and then press **RETURN**.
- Characters you type that are not echoed by the system appear in reverse text. For example, a user-supplied password appears as password.

- References to the *ConvexOS Programmer's Reference* appear in the form `adb(1)`, where the name of the man page is followed by its section number enclosed in parentheses.
- The ConvexOS prompt is printed as a percent sign (%). The pound sign (#) signifies the superuser prompt.

## Note

**A Note highlights information that may be of particular interest.**

## Reader Response

If you have comments or questions about the contents of this book, please notify the CONVEX documentation department by using the Reader's Comments form at the end of this document.

# Introduction 1

This guide is an introduction to using CONVEX Internet Services. No attempt at a comprehensive explanation has been made. If you require a technical summary or reference manual, refer to the *ConvexOS Programmer's Reference*, which contains man pages for the complete set of network services.

CONVEX Internet Services includes all of the utilities, databases, and protocols necessary to configure, maintain, and use TCP/IP protocols over Ethernet and HYPERchannel interfaces. If your installation is licensed for the optional CONVEX UltraNet Interface, CONVEX Internet Services also allows you to run TCP/IP over UltraNet communications hardware.

The best way to learn how to use the services described here is to try the examples as you read along. Before you do, however, check with your system manager to make sure the network is up and running, and the various daemons needed by the utilities have been started. Your system manager can also help you interpret any error messages you receive.

If you become interested in networking, you may want to read the *CONVEX Internet Services System Manager's Guide* and the *CONVEX Interprocess Communication Programming (IPC) Guide*. These manuals describe how to configure and maintain the network and how to write network application programs. For general information about CONVEX networking products, refer to *CONVEX Networking Concepts*.

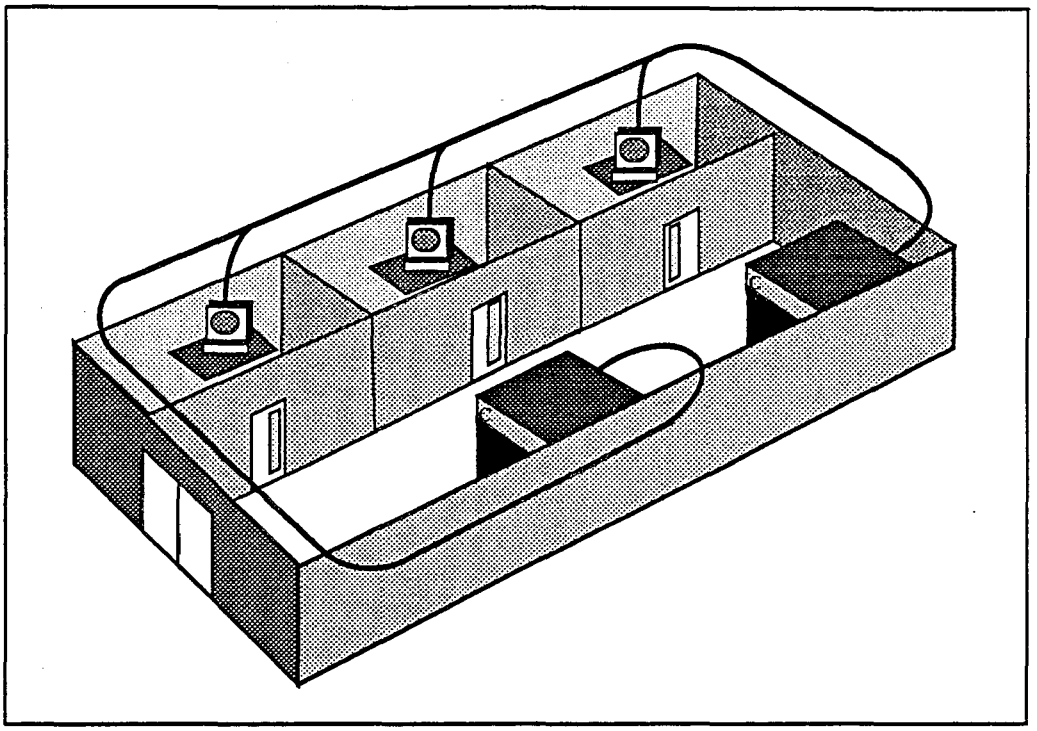
## CONVEX Internet Services

In the simplest case, a computer network is two computers (or even terminals) connected by communication lines. A common case of a simple network is a local area network (LAN), used to link machines in close proximity to one another. At the most complex level, a computer network might be a worldwide system of computers joined by telephone system trunk lines or by satellite links. The DARPA Internet, developed and maintained by the Defense Advanced Research Projects Agency, is a good example of this type of network. The DARPA Internet uses both leased telephone lines and satellite links to relay technical and scientific information worldwide. Other large networks, spanning continents in many cases, are maintained by technical firms, governments, major universities, and commercial time-sharing services.

Another large network that extends throughout the world is the UNIX-to-UNIX Communication Protocols (UUCP) network. UUCP connects UNIX systems via telephone lines. This network is used for sending files to and receiving files from other UNIX systems. However, you cannot log in to other machines on the UUCP network.

CONVEX Internet Services provides a LAN that can be connected to large external networks like the DARPA Internet or the UUCP network. You can log in to other machines on the LAN, copy files from one machine to another, check system status on the network, and perform various other tasks. A model of a simple CONVEX LAN is shown in Figure 1-1.

**Figure 1-1**
Model of a Simple CONVEX LAN



Machines on this network are not linked directly to each other. Instead, they are connected to a common communication line. The network shown uses coaxial cable as a communication line. Other network hardware includes UltraNet fiber-optic cables, HYPERchannel adapters, and Ethernet controllers.

Because the computers on this network are linked to a common communication line, users may have access (depending on protection and password schemes) to any machine on the LAN. Users may also have access to any external networks to which the CONVEX LAN is connected.

For information about UUCP, the mail system, and the line printer system, refer to *Managing ConvexOS: Configuration Guide* and *Managing ConvexOS: Operations Guide.*

# Accessing Hosts on a Network

# 2

This chapter describes files used to define hosts on a network and to grant network access to users.

## Host Name Database

To use programs and utilities included with CONVEX Internet Services, you must know names of other hosts with which you need to communicate. To obtain a list of host names, consult your system manager or look in the host name database. The host name database contains logical host names and addresses for other networked machines. It either resides in the */etc/hosts* file or is maintained by the Berkeley Internet Name Domain (BIND) system. (If your system runs NFS, you can configure the Yellow Pages Service (YP) to maintain a distributed host database. NFS is an optional product. For information about configuring YP, refer to the *CONVEX Network File System System Manager's Guide.*)

The first few lines of the /etc/hosts file are similar to those in Figure 2-1.

**Figure 2-1**
Sample /etc/hosts File

```
# Berkeley Host Database
#
#
127.1      localhost
#
# The format is:
#      internet-address official-hostname aliases...
#
# Local Net 130.50 -- 10Mb/s Ethernet
#
130.50.4   convex1 dopey
130.50.5   convex2 sleepy
130.50.6   convex3 sneezy
130.50.7   convex4 doc
130.50.9   convex5 bashful
130.50.10  convex6 grumpy
```

This file consists of two types of data: internet addresses in "dot" notation (`130.50.9`) and logical host names (`convex5` and `bashful`).

Host names consist of two parts: the "official" host name and the alias(es). For example, in the host name:

```
convex6 grumpy
```

`convex6` is the official name, and `grumpy` is an alias. An alias enables you to reference a particular host with a symbolic name. Names are usually easier to remember than numbers, and you can make the aliases descriptive of the purpose for which each machine is used.

If your system uses a name server, host name data may or may not reside on the local system, depending on the type of server configured. Consult the *CONVEX Internet Services System Manager's Guide* for information about locating host names and addresses maintained by the name server.

## Network Access Files

Each machine on the network can accept or refuse logins by remote users. The system manager specifies whether users logging in from other machines must supply a password, or even if remote logins are permitted on a machine. Two files control this type of access: */etc/hosts.equiv* and *.rhosts*. (If your system runs NFS, the Yellow Pages Service (YP) uses the /etc/netgroup file to control user access. NFS is an optional product. For information about configuring YP, refer to the *CONVEX Network File System System Manager's Guide.*)

The /etc/hosts.equiv file contains a list of machine names that you can access from the local machine without using a password (using the same login name on the remote machines). When you use `rlogin` to remotely log in to a machine that is not listed in the /etc/hosts.equiv file, you are prompted for a password unless you are logged in as root. Figure 2-2 shows a sample /etc/hosts.equiv file.

**Figure 2-2**
Sample /etc/hosts.equiv File

```
convex1
dopey
sleepy
sneezy
doc
bashful
grumpy
```

## Note

**For security reasons, access to the root account is not controlled by /etc/hosts.equiv. You must either have an appropriate entry in root's /.rhosts file, or you must supply the root password when you log in remotely.**

Each user's home directory may also contain a private equivalence list in a file called .rhosts for use by `rlogin`. Entries in this file contain remote host names (not aliases) and user names specifying which users on which machines are permitted to log in without supplying passwords. For example, if you use different login names on different machines, add entries to your .rhosts file containing your other login names and the names of the machines from which you want to access your local account. The example in Figure 2-3 on the following page shows a .rhosts file for a user named `rsmith` on the local machine who uses login names `smith` and `bobsmith` on remote machines.

Figure 2-3
Sample .rhosts File

```
aries smith
aries bobsmith
```

In the above example, both `smith` and `bobsmith` may access `rsmith`'s account on `aries` without entering a password.

**Note**

**Your .rhosts file will not be honored if it has either group or world write permissions. You must use official host names (the first name listed in /etc/hosts file entries), not aliases, in the .rhosts file.**

# Using Berkeley Utilities 3

This chapter describes how to use utilities designed specifically for communicating with other UNIX or ConvexOS systems. These utilities—`rlogin`, `rcp`, `rsh`, `rwho`, `rdist`, and `ruptime`—are very easy to use (some do not even require arguments). You can find more information about these utilities in the *ConvexOS Programmer's Reference.*

## Using rlogin

The *rlogin* utility allows you to remotely log in to other ConvexOS or UNIX systems. Unlike `telnet`, discussed in the next chapter, `rlogin` preserves window sizes, which is useful if you are using a window manager such as Suntools.

To use `rlogin`, enter the command and the name of the machine to which you want access. For example, if you want to remotely log in to a machine called `aries`, enter

```
rlogin aries
```

The remote system may or may not prompt you for a password, depending on what is specified in the /etc/hosts.equiv file on the local machine or the .rhosts file in your home directory on the remote machine. If you are prompted for a password, enter the password you use on the machine to which you are logging in.

Log out from the remote machine by entering either **CTRL-D** or `logout`.

You can use `rlogin` to get from one machine to another in a series. If you lose track of which machine you are currently logged in to, use the `hostname` command to display the name of the current machine, as shown in the screen in Figure 3-1 on the following page.

Figure 3-1
Remotely Logging In to Multiple Machines

```
% hostname
aries
% rlogin gemini
Last login: Thu Aug 20 08:56:49 from aries
% hostname
gemini
% rlogin taurus
Last login: Mon Aug 17 14:16:31 from gemini
% hostname
taurus
% logout
Connection closed.
% hostname
gemini
% logout
Connection closed.
% hostname
aries
%
```

## Aborting and Suspending Remote Logins

When you log in to a series of machines as shown in Figure 3-1, logging out returns you to the previous machine in the series. A quicker way to get back to the original machine is to use the tilde period (~.) command. This command aborts all `rlogin` processes and returns you to the original machine, as shown in Figure 3-2.

Figure 3-2
Aborting a Remote Login

```
aries% rlogin gemini
Last login: Thu Aug 20 08:56:49 from aries
gemini% rlogin taurus
Last login: Mon Aug 17 14:16:31 from gemini
taurus% ~.
Connection closed.
aries%
```

~CTRL-Z suspends the remote login, and `fg` resumes it. When you suspend a remote login, you are returned to the command level on your local machine, as shown in Figure 3-3.

Figure 3-3
Suspending and Resuming a Remote Login

```
aries% rlogin gemini
Password: password
Last login: Fri Dec 18 15:19:34 from aries
ConvexOS V8.0 installed 10/31/89.
gemini% cd test.dir
gemini% ~CTRL-Z
Stopped
aries% cd
aries% fg
rlogin gemini
RETURN
gemini% logout
```

**Note**

**The character used as the suspend command depends upon which shell you use on the local system. You must type the suspend command as the first character in the command line.**

## Logging In to a Machine With No Home Directory

If you remotely log in to a machine on which you have no home directory, the system assigns you a default home directory of "/" and displays an error message. If you think that you should have a home directory on a machine and you see the error message shown in Figure 3-4, ask your system manager for an explanation.

Figure 3-4 is a sample `rlogin` session where the user has no home directory on `harpo`.

Figure 3-4
Remote Login with No Home Directory

```
aries% rlogin harpo
Password: password
No directory! Logging in with home=/
Last login: Fri Dec 18 13:43:34 from aries
ConvexOS V8.0 installed 10/31/89
harpo% pwd
/
harpo%
```

## Logging In Under Different User Names

To use different user names on different machines, you must specify the appropriate user name when you use `rlogin`. To do this, precede the user name with the `-l` option as shown in Figure 3-5.

Figure 3-5
Remote Login Under a Different User Name

```
aries% whoami
jones
aries% rlogin harpo -l bjones
Password: password
Last login: Fri Dec 18 13:43:34 from aries
ConvexOS V8.0 installed 10/31/89
harpo% whoami
bjones
harpo%
```

In this example, the user begins on `aries`, and then uses `rlogin` to work on a machine called `harpo`. The user logs in to `harpo` with the user name `bjones`.

## rlogin Error Messages

When `rlogin` cannot establish a connection to the requested host, you receive an error message and remain logged in to the local machine. The screen in Figure 3-6 shows typical error messages.

Figure 3-6
`rlogin` Error Messages

```
% rlogin ariel
ariel: unknown host
% rlogin testhost
testhost: Connection refused
% rlogin simon
simon: Network unreachable
% rlogin dorian
dorian: Connection timed out
%
```

The error messages in the screen have the following meanings:

| | |
|---|---|
| `unknown host` | The remote machine is not defined in the host name database. |
| `Connection refused` | The remote machine exists in the host name database, but the host returned a message refusing the connection. |
| `Network unreachable` | There is no routing table entry for the remote machine. |
| `Connection timed out` | The remote host did not respond to the request for a connection. This message usually indicates that the remote host is down. |

If you receive an error message when trying to access a host you believe to be on the network, ask your system manager for an explanation.

## Copying Files Between Systems

The *rcp* utility copies files to or from other systems that support BSD networking. Access to files on remote systems is controlled in the same way as with the `rlogin` command. You must have a .rhosts or /etc/hosts.equiv file entry to use `rcp`. In other words, to use `rcp`, you must be able to `rlogin` to the remote system without supplying a password.

To copy a file from a remote machine, enter the following command sequence:

`rcp` *remote_host:filename local_file*

**Note**

**Remote relative pathnames are interpreted relative to your remote home directory.**

For example, enter the following command sequence to copy the file groucho from your home directory on the remote host `duck_soup` to the local file chico.

```
rcp duck_soup:groucho chico
```

Copy files from the local machine to the remote machine by reversing the position of the arguments. For example, to copy the contents of the local file chico to the file groucho in your home directory on the remote machine, enter:

```
rcp chico duck_soup:groucho
```

You can copy an entire directory subtree by using the `-r` (recursive) option. You may use `rcp` to copy more than one file providing the destination is an existing directory. For example, the following command copies all the contents of the local directory tuna_fish into the directory big_store on the remote host `harpo`.

```
rcp -r tuna_fish harpo:big_store
```

If the destination directory on the remote host does not exist, it is created.

You can use wildcards to copy files with `rcp`. For example, if you want to copy all of your FORTRAN programs from a local directory to a directory on another machine, use a command similar to the following:

```
rcp *.f harpo:fortran_dir
```

In the previous example, all the FORTRAN source files in the current directory on `aries` (those ending in `.f`) are copied to the directory called fortran_dir on the remote machine `harpo`.

You can also use wildcards to copy all of the FORTRAN programs from a directory on a remote machine to a local directory. To do so, enter

`rcp harpo:fortran_dir/\*.f` *local_dir*

You can enclose the host and file name in quotes, as in

`rcp 'harpo:fortran_dir/*.f'` *local_dir*

in which case you do not use the backslash (" \").

To copy files to and from a remote machine on which your user name is different from that on the local machine, use a command similar to the following:

**rcp** *local_file remote_user@remote_host:remote_file*

## Executing Commands on a Remote System

Use the `rsh` utility to execute commands across the network. For example, to compile a FORTRAN program called cad1.f on a remote host named `gemini`, enter

```
rsh gemini fc cad1.f
```

If you use a different login name (for example, `bjones`) on the other system, use the `-l` option as you did for `rlogin`. For example:

```
rsh gemini -l bjones fc cad1.f
```

If you do not specify a command, you are logged in to the remote host.

Access to remote systems via the `rsh` command is controlled in the same way as with the `rlogin` command. You must have a .rhosts or /etc/hosts.equiv entry to use `rsh`. In other words, to use `rsh`, you must be able to `rlogin` to the remote system without supplying a password.

You cannot run interactive commands by using `rsh`.

## Checking the Status of a Remote System

Use the `ruptime` command to check the status of other machines on your local network, or on your local subnet if your network is divided into subnets. To use `ruptime`, enter

```
ruptime
```

Your screen displays output similar to Figure 3-7.

Figure 3-7
Sample `ruptime` Output

```
aries      up   19+21:15,  16 users,  load 2.91, 1.10, 1.51
gemini     up   7:20,      31 users,  load 6.37, 5.10, 5.49
harpo      up   33+11:32,   0 users,  load 0.00, 0.00, 0.00
groucho    down 2:35
```

Output from `ruptime` is similar to that produced by the `uptime` command, except that the `ruptime` notation shows the days of `uptime` as a number followed by a plus sign (+). For example, in the first entry above, the machine called `aries` has been up for 19 days, 21 hours, and 15 minutes. You can tell at a glance how long each system has been up, how many users are currently logged in, and what the load averages for remote machines are.

### Note

**Because some repeaters and routers do not recognize broadcast messages, the `ruptime` command may not work for hosts other than those on your local network or subnet.**

## Listing Remote System Users

If your network has broadcast capabilities, you can use *rwho* to determine who is remotely logged in to the machines on your local network, or your local subnet if your system uses subnetting. To use `rwho`, enter

```
rwho
```

Your screen displays output similar to Figure 3-8.

Figure 3-8
Sample `rwho` Output

```
chekhov  aries:tty30      Mar 25 08:46 :23
kirk     gemini:tty04     Mar 25 09:48
mccoy    gemini:tty05     Mar 25 08:46 :05
scotty   aries:ttyi2      Mar 25 11:28 :01
spock    gemini:tty16     Mar 25 08:03
sulu     aries:ttyi0      Mar 25 09:38 :57
uhura    gemini:tty0c     Mar 25 10:37 :22
```

Various machines and terminal lines are specified in the output. In the example above, data has been returned from two machines, `aries` and `gemini`. The date and time of when each user logged in, followed by the idle time in minutes are displayed. Users idle for more than an hour are not shown unless you specify the `-a` option.

You can list remote users in reverse alphabetical order by specifying the `-r` switch.

### Note

**Because some routers do not recognize broadcast messages, the `rwho` command may not work for hosts other than those on your local network or subnet. The `rwho` command only works for hosts that run the `rwho` daemon (`rwhod`).**

## Distributing Files to Remote Systems

The *rdist* program allows you to maintain identical copies of files on multiple hosts. By default, `rdist` distributes only those files that are older on the remote host than on the local host. It does so by comparing the last modification time and the file size of the local file to those of the remote file.

You can run `rdist` commands from the command line or pass it a file of commands. If you do not supply the name of a file containing commands, `rdist` looks for commands in a file named "distfile," then in "Distfile."

Typical entries in the command file have one of the following formats:

*variable_name* = *name_list*

[*label*:] *source_list* -> *destination_list*
*command_list*

where:

| | |
|---|---|
| *variable_name* | Name assigned to the *name_list* for later reference. |
| *name_list* | List of files, directories, or hosts *command_list* will act upon. |
| *label* | Optional identifier for later reference. |
| *source_list* | List of files or directories on the local host to use as master copies for distribution. |
| *destination_list* | List of hosts to which the master copies are distributed. |
| *command_list* | `rdist` commands. (Refer to the `rdist`(1) man page in the *ConvexOS Programmer's Reference* for a complete list of commands.) |

The first format assigns to *variable_name* the list given as *name_list*. For example, the command

```
FILES = (.login .cshrc .mailrc .logout .rhosts )
```

assigns a list of files to the variable `FILES`. Likewise,

```
HOSTS = ( aries gemini libra pisces )
```

assigns a list of host names to the variable `HOSTS`. Variable definitions are used in subsequent `rdist` commands.

Use the second format to distribute files to other hosts. *source_list* specifies a list of files or directories on the local host which `rdist` is to use as the master copy for distribution. *destination_list* lists hosts to which the master files are copied. For example, the command sequence:

```
dotfiles: ${FILES} -> ${HOSTS}
        install;
```

distributes copies of the defined FILES to the hosts defined as HOSTS. You can use this method to distribute your control files to all systems on which you have an account.

Figure 3-9 shows an `rdist` command file to update `jjones'` control files on all systems on which she has an account.

Figure 3-9
Sample distfile

```
FILES = ( .login .cshrc .mailrc .logout .rhosts )
HOSTS = ( aries gemini libra pisces )
dotfiles: ${FILES} -> ${HOSTS}
         install -y;
```

The `-y` option prevents updating remote files that have been modified more recently than their corresponding source files.

The `rdist` program also provides a better alternative to using `rcp` to copy files from one system to another. By using `rdist`, you can preserve the owner, group, permissions, and time of last modification. For example, to copy the contents of the directory tuna_fish into the directory big_store on the remote host harpo, use either of the following commands:

```
rcp -r tuna_fish harpo:big_store
rdist -c tuna_fish harpo:big_store
```

The second command preserves information about the file, whereas the first command creates new ownership, permission, and modification information.

The `rdist` command has many more options than those discussed here. Refer to the `rdist`(1) man page for more detailed information.

# Using DARPA Internet Utilities

# 4

This chapter describes utilities used to communicate with systems that do not run BSD networking, but use basic TCP/IP protocols. Examples are given for only a few of the available commands for each utility. Refer to the *ConvexOS Programmer's Reference* for more information about these utilities.

## Using telnet

The `telnet` program provides a user interface to the TELNET protocol, a protocol used for machine-to-machine communication. Like `rlogin`, `telnet` opens connections to other machines. Unlike `rlogin`, however, which makes connections only between systems running BSD networking, `telnet` is used to connect to any machine that runs the TELNET protocol, regardless of its operating system or architecture. For connections between machines running BSD networking, `rlogin` is the preferred method, because it preserves window sizes when you are using a window manager such as Suntools.

Access `telnet` by entering

**`telnet`** [*remote_host*]

where *remote_host* is the name of the host to which you wish to connect. (Check with your system manager or in the /etc/hosts file for names of machines on your local network.) For example, if you want to connect to a system called `aries`, enter

```
telnet aries
```

If you supply a remote host name on the command line, you receive the regular login banner for the remote machine, as shown in Figure 4-1.

Figure 4-1
Sample `telnet` Sequence

```
virgo% telnet aries
Trying...
Connected to aries.
Escape character is '^]'.
Cad System Machine (aries)
login: chico
Password: password
Last login: Mon Aug 24 13:32:15 from taurus
aries%
```

If you omit *remote_host*, `telnet` immediately issues the `telnet>` prompt.

## Invoking telnet

After you log in to the remote system, invoke `telnet` at any time by entering the escape character displayed on the screen when the connection is made. The default escape character is CTRL-]. Figure 4-2 shows `telnet` invoked through the use of the escape sequence.

Figure 4-2
Invoking `telnet` with the Escape Sequence

```
aries% CTRL-]
telnet>
```

With the `escape` option, you can change the escape sequence to any string you prefer. Use the following procedure:

1. Type ^] to get to the `telnet` command level. You should see the `telnet>` prompt.
2. Enter **`set escape`** after the prompt.
3. Enter the new escape sequence, then type RETURN to return to the command level on the remote machine. Figure 4-3 shows an example of this procedure.

Figure 4-3
Changing the `telnet` Escape Sequence

```
aries% CTRL-]
telnet> set escape @
escape character is '@'.
telnet>
```

Once you invoke `telnet`, you can display an online command summary or information about a particular command by entering the following at the `telnet>` prompt:

`telnet> ?` [*command*]

where *command* is the command for which you need instructions. If you do not supply a command name, a summary of `telnet` commands is displayed, as shown in Figure 4-4 on the following page.

Figure 4-4
`telnet` Command Summary

```
telnet> ?
Commands may be abbreviated. Commands are:

close       close current connection
display     display operating parameters
mode        try to enter line-by-line or character-at-a-time mode
open        connect to a site
quit        exit telnet
send        transmit special characters ('send ?' for more)
set         set operating parameters ('set ?' for more)
unset       unset operating parameters ('unset ?' for more)
status      print status information
toggle      toggle operating parameters ('toggle ?' for more)
slc         change state of special charaters ('slc ?' for more)
z           suspend telnet
!           invoke a subshell
?           print help information
telnet>
```

The `z` command suspends `telnet` and returns you to the command level on your local machine, as shown in Figure 4-5.

Figure 4-5
Suspending and Resuming `telnet`

```
aries% CTRL-]
telnet> z

Stopped
virgo% hostname
virgo
virgo% fg
telnet aries
RETURN
aries%
```

## Obtaining telnet Status

The `status` option reports the name of the machine to which you are currently connected, the current escape sequence, and other information about the connection. An example of `status` option output is shown in Figure 4-6.

Figure 4-6
Obtaining `telnet` Status

```
telnet> status
Connected to aries.
Operating with LINEMODE option
No line editing
Local catching of signals
Special characters are local values
Remote character echo
Local flow control
Escape character is '@'.
```

## Using telnet LINEMODE

Once `telnet` opens a connection, it attempts to enable LINEMODE. If the remote machine does not support LINEMODE, `telnet` reverts to either character-at-a-time mode or an older style line-by-line mode, depending on which mode the remote machine supports. You can use the `status` command to find out which mode `telnet` is using.

Using `telnet` in LINEMODE improves performance by reducing the number of network packets transmitted.

In LINEMODE, the remote system performs character processing normally done by the local system. The remote system relays information to the local system to enable input editing or character echoing. The remote system also relays changes to special characters that occur on the remote system and need to take effect on the local system.

For more information about using LINEMODE, refer to the `telnet`(1) man page in the *ConvexOS Programmer's Reference*.

## Ending a telnet Session

The `close` option is useful for opening a connection with another remote host, because it closes your current connection and returns you to `telnet` command level. Use it as shown in Figure 4-7.

Figure 4-7
Using the `close` and `open` Commands

```
telnet> close aries
Connection closed.
telnet> open groucho
Trying 130.50.75.1...
Connected to groucho.
Escape character is '^]'.
Lab Machine (groucho)
login: chico
Password: password
Last login: Mon Aug 24 13:32:15 from aries
groucho%
```

End a `telnet` session by entering `quit` or `q` after the `telnet>` prompt, as shown in Figure 4-8.

Figure 4-8
Ending a `telnet` Session

```
telnet> q
Connection closed.
aries%
```

## Using ftp

The *ftp* program provides the user interface to the File Transfer Protocol (FTP), which transfers files to and from a remote host. Access `ftp` by entering:

```
ftp [remote_host]
```

The `ftp>` prompt is displayed after you establish a connection and log in. The prompt indicates that `ftp` is awaiting your commands. Figure 4-9 shows a typical sequence of commands to establish a connection with `ftp`.

**Figure 4-9**
Connecting to a Remote System Using `ftp`

```
aries % ftp aries
Connected to aries.
220 aries FTP server (Version 4.163 Thu Jul 19 14:28:30 CDT 1990)
ready.
Name (aries:joebob): joebob
331 Password required for joebob.
Password:
230 User joebob logged in.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp>
```

To use `ftp`, you must have a password on your account on the remote system. If you do not have a password on the remote system, `ftp` displays an error message informing you that the login failed. Ask your system manager to set up your account with a password on the remote system.

If your account on the remote system uses a default shell that is not listed in the /etc/shells file, `ftp` displays an error message informing you that the login failed. To correct the problem, ask your system manager to add the shell to the /etc/shells file.

## Getting Help with ftp

To use the `ftp` online help facility, enter

**`help`** [*command*]

where command is the command with which you need help. If you omit the command argument, a list of `ftp` commands is displayed, as shown in Figure 4-10.

**Figure 4-10**
`ftp> help` Command

```
ftp> help
Commands may be abbreviated. Commands are:

!           delete      mdelete     proxy       runique
$           debug       mdir        sendport    send
account     dir         mget        put         size
append      disconnect  mkdir       pwd         status
ascii       form        mls         quit        struct
bell        get         mode        quote       system
binary      glob        modtime     recv        sunique
bye         hash        mput        remotehelp  tenex
case        help        nmap        rstatus     trace
cd          image       nlist       rhelp       type
cdup        lcd         ntrans      rename      user
close       ls          open        reset       verbose
cr          macdef      prompt      rmdir       ?
ftp>
```

The operating system on the remote host may not understand some of the commands listed in Figure 4-10. You can determine which commands are supported by the remote host by using the `remotehelp` command.

To obtain help information from the remote machine, enter

**`remotehelp`** [*command*]

You can also obtain the status of the remote machine by entering

**`remotestatus`**

## Obtaining ftp Status

The `ftp status` command reports the name of the machine to which you are connected and other information about the connection. An example of `status` command output is shown in Figure 4-6.

Figure 4-11
Obtaining `ftp` Status

```
ftp> status
Connected to aries.
No proxy connection.
Mode: stream; Type: binary; Form: non-print; Structure: file
Verbose: on; Bell: off; Prompting: on; Globbing: on
Store unique: off; Receive unique: off
Case: off; CR stripping: on
Ntrans: off
Nmap: off
Hash mark printing: off; Use of PORT cmds: on
ftp>
```

## Manipulating Files and Directories

In most cases, you invoke `ftp` either to retrieve a file from the remote machine or to transfer a file to one. You transfer files either in ASCII or in binary, by specifying the transfer type with the `type` command. If you transfer ASCII files to or from a system that does not run ConvexOS, you should always set the transfer `type` to `ascii`.

### Note

**Make sure you use `binary` mode to transfer binary files, such as `tar` files and executables. If you transfer files such as executables using `ascii` mode, the files will not execute on the remote machine. You can determine a file's type by using the `file` command. (Refer to the `file`(1) man page in the *ConvexOS Programmer's Reference*.)**

Use the `get` command to retrieve a file from another system, and the `put` command to transfer a file to another system. For example, to retrieve the file tuttifrutti, issue the following command:

```
get tuttifrutti lrichard
```

The file is retrieved from the remote machine and written to the local file, lrichard. (Remember that you have already invoked `ftp` and opened a connection to the remote machine. For this reason, you need not specify any host names as you did with `rcp`.)

Similarly, to write tuttifrutti to a file named richard_remote, use the `put` command as follows:

```
put tuttifrutti richard_remote
```

The `mput` and `mget` commands are used for writing or retrieving multiple files. To write a series of files (for example, larry, curly, and moe), use `mput` as follows:

```
mput larry curly moe
```

Files are written to your current working directory on the remote machine. If you prefer to write the files to another directory, remember to use `cd` before you issue the `mput` command.

Before any files are transmitted, you are given an opportunity to verify the file list, as shown in Figure 4-12.

Figure 4-12
Sample mput Sequence

```
ftp> mput larry curly moe
mput larry? y
     .
     .
     .
mput curly? y
     .
     .
     .
mput moe? y
     .
     .
     .
ftp>
```

mput is most widely used with wildcards ("*") to transfer whole classes or directories of files at one time. For example, the command

```
mput *.f
```

transfers all files in the current directory that end in ".f".

## Note

**If you use the prompt command to enable prompting, ftp prompts you during multiple file transfers to allow you to selectively retrieve or store files. By default, prompting is enabled.**

**mget works similarly to mput. To retrieve a set of files (for example, aries, groucho, and chico) from the remote machine, enter**

```
mget harpo groucho chico
```

As with mput, if you have enabled prompting, you are prompted for an affirmative response before any files are transmitted. You may also use wildcards with this command as with mput. Files are written to your current local working directory.

If you prefer to append your file to a file on the remote machine, use append. For example, to append the local file hounddog to the remote file richard_remote, enter

```
append hounddog richard_remote
```

Delete richard_remote by entering:

```
delete richard_remote
```

As you might expect, you use mdelete to delete multiple files.

To delete a directory, enter

**rmdir** *directory_name*

To create a directory, enter

**mkdir** *directory_name*

To change directories on the remote machine, enter

**cd** *directory_name*

Use the `lcd` command to change directories on the local machine.

**lcd** *remote_directory*

To list contents of a remote directory, enter

**ls**

You receive output similar to Figure 4-13.

Figure 4-13
Sample `ls` Output

```
ftp> ls
200 PORT command okay.
150 Opening data connection for /bin/ls (192.18.44.1,50465) (0\
bytes).
.cshrc
.login
.logout
.mailrc
bin
mcsdirs
os.pages
util.pages
226 Transfer complete.
154 bytes received in 0.29 seconds (0.52 Kbytes/s)
```

Output is routed to your terminal. If you prefer to route output to a local file, specify the name of the file as the second argument. As an alternative to `ls`, `dir` displays file-system structure, the length of each file, and the date each was created. To use `dir`, enter

**dir**

Your output should resemble that in Figure 4-14 on the following page.

**Figure 4-14**
Sample `dir` Output

```
ftp> dir
200 PORT command okay.
150 Opening data connection for /bin/ls (192.18.44.1,54414) (0
bytes).
total 39
-rw------- 1   chico   docn      317  Jun 22 10:48 .cshrc
-rwxr-xr-x 1   chico   docn     1470  Oct 1 1984   .emacs_pro
-rw-r--r-- 1   chico   docn      410  Apr 22 15:13 .login
-rwxr--r-- 1   chico   docn       31  Jan 16 1986  .logout
-rw-r--r-- 1   chico   docn       36  Feb 25 1985  .mailrc
drwxrwxr-x 2   chico   docn      512  Feb 7 1986   bin
-rw-r--r-- 1   chico   docn     1396  Jun 23 15:56 mac.sup
drwxrwxr-x 3   chico   docn      512  Jul 1 16:13  mcsdirs
-rw-rw-r-- 1   chico   docn     1476  Jul 28 14:37 os.pages
-rw-rw-r-- 1   chico   docn    19560  Jul 28 14:43 util.pages
226 Transfer complete.
1028 bytes received in 0.51 seconds (2 Kbytes/s)
ftp> bye
221 Goodbye.
%
```

If all you are interested in is a file's size or the time it was last modified, you can obtain just that information. To display the size of a file, enter

**`size`** *file*

Similarly, to determine when a file was last modified, enter

**`modtime`** *file*

## Ending an ftp Session

End your `ftp` session with either `close`, `quit`, or `bye`. Entering `close` after the `ftp>` prompt terminates the session and returns you to the `ftp` command interpreter, as shown in Figure 4-15.

Figure 4-15
Terminating `ftp` with the `close` Command

```
virgo% ftp aries
Connected to aries
220 aries FTP server (Version 4.100 Wed Jul 22 23:32:21 CDT 1987)
ready.
Name (aries:): chico
Password (aries:chico): password
331 Password required for chico.
230 User chico logged in.
ftp> close
221 Goodbye.
ftp>
```

Entering `quit` or `bye` ends the `ftp` session and returns you to the shell, as shown in Figure 4-16.

Figure 4-16
Ending an `ftp` Session with `bye`

```
ftp> bye
aries% hostname
aries
aries%
```

You can also use the EOF command (**CTRL-D**) to end an `ftp` session and return to the shell.

## Using tftp

The `tftp` program provides the user interface to the Internet Trivial File Transfer Protocol (TFTP). Like `ftp`, `tftp` allows users to transfer files to and from remote hosts. It is normally used to transfer files, such as boot files, from the CONVEX system to remote workstations. Unlike `ftp`, `tftp` does not require a user account or password on the remote host (refer to "Access Restrictions," below).

Run `tftp` by entering:

```
tftp [remote_host]
```

The prompt, `tftp>`, indicates that `tftp` is awaiting your commands.

Unlike `ftp`, `tftp` does not maintain connections between transfers. If you specify *remote_host* on the command line, `tftp` uses that host as the default for future transfers. Likewise, the `connect` command does not actually establish a connection, but merely remembers which host to use for transfers.

**Note**

**You do not have to use the `connect` command; you can specify the remote host as part of the `get` or `put` commands.**

### Access Restrictions

Because the TFTP protocol performs no user validation, the remote host will probably have some sort of file access restrictions in place. On CONVEX machines, if the `tftp` daemon is started with the `-s` option, access to files via `tftp` is restricted to the directory specified when the `tftp` daemon is started (*homedir*), or to the default directory, /tftpboot.

In addition, `tftp` restricts read access to publicly-readable files. You may only write files that already exist and are publicly writable.

For more information, refer to the `tftp`(1C) and `tftpd`(8C) man pages and the *CONVEX Internet Services System Manager's Guide.*

### Getting Help with tftp

To use the `tftp` online help facility, enter

```
? [command]
```

where *command* is the command with which you need help. If you omit *command*, a list of `tftp` commands is displayed, as shown in Figure 4-17 on the following page.

Figure 4-17
`tftp` Help Information

```
tftp> ?
Commands may be abbreviated. Commands are:

connect         connect to remote tftp
mode            set file transfer mode
put             send file
get             receive file
quit            exit tftp
verbose         toggle verbose mode
trace           toggle packet tracing
status          show current status
binary          set mode to octet
ascii           set mode to netascii
rexmt           set per-packet retransmission timeout
timeout         set total retransmission timeout
?               print help information
tftp>
```

## Obtaining tftp Status

The `tftp status` command reports the name of the machine to which you are connected and other information about the connection. An example of `status` command output is shown in Figure 4-18.

Figure 4-18
Obtaining `tftp` Status

```
tftp> status
Connected to aries.
Mode: netascii Verbose: off Tracing: off
Rexmt-interval: 2 seconds, Max-timeout: 10 seconds
tftp>
```

## Transferring Files and Directories with tftp

You invoke `tftp` to retrieve a file from the remote machine or transfer a file to one. Files can be transferred in either ASCII or in binary by specifying the transfer type with the `type` command. If you transfer ASCII files to or from a system that does not run ConvexOS, you should always use the default transfer type, `ascii`.

**Be sure to use `binary` mode to transfer binary files, such as `tar` files and executables. If you transfer files such as executables using `ascii` mode, the files will not execute on the remote machine. You can determine a file's type by using the `file` command. (Refer to the `file(1)` man page in the *ConvexOS Programmer's Reference*.)**

### Using the tftp get Command

Use the `get` command to retrieve files or directories from a remote host. Enter the `get` command using one of the following formats:

**`get`** *remote_name*
**`get`** *remote_name local_name*
**`get`** *file1 file2 file3 ... fileN*

You specify remote file names in one of two forms: as a file name on the remote host, if the host has already been specified on the command line or with a `connect` command; or as `hostname:filename` if you have not specified a remote host.

For example, to retrieve the file boot.sun from a remote host, issue one of the following commands:

```
get boot.sun
get boot.sun bootfile
get aries:boot.sun
get aries:boot.sun bootfile
```

You must use a format that includes the remote host name if you have not previously specified the name of the remote host on the command line or with a `connect` command. If you try to use the get command without first specifying a remote host name, `tftp` displays the message shown in Figure 4-19.

Figure 4-19
Failing to Specify a Remote Host Before Executing a `tftp get` Command

```
virgo% tftp
tftp> put bootfile
No target machine specified.
tftp>
```

To retrieve multiple files when you have already specified a remote host, enter a command similar to

```
get boot.sun3 tpboot.sun in.tftpd
```

Figure 4-20 shows a successful file transfer using the `tftp get` command.

Figure 4-20
Retrieving a File with the `tftp get` Command

```
virgo% tftp aries
tftp> get bootfile
Received 1218 bytes in 0.5 seconds
tftp>
```

**Note**

**If the remote host is a CONVEX system and the `-s` option is set when the `tftp` daemon is configured, you may retrieve only publicly-readable files located in the /tftpboot directory tree or in the directory tree specified when the remote `tftp` daemon is started.**

### Using the tftp put Command

Use the `put` command to transfer files to a remote system. Enter the `put` command using one of the following formats:

**put** *local_name*
**put** *local_name remote_name*
**put** *file1 file2 file3 ... fileN* [*remote_directory*]

For example,

```
put bootfile
put bootfile boot.sun
put aries:bootfile
put aries:bootfile boot.sun
put aries:file1 file2 file3 /tftpboot
```

As with the `get` command, you must specify file names as `hostname:filename` unless you have already specified the remote host on the `tftp` command line or with the `connect` command. If you try to use the `put` command without having first specified a remote host name, `tftp` displays an error message, as shown in Figure 4-21.

Figure 4-21
Failing to Specify a Remote Host Before Executing a `tftp put` Command

```
virgo% tftp
tftp> get bootfile
usage: get host:file host:file ... file, or
       get file file ... file if connected
tftp>
```

Figure 4-22 shows a successful file transfer using the `tftp put` command.

Figure 4-22
Transferring a File with the `tftp put` Command

```
virgo% tftp
tftp> put bootfile
Send 1218 bytes in 0.1 seconds
tftp>
```

**Note**

**If the remote host is a CONVEX system and the `tftp` daemon is configured with the -s option set, you may transfer only publicly-writable files that already exist in the /tftpboot directory tree or in the directory tree specified when the remote `tftp` daemon is started.**

## Ending a tftp Session

End your `tftp` session with the `quit` command, as shown in Figure 4-23.

Figure 4-23
Ending a `tftp` Session with the `quit` Command

```
tftp> quit
aries%
```

# Command Summary

The following table lists the syntax of each of the utilities discussed in this guide. For more information on these commands, refer to the corresponding man pages in the *ConvexOS Programmer's Reference.*

## Table A-1: Command Summary

| Command Syntax | Function |
|---|---|
| **ftp** *remote_host* | Establish connection with specified host |
| ftp>! [*command* [*args*] ] | Invoke interactive shell on local machine |
| ftp>$ *macro_name* [*args*] | Execute the macro that was defined by the **macdef** command |
| ftp>? | A synonym for **help** |
| ftp>**account** [*passwd*] | Supply a supplemental password required by remote machine |
| ftp>**append** *local_file* [*remote_file*] | Append a local file to a remote file |
| ftp>**ascii** | Set file transfer type to ascii |
| ftp>**bell** | Sound a bell after each file transfer |
| ftp>**binary** | Set file transfer type to binary |
| ftp>**bye** | Exit ftp; return to shell |
| ftp>**case** | Toggle remote computer file name case mapping during mget commands |
| ftp>**cd** *directory_name* | Change working directory on remote host |
| ftp>**cdup** | Change the remote working directory to the parent of the current remote machine working directory |
| ftp>**close** | Exit ftp; return to ftp command interpreter |
| ftp>**cr** | Toggle carriage-return stripping during ascii file retrieval |
| ftp>**debug** [debug-value] | Toggle debugging mode; set debug level |

## Table A-1: Command Summary (continued)

| Command Syntax | Function |
|---|---|
| `ftp>delete` *remote_file* | Delete a remote file |
| `ftp>dir` [*remote_directory*] [*local_file*] | List contents of a remote directory |
| `ftp>disconnect` | A synonym for `close` |
| `ftp>form` *format* | Set file transfer form to *format* |
| `ftp>get` *remote_file* [*local_file*] | Write a remote file to a local destination |
| `ftp>glob` | Toggle filename expansion for `mdelete`, `mget`, and `mput` |
| `ftp>hash` | Toggle hash-sign printing for each data block transferred |
| `ftp>help` [*command*] | Display help information |
| `ftp>lcd` [*directory_name*] | Change working directory on local machine |
| `ftp>ls` [*remote_directory*] [*local_file*] | List contents of a remote directory |
| `ftp>macdef` *macro_name* | Define a macro |
| `ftp>mdelete` *remote_files* | Delete multiple remote files |
| `ftp>mdir` *remote_files local_file* | List contents of multiple remote directories |
| `ftp>mget` *remote_files* | Write multiple remote files to current local working directory |
| `ftp>mkdir` *directory_name* | Create a directory on the remote machine |
| `ftp>mls` *remote_files local_file* | Print a list of the files in multiple remote directories |
| `ftp>mode` [*mode_name*] | Set the file transfer mode to *mode_name* |
| `ftp>modtime` *file_name* | Show the last modification time of the remote file |
| `ftp>mput` *local_files* | Write multiple local files to current remote working directory |
| `ftp>nlist` [*remote_directory*] [*local_file*] | Print contents of the remote directory |
| `ftp>nmap` [*inpattern outpattern*] | Set or unset the filename mapping mechanism |
| `ftp>ntrans` [*inchars* [*outchars*] ] | Set or unset the filename character translation mechanism |
| `ftp>open` *host [port]* | Establish a connection to the specified host FTP server |
| `ftp>prompt` | Toggle interactive prompting |
| `ftp>proxy` *ftp_command* | Execute an `ftp` command on a secondary control connection |

## Table A-1: Command Summary (continued)

| Command Syntax | Function |
|---|---|
| ftp>**put** *local_file* [*remote_file*] | Write a local file to a remote destination |
| ftp>**pwd** | Print the name of the current remote working directory |
| ftp>**quit** | A synonym for bye |
| ftp>**quote** *arg1 arg2* . . . | Send the arguments, verbatim, to remote FTP server. |
| ftp>**recv** *remote_file* [*local_file*] | A synonym for get |
| ftp>**remotehelp** [*command_name*] | Request help from the remote server |
| ftp>**rstatus** [*file_name*] | Show status of remote machine or remote file |
| ftp>**rename** [*from*] [*to*] | Rename file on remote machine |
| ftp>**reset** | Clear reply queue |
| ftp>**rmdir** *directory_name* | Delete a remote directory |
| ftp>**runique** | Toggle storing of files on local system with unique file names |
| ftp>**send** *local_file* [*remote_file*] | A synonym for put |
| ftp>**sendport** | Toggle use of PORT commands |
| ftp>**size** *file_name* | Return size of *file_name* on remote machine |
| ftp>**status** | Show the current status of ftp |
| ftp>**struct** [*struct_name*] | Set file transfer structure to *struct_name* |
| ftp>**sunique** | Toggle storing of remote files under unique file names |
| ftp>**system** | Show type of operating system running on remote machine |
| ftp>**tenex** | Set file transfer type to that needed for TENEX machines |
| ftp>**trace** | Toggle packet tracing |
| ftp>**type** [*type_name*] | Set file transfer type |
| ftp>**user** *user_name* [*password*] [*acct*] | Identify yourself to the remote FTP server |
| ftp>**verbose** | Toggle verbose mode |
| **hostname** | Show name of current host |
| **rlogin** *remote_host* | Log in to the remote host |
| **rcp** *remote_host:filename local_file* | Copy a remote file to a local destination |

## Table A-1: Command Summary (continued)

| Command Syntax | Function |
|---|---|
| **rcp** *local_file remote_host:filename* | Copy a local file to a remote destination |
| **rcp** *user@host:filename local_file* | Copy a remote file under a different user name to a local destination |
| **rsh** *remote_host command_name* | Execute specified command on a remote host |
| **ruptime** [*machine*] | Check uptime status of a remote machine |
| **rwho** [*username*] | Show who is logged in to a remote machine |
| **telnet** *remote_host* | Establish connection with specified host |
| telnet>**!** [*command*] | Execute a single command on the local system |
| telnet>**?** | Display "help" information |
| telnet> **close** | Close connection with specified host |
| telnet>**display** [*argument...*] | Displays all, or some, of the `set` and `toggle` values |
| telnet>**erase** *character* | Change erase character from default to *character* |
| telnet> **escape** *character* | Change `telnet` escape sequence |
| telnet>**flushoutput** *character* | Change character used to flush output buffer |
| telnet>**interrupt** *character* | Change telnet interrupt character |
| telnet>**kill** *character* | Change `telnet kill` character |
| telnet>**lnext** *character* | Change `telnet lnext` character |
| telnet>**mode** *type* | Select `line` or `character` mode |
| telnet> **open** *remote_host* [*port*] | Open a connection to the named host |
| telnet> **quit** | Close any open `telnet` session and exit |
| telnet>**reprint** *character* | Change `telnet reprint` character |
| telnet>**send** *arguments* | Send one or more special character sequences to remote host |
| telnet>**set** *argument value* | Set any one of a number of `telnet` variables to a specific value |
| telnet>**slc** *state* | Set or change the state of special characters when in LINEMODE |
| telnet>**start** *character* | Change `telnet start` character |
| telnet> **status** | Report current escape sequence and host |

## Table A-1: Command Summary (continued)

| Command Syntax | Function |
|---|---|
| telnet>**stop** *character* | Change `telnet stop` character |
| telnet>**susp** *character* | Change `telnet` suspend character |
| telnet>**toggle** *arguments* | Toggle various control flags |
| telnet>**tracefile** *filename* | File to contain output from `netdata` or `option` tracing |
| telnet>**unset** *arguments* | Set `telnet` variables to FALSE |
| telnet>**worderase** *character* | Change character used to erase words |
| telnet> **z** | Suspend `telnet` (using `csh` only) |

# Reporting Problems

B

This appendix introduces the CONVEX Technical Assistance Center (TAC) and the `contact` utility.

The `contact` utility is an online system for reporting problems to the TAC. To use it, enter **`contact`** at the system prompt and answer the questions as they appear on the screen.

This appendix describes:

- Prerequisites for using `contact`.
- Tips for using `contact`.
- The step-by-step process `contact` takes you through.

## Technical Assistance Center

The CONVEX Technical Assistance Center (TAC) is staffed by technical specialists who can address diverse questions and problems that arise in a supercomputing environment. If you have a hardware, software, or documentation question, contact the TAC. This group stands ready to solve such problems.

## The contact Utility

The TAC recommends using the `contact` utility to report a hardware, software, or documentation problem. The `contact` utility is an interactive program that helps the TAC track reports and route them to the CONVEX personnel most qualified to fix a problem.

After you invoke `contact`, it prompts you for information about the problem. When you finish your report, `contact` electronically mails it to the TAC. The TAC notifies you within 48 hours that your report has been received.

Use of the `contact` utility requires:

- UNIX-to-UNIX Communication Protocol (UUCP) connection to the TAC.
- Full path name of the program or utility in question.
- Version number of the program or utility in question.

## UUCP Connection

Before using `contact`, ask your system administrator if your site has a UUCP connection to the TAC. A UUCP connection allows files to be copied from one UNIX-based system to another. The `uucp` (UNIX-to-UNIX copy) command relies on either a dial-up or hard-wired UUCP communication line.

## Finding the Program Path Name

To determine the full path name of the program or utility in question, use the `which` command. Figure B-1 illustrates use of the `which` command to find the full path name of the loader utility.

Figure B-1
Using the `which` Command

```
> which ld
/bin/ld
>
```

In this example, the full path name of the loader is `/bin/ld`.

If you use the C shell (`csh`), you can also use the `whence` command to find the program path name. The `whence` command works like `which`, but faster.

For more information on the `which` command, refer to the `which`(1) man page. You can also use the `info` online information system by entering **`info which`** at the system prompt.

## Finding the Program Version Number

To determine the version number of the program or utility in question, use the `vers` command. Figure B-7 illustrates use of the `vers` command to find the version number of the loader (`ld`) utility. Enter **`vers`**, then the path name of the program or utility.

Figure B-2
Using the `vers` Command

```
> vers /bin/ld
/bin/ld: 7.0
>
```

In this example, the loader version number is 7.0.

For more information on the `vers` command, refer to the `vers`(1) man page. You can also use the `info` online information system by entering **`info vers`** at the system prompt.

## Using contact

The `contact` utility prompts for the following information:

- Your name, title, phone number, and corporate name.
- Name and version of the product.
- One-line summary of the problem.
- Detailed description of the problem.
- Priority of the problem.
- Instructions on how to reproduce the problem.
- Comments about the problem.
- Comments about the documentation relating to the problem.
- Files to include in the `contact` report.

Following is a step-by-step discussion of these prompts.

### Step 1a

To invoke the `contact` utility, enter **`contact`** at the system prompt. The system responds with a welcome message and a series of questions regarding your hardware, software, or documentation question. Figure B-3 illustrates use of the `contact` command.

**Figure B-3**
Beginning a `contact` Session

```
> contact
Welcome to contact version 0.11 ()

Enter your name, title, phone number, and corporate name (^D to terminate)
>
```

### Step 1b

If there is a .contact file in your home directory, `contact` skips the first prompt. (Refer to "Using a .contact File" for more information.) Figure B-4 illustrates the `contact` command and the system response when you have a .contact file in your home directory.

**Figure B-4**
Beginning a `contact` Session with a .contact File

```
> contact
Welcome to contact version 0.11 ()

Enter the name of the product involved
>
```

## Step 2

The `contact` utility prompts for the version number of the product. If you do not know the version number, press **CTRL-Z** to suspend the session.

Use the `which` (or `whence` if you use `csh`) and `vers` commands to find the version number of the product. Use the `fg` command to return to the session, and enter the version number in the form X.X or X.X.X.X.

## Step 3

The `contact` utility prompts for a one-line summary of the problem. This summary is the subject header in any further correspondence regarding the problem. Please make this summary as descriptive as possible in one line.

## Step 4

The `contact` utility prompts for a detailed description of the problem. Please make this description as complete as possible. Include source code and a stack backtrace when possible. (Refer to the `adb`(1) or `csd`(1) man page for information on obtaining a stack backtrace.) The more information you provide, the quicker the TAC can isolate and solve the problem.

## Step 5

The `contact` utility prompts for the priority of the problem. Figure B-5 illustrates this prompt and priority levels from which to choose.

Figure B-5
Specifying Priority
of a Problem

```
Enter a problem priority, based on the following:
1) Critical     - work cannot proceed until the problem is resolved.
2) Serious      - work can proceed around the problem, with difficulty.
3) Necessary    - problem has to be fixed.
4) Annoying     - problem is bothersome.
5) Enhancement  - requested enhancement.
6) Informative  - for informational purposes only.
>
```

## Step 6

The `contact` utility prompts for an explanation of how to reproduce the problem. Please include the command syntax, the options you used, and anything else you did to run the program.

## Step 7

The `contact` utility prompts for any other pertinent comments. Please include all relevant information.

## Step 8

The `contact` utility prompts for suggestions regarding documentation supporting the product. Indicate whether the documentation could be revised to address the problem.

## Step 9

The `contact` utility prompts for names of files necessary to reproduce the problem. Figure B-6 illustrates this prompt and sample user response.

Figure B-6
Including Files in a `contact` Report

```
Are there any files that should be included in this report (yes | no)?
> yes
Please enter the names of the files, one to a line (^D to terminate)
> test.f
> ~/subroutines/sub.f
>
```

**Note**

**Tilde-escape sequences are not recognized in responses to this prompt. In `contact`, a tilde in this section indicates your home directory. This convention is based on use of the tilde for expanding file names in `csh`.**

If you specify small text files, they are automatically included in the `contact` report. If the files are too large to be included in this report, `contact` gives further instructions on how to submit these files.

To specify a directory, combine directory files into a single file using the `tar` command (refer to the `tar`(1) man page for further information) or enter each file name in the directory on a single line in the `contact` report.

## Step 10

The `contact` utility prompts you to review, edit, submit, or abort the report. Figure B-7 illustrates this prompt.

Figure B-7
Prompt to Review, Edit, Submit, or Abort Report

```
Please select one of the following options:
1) Review the problem report.
2) Edit the problem report.
3) Submit the problem report.
4) Abort the problem report.
>
```

Choose the number of the option you want to select. These options let you do the following:

**Review** Review the text of the `contact` report. You are then prompted again to select an option.

**Edit** Edit the text of the `contact` report. If you choose to edit the report, `contact` opens your default text editor.

**Submit** Send the report to the CONVEX TAC. The TAC notifies you within 48 hours that your report has been received. Choosing this option exits the `contact` utility and returns you to the shell.

**Abort** Save the text of the report in a file named ~/dead.report. Choosing this option exits `contact` and returns you to the shell.

## Tips for Using contact

The `contact` utility is interactive and easy to use. This section lists tips to help you use it efficiently. In particular, this section explains how to:

- Use a `.contact` file.
- Abort a `contact` session.
- Resubmit an aborted report.
- Suspend a `contact` session.
- Move within `contact` from one prompt to another.
- Use tilde-escape sequences in the `contact` utility.

### Using a .contact File

When you invoke `contact`, it first prompts for your name, title, phone number, and company name. You can, however, create a .contact file to skip this first prompt.

Follow these steps to create a .contact file:

1. Create a .contact file in your home directory.
2. Enter your name, job title, phone number, and company name, each on a new line.

When you invoke `contact`, it automatically includes the .contact file as input for the first prompt and proceeds to the next prompt.

### Aborting the Report

To abort a `contact` report, either press the interrupt key (usually **CTRL-C**) or choose the abort option when prompted by the `contact` utility. Using **CTRL-C** to abort does not save the contents of the report. Using the `abort` option saves the contents of the report in a file named ~/dead.report.

### Submitting the dead.report File

After you abort a `contact` session, the `contact` utility saves the report in a file named ~/dead.report. Using the `contact` command with the `-r` option automatically merges the contents of the ~/dead.report file into the new `contact` session. Enter

```
contact -r
```

and `contact` finds the ~/dead.report file and merges it into the `contact` report. You can then edit the report. When you end the editing session, `contact` resumes at the final prompt, which asks you to review, edit, submit, or abort the report.

## Suspending a Report

Sometimes it is necessary to stop in the middle of a `contact` report and return to the shell (for instance, to suspend the `contact` session to find the program path name or version number). To suspend the `contact` session, press **CTRL-Z**.

To return to the `contact` session, type `fg`. Using **CTRL-Z** and the `fg` (foreground) command, you can switch between the `contact` utility and the shell. You cannot, however, use **CTRL-Z** and `fg` to switch back and forth in the Bourne shell (`sh`).

## Ending a Response

The `contact` utility prompts for information pertinent to your hardware, software, or documentation question. Some prompts require one-line responses; to move to the next prompt, press **RETURN**. Other prompts require more than a one-line response; to move to the next prompt, press **CTRL-D**.

## Tilde-Escape Sequences

The `contact` utility treats input beginning with a tilde (~) as a special sequence. The character following the tilde is considered a request for a special function. You can use the following tilde sequences within `contact`:

| | |
|---|---|
| **~e** | Start the text editor (defined in the `EDITOR` environment variable). |
| **~h** | Display a list of available tilde-escape sequences. |
| **~p** | Print the `contact` report to the terminal screen. |
| **~r** *filename* | Read the contents of *filename* as a response to the current prompt. Some prompts require only a one-line response. This tilde-escape sequence works only for prompts that allow more than a one-line response. |
| **~~** | Insert a single tilde as the first character in the line. |

# Index

## G



## H



## I



## L



## M



## N



## O



## P

## R



## S



## T



## U

## V



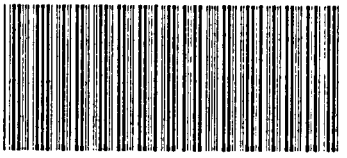
## W



## Y

# DSW-141

CONVEX COMPUTER CORPORATION



COVNEX INTERNET SERVICES USER'S GUIDE
710-002530-204

PRINTED IN U.S.A.