# C-Kermit User's Guide

# for FlexOS-286™

Adapted from
Fifth Edition, Revision 1

Frank da Cruz, editor

Columbia University Center for Computing Activities
New York, New York 10027

1073-2353-001

To obtain more information on the KERMIT protocol or KERMIT itself, contact:

KERMIT Distribution
Columbia University Center for Computing Activities
612 West 115th Street
New York NY 10025

The KERMIT protocol was named after Kermit the Frog, star of the television series The Muppet Show, and is used by permission of Henson Associates, Inc.

This manual is written for the C-Kermit user and requires familiarity with FlexOS commands and command entry. See the FlexOS User's Guide for the description of FlexOS operation.

# Contents

# C-Kermit Overview

## 1.1 KERMIT File Transfer Protocol

KERMIT is a protocol for transferring files between computers of all sizes over asynchronous serial lines. The protocol defines a packet format and uses checksums and packet-retransmission to ensure data integrity. KERMIT is non-proprietary, thoroughly documented, well tested, and in wide use. The protocol and the original implementations were developed at Columbia University and have been shared with many other institutions, some of which have made contributions of their own.

## 1.2 C-Kermit Implementation

The C-Kermit implementation of the KERMIT protocol is for computers running FlexOS™. It requires the following of the system:

- The computer system can receive 7 or 8 bit ASCII encoded characters over a communications line.
- The host accepts all printable ASCII characters without transforming them in any way.
- A single control character can be passed from one system to another without being transformed.
- If the host requires a line terminator, the terminator can be a single control character different from the packet start-of-header character (normally a Ctrl-A)
- The host's terminal is capable of receiving a burst of 40 to 100 characters at normal transmission speeds. (The normal packet is 40 to 100 characters long.)

The KERMIT protocol does not require the following:

- A specific baud rate
- XON/XOFF flow control
- Full duplex capability
- The ability to send or receive 8-bit bytes

## 1.3 C-Kermit Commands

C-Kermit supports two command-entry modes: interactive and command-line. Table 1-1 describes the frequently-used interactive mode commands. Figure 1-1 lists all of the interactive and command-line mode commands. Complete command descriptions are provided in Section 2.

**Note:** Throughout this manual, local refers to the system that you are using directly, remote refers to the system to which you are connected by C-Kermit.

## 1.4 C-Kermit Server

The C-Kermit server is a Kermit mode useful for transferring multiple files with different names. You invoke server mode with the C-Kermit SERVER command. The command can be entered either directly or remotely from a computer functioning as a dumb terminal. While a computer is in server mode, it is unavailable for local command execution unless C-Kermit is running in the background (see the BACK command description in the FlexOS User's Guide).

You communicate with a remote C-Kermit server from the local system using the GET, SEND, and REMOTE commands. Use GET to get a file or files from the server; use SEND to send a file or files. REMOTE lets you issue a limited set of C-Kermit file management commands to the remote computer. For example, the command

```
   C-Kermit>REMOTE DIR
```

displays the contents of the C-Kermit server's current directory. See the REMOTE command description in Section 2 for the list of C-Kermit commands available on remote servers.

You terminate remote server mode from the local system with the FINISH and BYE commands. If you started server mode on your local system, you can only terminate it if you invoked C-Kermit to run in the background. Use the FlexOS PROCESS command to determine the C-Kermit process ID and to terminate the process.

## 1.5 C-Kermit Initialization

Upon entering interactive mode, C-Kermit looks in your current directory for and executes the commands in an initialization file called CKERMIT.INI. These commands are typically a series of C-Kermit SET commands required to modify the default protocol parameters so that C-Kermit can run.

You construct the CKERMIT.INI file in the same way you build files for the TAKE command. See the TAKE description in Section 2 for instructions.

**Table 1-1. Frequently-used C-Kermit Commands**

| Command | Description |
|---|---|
| CONNECT | Make a "virtual terminal" connection to the remote system. After the CONNECT command completes, keyboard input is output through the current serial port and serial port input is displayed on the console screen. |
| SEND | Send a file or files to another computer running KERMIT. You send a group of files using the * and ? wildcard characters. SEND can be invoked locally (that is, you send files to another computer) or remotely (that is, you send the files from the remote computer to your computer). |
| RECEIVE | Receive a file or files from another computer running KERMIT. Files are received and recorded until the SEND command invoked at the remote computer completes. |
| SET | Set communication line parameters. C-Kermit has default values for all parameters. Use the SET command to change the defaults. You can also set parameters with a CKERMIT.INI file. |
| SHOW | Display the values of SET options. |
| HELP | Display a summary of C-Kermit commands. |
| BYE | Terminate the remote Kermit server and log off the remote host. |
| EXIT | Terminate C-Kermit on the local system. |
| ? | List the commands, options, or operands that are possible at this point. |

| Local Commands | SET Options | Remote Commands |
|---|---|---|
| ! | block-check | chdir |
| bye | delay | delete |
| close | duplex | directory |
| connect (-c, -n) | end-of-packet | help |
| chdir | escape-character | host |
| delete | file (-i, -w, -q) | space |
| dial | flow-control | type |
| directory | handshake (-t) | who |
| echo | line (-l) | |
| exit | modem-dialer | |
| finish (-f) | packet-length | |
| get (-g) | pad-character | |
| help (-h) | padding | |
| log (-d) | parity (-p) | |
| quit | prompt | |
| receive (-r, -k, -a) | speed (-b) | |
| remote | start-of-packet | |
| script | timeout | |
| send (-s, -a) | | |
| server (-x) | | |
| set | | |
| show | | |
| space | | |
| statistics | | |
| take | | |
| type | | |

Command-line functions are shown in parenthesis.

**Figure 1-1. C-Kermit Commands**

End of Section 1

## C-Kermit Commands

C-Kermit supports two command modes:

- **Command-line:** In this mode, you specify the function to be performed in the C-Kermit command line. For example, the following command selects the serial port, sets the baud rate, establishes a connection, and receives a file:

  ```
  C-KERMIT -l ser: -b 1200 -c -r
  ```

  The C-Kermit functions in this example are -l, -b, -c, and -r. These and the other functions are described in Section 2.3 "Command-line Mode Functions" below. The command-line mode functions are a subset of the functions available in interactive mode.

- **Interactive:** In this mode, you enter commands from the C-Kermit prompt. For example, the following command sends the file GL.DAT to a remote server:

  ```
  C-Kermit> send gl.dat
  ```

  You enter interactive mode in one of two ways:

  - Type C-KERMIT alone at the shell prompt. See Section 2.4 for the interactive mode command descriptions.
  - Type C-KERMIT and a command-line function that sets a transfer characteristic but specifies no action. Section 2.3 distinguishes the commands that set transfer characteristics from those that specify actions.

Both modes allow command-line editing. In command-line mode, you use the shell's line-editing characters to edit the command line. In interactive mode, C-Kermit provides its own set of editing characters. These and the other C-Kermit special characters are described next.

## 2.1 Special Characters

Special C-Kermit characters are used in the following situations:

- to manage the connection when you are in dumb terminal
- to interrupt file transfers
- to edit the command line in interactive mode.

The following table lists the escape functions available when you are in dumb terminal. All of them are entered by typing Ctrl-\ (referred to as the escape character) followed by the character listed in the table.

| Character* | Description |
|---|---|
| C | Close the connection |
| S | Show connection status |
| ? | List escape options |
| B | Send a BREAK signal |
| 0 | Send a null |
| Ctrl-\ | Send the escape |

* Precede the character with Ctrl-\ to perform the function.

You use control characters to interrupt file transfers. The following lists the characters available and the type of interrupt performed:

| | |
|---|---|
| Ctrl-F | Stops the current file transfer and starts the next file if there are more files to transfer. |
| Ctrl-B | Stops the current file transfer and terminates the entire transfer operation if more files remain. |
| Ctrl-R | Resends the current packet. |
| Ctrl-A | Displays a status report for the current file transfer. |

Table 2-1 lists the interactive mode line editing characters. When you are in C-Kermit command-line mode, use the shell's line editing characters.

**Table 2-1. Interactive Mode Editing Characters**

| Character | Description |
|---|---|
| ? | Invokes the help facility. The response indicates what is expected to complete the command sequence. The example that follows illustrates the C-Kermit help facility. |
| Esc | Completes the current C-Kermit command or substitutes a default value. If your entry has no corresponding C-Kermit command, C-Kermit outputs a beep. The example that follows this table illustrates Esc key use. |
| Del | Deletes the character immediately preceding the cursor. Backspace and Control-H do the same thing |
| Ctrl-W | Erases one word left of the cursor. |
| Ctrl-U | Erases the entire line entry. |
| Ctrl-R | Redisplays the current line entry. |
| Space | Delimits a field such as a command, filename, and number. |
| Ctrl-I | Delimits fields in the same way as a space. |
| Carriage Return | Terminates a command sequence and causes C-Kermit to execute the command. |
| Linefeed | Works the same as carriage return. |
| Formfeed | Works the same as carriage return but clears the screen before executing the command. |
| Backslash (\) | Allows you to enter any of the above characters literally. Enter two backslashes to enter a single backslash. |

The following example demonstrates the C-Kermit interactive mode help facility and use of the Esc key. User entries are underscored, and "$" is used to represent the Esc key.

```
C-Kermit>set ? parameter, one of the following:
block-check        delay            duplex           end-of-packet
escape-character   file             flow-control     handshake
line               modem-dialer     packet-length    pad-character
padding            parity           prompt           speed
start-of-packet    timeout

C-Kermit>set d? parameter, one of the following:
delay         duplex

C-Kermit>set du$plex ?  one of the following:
full          half

C-Kermit>set duplex h$alf

C-Kermit>set du h
```

## 2.2 Notation Conventions

The command-line and interactive mode command descriptions use the following notation:

| | |
|---|---|
| fn | An ambiguous local file specification |
| fn1 | An unambiguous local file specification |
| rfn | An ambiguous remote file specification |
| rfn1 | An unambiguous remote file specification |
| n | A decimal number between 0 and 94 |
| c | An ASCII control character (decimal values 0 to 31 and 127) |
| dev | A logical device name |
| [ ] | Square brackets designate optional arguments |
| { } | Curly brackets designate alternative arguments |

All file specifications in C-Kermit commands can include a directory path.

## 2.3 Command-line Mode Commands

The command-line mode has two types of commands: one set invokes an action, such as a file transfer; the other set controls various aspects of file transfers, such as the baud rate and write-protect mode. If you do not enter an action command, C-Kermit leaves you in interactive mode after completing the transfer-control command. Table 2-2 lists the command-line mode action commands. Table 2-3 lists the commands used to control file transfer. Examples of these and the action commands follow this table.

### 2.3.1 Command-line Syntax

The C-Kermit parser allows you to enter multiple commands per entry and enter them out of the order of execution. For example, the following command has C-Kermit make a connection (-c), receive a file (-r), and then make another connection after receiving the file (-n):

```
C-KERMIT -c -n -r
```

You can enter the commands individually, with the commands separated by a space, or in groups. When you enter commands individually, each command must be preceded by a - (dash). When you group the commands, put a dash before the first command only. For example, the following command performs the same operation as the above command.

```
C-KERMIT -cnr
```

**Table 2-2. Action Commands**

| Command | Description |
|---|---|
| The following commands can be invoked on the local computer or the remote computer after a connection is established (-c or -n commands). | |
| -s fn | Send the file or files named in fn. |
| -r | Receive a file or files. C-Kermit waits to receive at least one file. |
| -k | Receive a file or files and write them to the screen instead of the disk. |
| -a fn1 | Name the file sent or received fn1. This command is used only with the -s or -r commands. |
| -x | Start server operation. |
| The following commands can be invoked only on the local computer. | |
| -c | Start a connection. This command puts you in dumb terminal mode. Everything entered, except for the special characters described on page 2-2, is output through selected serial port (see the -l command in Table 2-3). |
| -g rfn | Get the file or files specified by rfn from a server. Use -g rather than -r when the remote computer is in server mode. |
| -f | Terminate server mode on the remote computer. |
| -n | Put the local computer in dumb terminal mode after the completion of a -s, -r, -k, -g, or -f command. |
| -h | Display a brief description of C-Kermit commands. |

**Table 2-3. Transfer Characteristics Commands**

| Command | Description |
|---|---|
| -l dev | Select the serial port. |
| -b n | Select the baud rate. |
| -p x | Set parity where x is one of the following:<br>e even<br>o odd<br>m mark<br>s space<br>n none (default) |
| -t | Select half duplex communication. C-Kermit uses XON as the handshake character. The default communication mode is full duplex. |
| -i | Put C-Kermit in image mode. You must use image mode to transfer binary files. |
| -w | Select write protection. If you do not enable write protection, C-Kermit overwrites the existing file when the incoming file has the same name. |
| -q | Suppress screen update. Use this command when you run C-Kermit in the background. |
| -d | Enable debug mode for the following action. While C-Kermit is in debug mode, it records information in the file DEBUG.LOG (DEBUG.LOG is described in the LOG command description on page 2-14). |

### 2.3.2 Examples

The following examples demonstrate action and transfer commands.

```
    C-KERMIT -l ser: -b 1200 -cn -r
```

This command first selects the port defined by ser:, sets the baud rate to 1200 BPS, and makes a connection to the remote computer. C-Kermit then waits to receive a file after the user breaks the connection. After the file transfer is complete, C-Kermit re-establishes the connection leaving the local computer in dumb terminal mode.

```
    C-KERMIT -cntp m -r -a myfile
```

This command establishes a connection (-c) using half duplex (t) and mark parity (p m). After the user breaks from the connection, C-Kermit waits to receive a file. The file received is saved under the name "myfile." After receiving the file, C-Kermit re-establishes the connection.

```
    C-KERMIT -nf
```

This command sends a finish command to the remote server and then establishes a connection to the remote system.

```
    C-KERMIT -qg myfile.*
```

This command requests the remote Kermit server to send all files with the "myfile" filename and any or no extension. Screen output is suppressed and keyboard input is not sampled, allowing C-Kermit to run in the background.

```
    C-KERMIT -iwx
```

This command puts C-Kermit in server mode and enables it to receive binary files. Incoming file names that already exist on the local system are renamed.

```
    C-KERMIT
```

This command invokes interactive mode with all default settings.

## 2.4 Interactive Mode Commands

C-Kermit's interactive mode supports a series of commands, some of which take an additional argument. You can abbreviate the commands to any length that makes them distinguishable from other commands valid for that field. Type a question mark at any time to get information about what's expected or valid at that point. You can enter commands in uppercase or lowercase.

Table 2-4 lists the commands in task groupings. The command descriptions are presented in alphabetical order following the table.

**Table 2-4. Interactive Mode Command Summary**

To exchange a file or files:

RECEIVE: Receive a file or files
SEND: Send a file or files
GET: Get a file or files

To connect to a remote system:

SET: Set line and transfer characteristics
CONNECT: Connect to remote computer in dumb terminal mode
DIAL: Dial a telephone number

To enable and talk to the server:

SERVER: Start server on local or remote computer
REMOTE: Execute file management commands on remote server
FINISH: Terminate remote server

To get information:

HELP: Display summary of C-Kermit commands
SHOW: Display current parameter values
STATISTICS: Display statistics on the latest transaction

**Table 2-4 Continued.**

To manage files:

CHDIR: Change current directory.
DELETE: Delete a file.
DIR: Display the current directory.
SPACE: Display amount of disk space used.
TYPE: Display a file.

To execute a sequence of commands:

SCRIPT: Execute log on script for the remote computer.
TAKE: Execute sequence of commands from specified file.

To record transaction information:

LOG: Create packet, session, debug, or transaction log file.
CLOSE: Close packet, session, debug, or transaction log file.

To terminate Kermit:

BYE: Terminate remote server and log off remote host.
EXIT: Close log files and terminate local C-Kermit.
FINISH: Terminate remote Kermit server.
QUIT: Close log files and terminate local C-Kermit (same as EXIT).

### 2.4.1 FlexOS Command

```
C-Kermit> ! command
```

This command executes the specified FlexOS command. After the command completes, the user returns to C-Kermit interactive mode.

### 2.4.2 BYE

```
C-Kermit> BYE
```

This command terminates server mode on the remote Kermit and logs off the remote host.

### 2.4.3 CHDIR

```
C-Kermit> CHDIR dir_path
```

CD changes the current directory to the directory specified in dir_path. If a path is not specified, CHDIR displays the current directory path.

### 2.4.4 CONNECT

```
C-Kermit> CONNECT
```

CONNECT links your terminal to another computer in dumb terminal mode. While in this mode, all keyboard input except for the special escape characters is output to the selected serial port. Similarly, all characters received on this line are displayed on the console.

The default serial port is used unless you changed it with the SET LINE command. Current settings of speed, parity, duplex, and flow-control are in effect. If you have issued a LOG SESSION command, everything displayed on your screen is recorded in your session log file.

To terminate dumb terminal mode, enter the escape character (Ctrl-\) followed by 'c'.

CONNECT does not interpret incoming characters before sending them to the screen; that is, no terminal emulation is provided. If you want terminal emulation, you must pipe C-Kermit output to a terminal emulation program.

### 2.4.5 CLOSE

```
C-Kermit> CLOSE fnl
```

CLOSE closes the log file specified by fn1. Note that if you do not close a log file, C-Kermit closes them in response to EXIT or QUIT commands.

### 2.4.6 DELETE

```
C-Kermit> DELETE fn
```

DELETE erases the file or files specified by fn.

### 2.4.7 DIAL

```
   C-Kermit> DIAL telephone-number-string
```

DIAL controls dialout modems. If the telephone-number-string contains modem-dialer commands, you must define the modem-type and if necessary the port (see SET) before invoking DIAL.

For Hayes® dialers, two important switch settings are #1 and #6. Set #1 up so that DTR is asserted only when the line is 'open'. Set #6 up so that carrier-detect functions properly. Switches #2 (English versus digit result codes) and #4 (Hayes echoes modem commands) may be in either position.

### 2.4.8 DIR

```
   C-Kermit> DIR [fnl]
```

DIR lists all files in the current directory that match the file specification. If no file specification is entered, all files are displayed.

### 2.4.9 ECHO

```
   C-Kermit> ECHO string
```

ECHO displays the string on screen. This command is for use in TAKE files to display greetings, progress reports, and so forth.

### 2.4.10 EXIT

```
   C-Kermit> EXIT
```

EXIT terminates C-Kermit operation. Termination events include the following:

- Attempt to insure that the terminal is returned to normal.
- Relinquish the communication line assigned by SET LINE.
- Close open log files.
- Relinquish lock on the communications line.
- Hang up the modem, if the communications line supports DTR.

After EXIT from C-Kermit, your current directory is the same as when you started the program.

EXIT is issued implicitly whenever C-Kermit halts normally; for example, after a command line invocation or after certain kinds of interruptions.

### 2.4.11 FINISH

```
   C-Kermit> FINISH
```

FINISH terminates server mode on the remote Kermit without logging off the remote host.

### 2.4.12 GET

```
   C-Kermit> GET rfn
   C-Kermit> GET
```

The first example requests the file or files named rfn from the remote Kermit. The remote Kermit must be in server mode. In the second example, C-Kermit prompts you for the remote filename to get and the local filename to store it under. Use this form when the remote file name contains a space.

### 2.4.13 HELP

```
   C-Kermit> HELP
   C-Kermit> HELP command
   C-Kermit> HELP SET keyword
   C-Kermit> HELP REMOTE command
```

HELP displays explanations of C-Kermit commands. Brief help messages or menus are also available at interactive command level by typing a question mark at any point.

The HELP command with no arguments prints a brief summary of how to enter commands and how to get further help. HELP may be followed by a top-level C-Kermit command, such as SEND, to request information about a command. Commands such as SET and REMOTE have a further help level. For example, you can type the following to get help on the SET parity option:

```
   C-Kermit>HELP SET PARITY
```

### 2.4.14 LOG

```
C-Kermit> LOG {qualifier}[fnl]
```

LOG records transaction information in the file specified by fn1 or, if no name is specified, a default file. The type of information recorded depends upon the qualifier specified. The qualifiers are listed in Table 2-5.

**Table 2-5. LOG Command Qualifiers**

| Qualifier | Default File Name | Description |
|---|---|---|
| debugging | debug.log | Record internal C-Kermit transactions. This option degrades C-Kermit performance dramatically but provides an exhaustive log file. |
| packets | packet.log | Record all incoming and outgoing packets. This option is useful for tracking protocol problems with either the remote or local Kermit. |
| session | session.log | Record everything sent to the screen after a CONNECT command. Local messages and escape commands are not recorded. This option is useful for recording ASCII files from systems that do not have Kermit available. |
| transactions | transact.log | Record the name of all files sent and received with their time stamps, statistics, filename transformations, and transmission errors. |

### 2.4.15 QUIT

```
C-Kermit> QUIT
```

Terminates C-Kermit operation. QUIT is the same as EXIT. See the EXIT description for the explanation of termination events.

### 2.4.16 RECEIVE

```
C-Kermit> RECEIVE
C-Kermit> RECEIVE fnl
```

In the first example, C-Kermit waits to receive a file or files from another Kermit executing a SEND command. In the second example, C-Kermit waits for a file or files and names the first one received fn1.

### 2.4.17 REMOTE

```
C-Kermit> REMOTE command
```

REMOTE provides a means of executing file management commands on a remote computer in server mode. Table 2-6 lists the REMOTE command options.

**Table 2-6. REMOTE Commands**

| REMOTE Command | Remote Kermit Server Response) |
|---|---|
| CHDIR [directory] | Changes the current directory. |
| DELETE rfn | Deletes the remote file or files. |
| DIR [rfn] | Displays directory listing of remote file(s). |
| HOST command | Executes the command on the remote host. |
| SPACE | Reports disk usage on remote host. |
| TYPE [rfn] | Displays remote file(s) on your screen. |
| WHO | Tells who is logged on to remote host. |
| HELP | Displays remote kermit's server abilities. |

### 2.4.18 SCRIPT

```
C-Kermit> SCRIPT expect send [expect send . . .]
```

SCRIPT facilitates logging on and invoking programs on the remote computer. The expect argument is the prompt or message you expect the remote system to issue and send is a string (names, numbers, etc) you want to send in response. Separate the expect and send strings with a space. All SCRIPT commands must end with a send.

The SCRIPT command supports the following special characters:

| | |
|---|---|
| ~b | backspace |
| ~s | space |
| ~q | '?' (trapped by Kermit's command interpreter) |
| ~n | linefeed |
| ~r | carriage return |
| ~t | tab |
| ~' | single quote |
| ~~ | tilde |
| ~" | double quote |
| ~c | don't append a carriage return |
| ~ | an octal character (3-digit maximum) |

All strings sent are followed by a carriage return unless you include a ~c. Note that '\' characters in scripts must be doubled up.

Only the last 7 characters in expect strings are matched. A null expect (~0) or two adjacent hyphens (--) causes a short delay before proceeding to the next send sequence. A null expect always succeeds.

If the expect string does not arrive, the SCRIPT attempt fails. To catch strings that might arrive, C-Kermit provides a conditional expect sequence in the form:

```
expect-send-expect[-send-expect[...]]
```

where the hyphen-delimited send sequences (-send-) are followed as long as the previous expect fails. A null send (-~0-) or two adjacent hyphens (--) causes a carriage return to be sent. The examples below demonstrate conditional expect sequences.

You can debug SCRIPT commands by logging transactions (see the LOG command). This records all exchanges, both expected and actual.

The following examples show SCRIPT command use in a TAKE file.

### Example--Dial UNIX™ Host Site

The SCRIPT commands use a conditional expect sequence to wait for "login:" (gin:) and, if it doesn't come, to send a carriage return. (Because all sends by default include a carriage return, sending a null sends a carriage return.) Next, the user ID and password are sent. The first SCRIPT command ends by sending an x after a question mark (~q) arrives. The second SCRIPT command begins with a conditional expect for the Bourne shell $ prompt, then changes the directories and finally invokes the program wermit. Note the ~s use in the cd command to designate a space. If a space had been used instead, the send would have terminated with cd.

```
    set modem-dialer hayes
    set line /dev/tty77
    set baud 1200
    dial 9&5551212
    script gin:--gin:--gin: smith ssword: mysecret ~q x
    script $--$ cd~skermit $ wermit
    connect
```

### Example--Dial a Telenet™ Network

Telenet expects three returns with slight delays between them. These are sent following null expects. Four returns are sent to be safe before looking for the prompt. Next, the id and password are entered and telenet is instructed to connect to a host site (c 12345). The host has a data switch, and to "which system" it responds "myhost." This is followed by a TOPS-20 logon, a request to load Kermit, set even parity, and enter the Kermit server mode.

```
    set modem-dialer hayes
    set line /dev/cul0
    set baud 1200
    dial 9,5551212
    set parity even
    script ~0 ~0 ~0 ~0 ~0 ~0 ~0 ~0 @--@--@ id~saa001122 = 002211
    script @ c~s12345 ystem-c~s12345-ystem myhost @ joe~ssecret
    script @ kermit > set~sparity~seven > server
    send some.stuff
    get some.otherstuff
    bye
    quit
```

### 2.4.19 SEND

```
C-Kermit> SEND fn
C-Kermit> SEND fn rfnl
```

The first example sends the file or files specified by fn to the remote computer. The remote computer must be running in receive or server mode. The second example sends the local file fn to the remote kermit under the name rfn1.

**Note:** C-Kermit sends files from the current directory unless the file name is preceded by a path; it cannot traverse directories or create directories. C-Kermit does not ignore hidden files.

### 2.4.20 SERVER

```
C-Kermit> SERVER
```

SERVER places the computer in server mode. SERVER can be invoked locally or on the remote computer from dumb terminal mode. C-Kermit expects subsequent commands to come in packet form from another Kermit. In server mode, C-Kermit responds to the following commands only. The server response is shown in parenthesis.

GET (sends the file or files)
SEND (receives the file or files)
BYE (exits server mode and logs off host)
FINISH (exits server mode but does not log off)
REMOTE command (executes the command on the remote computer)

See the REMOTE command description for the list of REMOTE commands supported.

### 2.4.21 SET

```
C-Kermit> SET {keyword argument}
```

SET allows you to configure C-Kermit to accommodate a variety of environments. You make changes by entering SET followed by a keyword and a single argument. The keywords and arguments are defined as follows:

| <u>Keyword</u> | <u>Arguments</u> |
|---|---|
| BLOCK-CHECK | 1, 2, or 3 |

Sets the level of per-packet error detection. The values are defined as follows:

1 Single character 6-bit checksum
2 Two character 12-bit checksum
3 Three character, 16-bit cyclic redundancy check

The higher the block check value the greater the error detection and overhead. The default value of 1 is sufficient in most cases. Use 2 or 3 when transferring binary data over noisy lines.

| <u>Keyword</u> | <u>Arguments</u> |
|---|---|
| DELAY | n |

Sets the number of seconds to wait before sending packets following a SEND command. The default value is 5 seconds. Invoke this command on the remote computer if you need more time to escape back to your local computer to issue the RECEIVE command.

| <u>Keyword</u> | <u>Arguments</u> |
|---|---|
| DUPLEX | FULL or HALF |

Determines which system echos keyboard input during dumb terminal mode operation. Use FULL to have the remote computer perform the echo; HALF to have the local C-Kermit echo input.

| <u>Keyword</u> | <u>Arguments</u> |
|---|---|
| END-OF-PACKET | cc |

Sets the end-of-packet character. C-Kermit sends this character at the end of each packet. Usually this value is 0DH (carriage return). Some Kermits do not require an end of packet character while others require a different character.

| <u>Keyword</u> | <u>Arguments</u> |
|---|---|
| ESCAPE-CHARACTER | cc |

Sets the escape character used to abort connect mode and return to the interactive prompt. The default escape character is Ctrl-\ (1CH).

| Keyword | Arguments |
|---|---|
| FILE | DISPLAY, NAMES, TYPE, or WARNING |

Sets file related parameters. The arguments and their qualifiers are defined as follows:

| Argument | Qualifier |
|---|---|
| DISPLAY | ON or OFF |

Determines whether C-Kermit displays a progress indicator and allows interruption during file transfers. The default value is ON. If OFF, no progress indicator is shown and you cannot abort the transfer.

| Argument | Qualifiers |
|---|---|
| NAMES | CONVERTED or LITERAL |

Determines where file names are converted. The default value is converted. Converted outgoing filenames have path specifications stripped, lowercase letters raised to upper, tildes and extra periods changed to X and an X inserted in front of any name beginning with a period. Converted incoming files have uppercase letters lowered. Literal means that no filename conversions are done. Therefore, any directory path appearing in a received file specification must exist and be write-accessible.

| Argument | Qualifier |
|---|---|
| TYPE | BINARY or TEXT |

Determines whether each newline character in the file is converted to a carriage return and linefeed sequence. In text mode, newline characters are converted; in binary, they are not. The default is text mode.

| Argument | Qualifiers |
|---|---|
| WARNING | ON or OFF |

Determines whether C-Kermit automatically overwrites the local file when the incoming file has the same name or issues a warning first. The default is OFF. When on, C-Kermit checks for a match between the incoming file and the files in the current or specified directory. When there is a match, C-Kermit names the arriving file in the form filename.000. For example, if a file named JOB is present and incoming, C-Kermit records the incoming file as JOB.000; if JOB and JOB.000 exist, the incoming file is named JOB.001, and so forth.

| Keyword | Arguments |
|---|---|
| FLOW-CONTROL | NONE or XON/XOFF |

Determines whether xon/xoff handshaking is used to control program transfer. The default value is XON/XOFF. Enter NONE if the other system is incapable of XON/XOFF handshaking.

| Keyword | Arguments |
|---|---|
| HANDSHAKE | XON, XOFF, CR, LF, BELL, ESC, or NONE |

Deterines whether full- or half-duplex communication is performed and, if half-duplex, the turnaround handshake character. The default is NONE; this selects full-duplex communication. If you select any other value, half-duplex line turnaround handshaking is done. In half-duplex mode, C-Kermit replies to a packet only after it receives the handshake character selected or times out.

| Keyword | Arguments |
|---|---|
| LINE | device-name |

Selects the serial port used for file transfer and terminal connection. The default name is SER:.

| Keyword | Arguments |
|---|---|
| MODEM-DIALER | DIRECT, HAYES, or VENTEL |

Defines the type of modem-dialer (see the DIAL command description). The default value is DIRECT (no dialout modem).

| Keyword | Arguments |
|---|---|
| PACKET-LENGTH | n |

Specifies the maximum packet length to use. The default value is 90 bytes. Shorter packets are usefull on noisy lines or on systems with small buffers. The shorter the packet, the greater the overhead.

| Keyword | Arguments |
|---|---|
| PAD-CHARACTER | cc |

Specifies the pad character used to precede packets. The default pad character is NUL, ASCII 0.

| Keyword | Arguments |
|---|---|
| PADDING | n |

Defines the number of pad characters to ask for. The default value is 0.

| Keyword | Arguments |
|---|---|
| PARITY | EVEN, ODD, MARK, SPACE, or NONE |

Specifies the character parity to use in packet transmission and terminal connections. The default value is none. If parity is even, odd, mark, or space, C-Kermit attempts to use 8th-bit prefixing for transferring 8-bit binary data. This can only be done if the other Kermit agrees. When there is no agreement, 8-bit binary data cannot be transferred.

| Keyword | Arguments |
|---|---|
| PROMPT | string |

Specifies the interactive prompt. The default value is "C-Kermit>".

| Keyword | Arguments |
|---|---|
| SPEED | 0, 50, 75, 110,134.5, 150, 300, 600, 1200, 1800 2000, 2400, 3600,4800, 7200, 9600, or 19200 |

Selects the baud rate. The default value is 9600.

| Keyword | Arguments |
|---|---|
| START-OF-PACKET | cc |

Selects the character that signals the start of a packet. The default value is Ctrl-A (1). Do not change this value unless some software or hardware between the two Kermits traps or echoes this character, not allowing it to pass through the line.

| Keyword | Arguments |
|---|---|
| TIMEOUT | n |

Sets the packet interval timeout period. Normally, the initiating Kermit sets its packet timeout interval based on the opposite Kermit's requests. Use TIMEOUT to override this procedure and specify your own timeout. A value of 0 disables time outs; that is, C-Kermit waits forever for expected packets.

### 2.4.22 SHOW

```
C-Kermit> SHOW {parameters, versions}
```

The SHOW qualifiers select the following options:

| | |
|---|---|
| parameters | Displays the current values for all SET parameters. |
| versions | Displays the version number and dates of the internal modules. |

### 2.4.23 SPACE

```
C-Kermit> SPACE
```

SPACE displays the remaining space on the current drive.

### 2.4.24 STATISTICS

```
C-Kermit> STATISTICS
```

STATISTICS displays information about the last protocol transaction, including file and communication line I/O, and the current encoding options in effect (such as 8th-bit prefixing, repeat-count compression).

### 2.4.25 TAKE

```
C-Kermit> TAKE fnl
```

TAKE instructs C-Kermit to execute commands from the specified file. The file can contain any interactive C-Kermit commands, including TAKE. TAKE files can be nested to any reasonable depth. Use the ECHO command within the TAKE file to issue greetings, announce progress, etc.

Commands in the TAKE file follow the exact syntax of a command entered interactively. This means that special characters like a question mark must be quoted with a backslash if they are to be interpreted literally. See the SCRIPT command description for examples of TAKE files.

Commands executed from TAKE files are not echoed at the terminal. If you want to see the commands as well as their output, feed the command file to C-Kermit via redirected stdin.

Errors encountered during the execution of a TAKE file (such as failure to complete a DIAL or SCRIPT operation) cause termination of the current TAKE file, popping to the TAKE file that invoked it or to interactive mode. When C-Kermit is executed in the background, errors during execution of a TAKE file are fatal.

### 2.4.26 TYPE

```
    C-Kermit> TYPE fnl
```

TYPE displays the specified file on the console screen.

End of Section 2

# When Things Go Wrong

Connecting two computers can be a tricky business, and many things can go wrong. Before you can transfer files at all, you must first establish terminal communication. But successful terminal connection does not necessarily mean that file transfer will also work. And even when file transfer seems to be working, things can happen to ruin it.

## 3.1 Communication Line Problems

If you have a version of C-Kermit on your microcomputer, but the CONNECT command doesn't seem to work at all, please do the following:

- Make sure all the required physical connections have been made and have not wiggled loose. If you are using a modem, make sure the carrier light is on.
- If you have more than one connector on your micro, make sure you are using the right one.
- Make sure that the port is set to the right communication speed, or <u>baud rate</u>. Use the SHOW command to find out what the current baud rate is. C-Kermit has a SET BAUD command if you need to change the baud rate.
- Make sure that the other communication line parameters, like parity, bits per character, handshake, and flow control are set correctly.

You must consult the appropriate manuals for the systems and equipment in question.

If all settings and connections appear to be correct, and communication still does not take place, the fault may be in your modem. Internal modems (that plug into a slot inside the microcomputer chassis) do <u>not</u> work with C-Kermit.

C-Kermit normally expects to have full control of the communication port. However, it is sometimes the case that some communications equipment controls the line between the two computers on either end. Examples include modems (particularly "smart" modems), port contention or selection units, multiplexers, local networks, and wide-area networks. Such equipment can interfere with the C-Kermit file transfer protocol in various ways:

- It can impose parity upon the communication line. This means that the 8th bit of each character is used by the equipment to check for correct transmission. Use of parity will:
  - Cause packet checksums to appear incorrect to the receiver and foil any attempt at file transfer. In most cases, not even the first packet will get through.
  - Prevent the use of the 8th bit for binary file data.
- If terminal connection works but file transfer does not, parity is the most likely culprit. To overcome this impediment, you should find out what parity is being used and inform C-Kermit using the SET PARITY command.
- Communications equipment can also interpret certain characters in the data stream as commands rather than passing them along to the other side. For example, you might find your smart modem suddenly disconnecting you and placing a call to Tasmania. To work around such problems, put the device into "transparent" or "binary" mode. Refer to your modem manual for this procedure. In some cases, transparent mode also cancels parity processing and allows the use of the 8th bit for data.

## 3.2 The Transfer is Stuck

There are various ways in which C-Kermit file transfers can become stuck, but since many hosts are capable of generating timeout interrupts when input doesn't appear quickly enough, they can usually resend or "NAK" (negatively acknowledge) lost packets. Nevertheless, if a transfer seems to be stuck, you can type Ctrl-C or Ctrl-Break to simulate a timeout.

The following sections discuss various reasons why a transfer in progress could become stuck. Before examining these, first make sure that you really have a Kermit on the other end of the line, and you have issued the appropriate command: SEND, RECEIVE, or SERVER. If the remote side is not a server, remember that you must connect back between each transfer and issue a new SEND or RECEIVE command.

## 3.3 The Local System Stops

The local system itself sometimes crashes for reasons beyond C-Kermit's control, such as power fluctuations. If the console has not been updated for a long time, try these steps (in the following order) to restart:

- Check the connection. Make sure no connectors have wiggled loose from their sockets. If you're using a modem, make sure you still have a carrier signal. Reestablish your connection if you have to.
- Press Ctrl-C. This should clear up any protocol deadlock.
- If the problem was not a deadlock, restart the system and then restart C-Kermit, You may have to stop and restart C-Kermit on the remote system.

## 3.4 The Remote System Went Away

If your local system is working but the transfer is not proceeding, the remote system or the remote C-Kermit program might have crashed.

## 3.5 The Disk is Full

If your local floppy disk or remote directory is fullm, the C-Kermit on the machine where this occurs informs you and then terminates the transfer. You can continue the transfer by repeating the whole procedure either with another floppy or removing files.

## 3.6 Message Interference

You may find that file transfers fail occasionally and unpredictably. One explanation could be that terminal messages are being mixed with your file packet data. These could include system broadcast messages (like "System is going down in 30 minutes"), messages from other users ("Hi Fred, what's that C-Kermit program you're always running?"), notifications that you have requested ("It's 7:30, go home!" or "You have mail from..."). Most Kermit programs attempt to disable intrusive messages automatically, but not all can be guaranteed to do so. It may be necessary for you to "turn off" such messages before starting C-Kermit.

## 3.7 Remote System Errors

Various error conditions can occur on the remote system that could effect file transmission. Whenever any such error occurs, the remote Kermit normally attempts to send an informative error message to the local one, and then breaks transmission, putting you back at C-Kermit command level on the local system.

## 3.8 File is Corrupted

There are certain conditions under which C-Kermit's error detection mechanism fails. The most frequent causes are problems related to the file attributes, such as text vs binary, 7-bit vs 8-bit, blocked vs stream, and so forth. Some systems have their own peculiarities and for them C-Kermit has special commands to allow you to specify how a file should be sent or stored. However, these difficulties usually crop up only when sending binary files. Textual files should normally present no problem between any two C-Kermit programs.

End of Section 3