DECSYSTEM-20 KERMIT USER GUIDE

F. da Cruz, C. Gianone

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

January, 1988

1. DECSYSTEM-20 KERMIT

Authors: Frank da Cruz, Bill Catchings, Columbia University
Language: MACRO-20
Version: `4.2(262)`
Date: January 1988

Kermit-20 Capabilities At a Glance:

Local operation:	Yes
Remote operation:	Yes
Transfers text files:	Yes
Transfers binary files:	Yes
Wildcard send:	Yes
`^X/^Y` interruption:	Yes
Filename collision avoidance:	Yes
Timeouts:	Yes
8th-bit prefixing:	Yes
Repeat character compression:	Yes
Alternate block check types:	Yes
Communication settings:	Yes
Transmit BREAK:	Yes
IBM mainframe communication:	Yes
Transaction logging:	Yes
Session logging:	Yes
Debug logging:	Yes
Raw transmit:	Yes
Login scripts:	Yes
Act as server:	Yes
Talk to server:	Yes
Advanced commands for servers:	Yes
Local file management:	Yes
Command/init files:	Yes
Long packets:	No
Sliding windows:	No
Handle file attributes:	No

Kermit-20 is a program that implements the Kermit file transfer protocol for the Digital Equipment Corporation DECSYSTEM-20 mainframe computer. It is written in MACRO-20 assembly language and should run on any DEC-20 system with version 4 of TOPS-20 or later.

The Kermit-20 section will describe the things you should know about the DEC-20 file system in order to make effective use of Kermit, and then it will describe the special features of the Kermit-20 program.

1.1. The DEC-20 File System

The features of the DEC-20 file system of greatest interest to Kermit users are the form of the file specifications, and the distinctions between text and binary files.

DEC-20 File Specifications

DEC-20 file specifications are of the form

```
DEVICE:<DIRECTORY>NAME.TYPE.GEN;ATTRIBUTES
```

where the DIRECTORY, NAME, and TYPE may each be up to 39 characters in length, GEN is a generation (version number), and various attributes are possible (protection code, account, temporary, etc). Generation and attributes are normally omitted. Device and directory, when omitted, default to the user's own (or "connected") disk and directory. Thus `NAME.TYPE` is normally sufficient to specify a file, and only this information is sent along by Kermit-20 with an outgoing file.

The device, directory, name, and type fields may contain uppercase letters, digits, and the special characters "`-`" (dash), "`_`" (underscore), and "`$`" (dollar sign). There are no imbedded or trailing spaces. Other characters may be included by prefixing them (each) with a Control-V. The fields of the file specification are set off from one another by the punctuation indicated above.

The device field specifies a physical or "logical" device upon which the file is resident. The directory field indicates the area on the device, for instance the area belonging to the owner of the file. Kermit-20 does not transmit the device or directory fields to the target system, and does not attempt to honor device or directory fields that may appear in incoming file names; for instance, it will not create new directories.

The name is the primary identifier for the file. The type, also called the "extension", is an indicator which, by convention, tells what kind of file we have. For instance `FOO.FOR` is the source of a Fortran program named FOO; `FOO.REL` might be the relocatable object module produced by compiling `FOO.FOR`; `FOO.EXE` could an executable program produced by LOADing and SAVing `FOO.REL`, and so forth.

The DEC-20 allows a group of files to be specified in a single file specification by including the special "wildcard" characters, "`*`" and "`%`". A "`*`" matches any string of characters, including no characters at all; a "`%`" matches any single character. Here are some examples:

`*.FOR` All files of type `FOR` (all Fortran source files) in the connected directory.

`FOO.*` Files of all types with name `FOO`.

`F*.*` All files whose names start with F.

`F*X*.*` All files whose names start with F and contain at least one X.

`%.*` All files whose names are exactly one character long.

`*.%%%*` All files whose types are at least three characters long.

Wildcard notation is used on many computer systems in similar ways, and it is the mechanism most commonly used to instruct Kermit to send a group of files.

Text Files and Binary Files

The DEC-20, like most computers, has a file system with its own peculiarities. Like many other systems, the DEC-20 makes a distinction between *text files* and *binary files*. Text files are generally those composed only of printing characters (letters, digits, and punctuation) and "carriage control" characters (carriage return, line feed, form feed, tab). Text files are designed to be read by people. Binary files are designed to be read by a computer program, and may have any contents at all. If you use the DEC-20 TYPE command to display a text file on your terminal, the result will be intelligible. If you type a binary file on your terminal, you will probably see mainly gibberish. You can not always tell a text file from a binary file by its name or directory information, though in general files with types like `.TXT`, `.DOC`, `.HLP` are textual (as are "source files" for computer programs like text formatters and programming language compilers), and files with types like `.EXE`, `.REL`, `.BIN` are binary.

The DEC-20 has an unusual word size, 36 bits. It differs from most other systems by storing text in 7-bit, rather than 8-bit, bytes. Since text is encoded in the 7-bit ASCII character set, this allows more efficient use of storage. However, the word size is not a multiple of the normal byte size. The DEC-20 therefore stores five 7-bit characters per word, with one bit left over.

It is also possible to store files with other byte sizes. The common layouts of bytes within a word are shown in Figure 1-1.

7: Text Files: Five 7-bit bytes per word.

```
+------+------+------+------+------++
|      |      |      |      |      ||
+------+------+------+------+------++
0      7      14     21     28     35
```

Normally, bit 35 is unused and set to zero. However, in EDIT (or SOS, or OTTO) line-numbered files, bit 35 is set to 1 when the word contains a line number.

8: "Foreign" binary files: Four 8-bit bytes per word.

```
+-------+-------+-------+-------+---+
|       |       |       |       |   |
+-------+-------+-------+-------+---+
0       8       16      24      32  35
```

Bits 32-35 are unused.

36: "Native" binary files: One 36-bit byte per word.

```
+-----------------------------------+
|                                   |
+-----------------------------------+
0                                   35
```

All bits are used.

Figure 1-1: DECSYSTEM-20 Word/Byte Organization

The minimum unit of disk allocation on the DEC-20 is a *page*, 512 36-bit words, or 2560 7-bit characters, or 2048 8-bit bytes. Any file that contains at least one bit of information occupies at least a full page on the disk. The directory information for a file includes the number of pages occupied on the disk, the bytesize of the file, and the number of bytes of that size which are in the file. This information can be seen by using the DEC-20 VDIRECTORY command, for instance

```
@vdir foo.*

   PS:<MY-DIRECTORY>
 Name     Protection    Pages Bytes(Size) Creation
 FOO.COM.1;P774242          1 384(8)     27-Dec-83
    MAC.1;P774242           1 152(7)     27-Dec-83
   .REL.1;P774242           1 39(36)     27-Dec-83
   .EXE.1;P774242           2 1024(36)   27-Dec-83

 Total of 5 pages in 4 files
```

In this example, `FOO.MAC` occupies 1 page, and is composed of 152 7-bit bytes. This file is textual (program source for the MACRO assembler), 152 characters long. Programs which read text files (such as text editors, program compilers, the TYPE command, etc) determine the end of a file from the byte count specified in the directory. Kermit-20 determines the end of file in the same way, so although `FOO.MAC` occupies an entire 2560-

byte page of storage, only the first 152 characters are transmitted. Binary files, such as FOO.EXE (an executable DEC-20 program), tend to occupy full pages. In this case too, Kermit-20 uses the byte count to determine the end of file.

Why do you need to know all this? In most cases, you don't. It depends on whether you are using the DEC-20 as your "home base".

Using a Microcomputer to Archive DEC-20 Files

Most computers (other than the DEC-10 and DEC-20) store characters in 8-bit bytes. Let's call any such system an 8-bit-byte system. Microcomputers that run CP/M or MS-DOS or PC-DOS, and any computers than run Unix, store these 8-bit bytes in a linear sequence. Certain other 8-bit-byte systems (PDP-11 or VAX systems with FILES-11, IBM mainframes) have more complex file formats. This discussion applies to all linear 8-bit-byte systems, including most popular microcomputers.

Kermit can send any "native" DEC-20 sequential file, text or binary, to an 8-bit-byte system and bring it back to the DEC-20 restored to its original form. If you are using a microcomputer to archive your DEC-20 files, you need never concern yourself with details of byte size or file format. The same holds true between two DEC-20s, or a DEC-10 and a DEC-20.

There is, however, one special complication of which you should be aware. Certain microcomputer operating systems, notably CP/M, do not have an entirely satisfactory way of indicating the end of file. The file length is recorded in blocks rather than bytes. For text files, the end of file is marked within a block by inserting a Control-Z after the last data character. Binary files, however, might easily contain Control-Z characters as data. Therefore, in order not to lose data, these systems must transmit binary files in complete blocks. If the binary file is of foreign origin (for instance, from a DEC-20), and it did not happen to fill up the last block when it was transferred to the micro, then when that file is sent back to the system of origin in "binary mode," junk will appear at the end (if it is sent back in "text mode," it could be truncated at the first data byte that happened to correspond to Control-Z). For DEC-20 programs in .EXE format, this generally has no effect on the runnability or behavior of the program. But for other binary files, particularly internal format numerical data or relocatable program object (.REL) files, the junk could have bad effects. For instance, extraneous data at the end of a .REL file will generally cause LINK to fail to load the file.

Most microcomputer Kermit programs have commands to control end-of-file detection -- commands like SET FILE TEXT, SET FILE BINARY, SET EOF CTRLZ.

Using the DEC-20 to Archive Microcomputer Files

You can use Kermit to send textual files from a microcomputer or any 8-bit system to the DEC-20 with no special provisions, since Kermit-20 stores incoming characters in 7-bit bytes as text unless you explicitly instruct it otherwise. But Kermit-20 has no automatic way of distinguishing an incoming binary file from an incoming text file.[1] Binary files from 8-bit-byte systems generally contain significant data in the 8th bit, which would be lost if the incoming characters were stored in 7-bit bytes, rendering the file useless when sent back to the original system. Thus if you want to use Kermit to store foreign 8-bit binary data on the DEC-20, you must tell it to store such files with a bytesize of 8 rather than 7. This can be the source of much confusion and inconvenience. In particular, you cannot use a "wildcard send" command to send a mixture of text and binary files from an 8-bit-byte system to the DEC-20; rather, you must send all text files with Kermit-20's file bytesize set to 7, and all 8-bit binary files with the bytesize set to 8.

---

[1]Unless the incoming file has an "ITS Binary Header"; see below.

Once you get the foreign binary file into the DEC-20, stored with the correct bytesize (as `FOO.COM` is stored in the example above), you need take no special measures to send it back to its system of origin. This is because Kermit-20 honors the bytesize and byte count from the directory. For instance, if you told Kermit-20 to `SEND FOO.*`, every file in the example above would be transmitted in the correct manner, automatically.

The previous discussion assumes you want to store text files in usable form on the DEC-20. However, if you are using the DEC-20 purely as a repository for your microcomputer files, and you have no desire to display or share the contents of those files on the DEC-20, you can SET FILE BYTESIZE 8 for all incoming files, both text and binary. When the files are sent back to a microcomputer, they will be stored correctly.

Files Kermit-20 Cannot Handle

The Kermit protocol can only accommodate transfer of *sequential* files, files which are a linear sequence of bytes (or words).

Some files on the DEC-20 are not sequential, and cannot be successfully sent or received by Kermit-20. These include directory files, files with holes (missing pages), ISAM files, and RMS files. These files require external information (kept in the DEC-20's file descriptor block and/or index table) in order to be reconstructed; when sending files, Kermit-20 presently transmits only the file name and the contents of the file. External control information and file attributes are not transmitted.

1.2. Program Operation

Kermit-20's prompt is "`Kermit-20>`". Kermit-20 will accept a single command on the Exec command line, like this:

```
@
@Kermit send foo.bar
```

the file is sent

```
@
```

or you can run the program interactively to issue several commands, like this:

```
@
@Kermit

TOPS-20 Kermit version 4.2(262)

Kermit-20>send foo.*
```

files are sent

```
Kermit-20>statistics
```

performance statistics are printed

```
Kermit-20>receive
```

files are received

```
Kermit-20>exit
@
```

During interactive operation, you may use the TOPS-20 help ("?") and recognition (ESC) features freely while typing commands. A question mark typed at any point in a command displays the options available at that point; typing an ESC character causes the current keyword or filename to be completed (or default value to be supplied), and a "guide word" in parentheses to be typed, prompting you for the next field. If you have not typed sufficient characters to uniquely specify the keyword or filename (or if there is no default value) then a beep will be sounded and you may continue typing.

Command keywords may be abbreviated to their shortest prefix that sets them apart from any other keyword valid in that field.

If you have a file called `KERMIT.INI` in your login directory, Kermit-20 will execute an automatic TAKE command on it upon initial startup. `KERMIT.INI` may contain any Kermit-20 commands, for instance SET commands, or DEFINEs for SET macros to configure Kermit-20 to various systems or communications media.

Kermit-20 provides most of the commands possible for an "ideal" Kermit program, as described in the main part of the *Kermit User Guide*. The following sections will concentrate on system-dependent aspects of Kermit-20.

1.3. Remote and Local Operation

Kermit-20 normally runs in remote mode, with the user sitting at a PC. But Kermit-20 can also run in local mode. Local operation of Kermit-20 is useful if the DEC-20 has an autodialer, or a hardwired connection to another computer. When in local mode, file transfer takes place over an assigned TTY line, and Kermit-20 is free to update your screen with status information, and to listen to your keyboard for interrupt characters.

Local Operation of Kermit-20:

```
 DECSYSTEM-20
+---------------------------------------+
|                                       |
|    +---------------------+            |
|    | Your Job            |            |
|    |                     |            |
|    |   +------------+    | <--Commands | Your Job's
|    |   | Kermit-20  +----+-------------+----------------- (-: You
|    |   |            |    | Display---> | Controlling TTY
|    |   |            |    |             |
|    |   |            |    |             |
|    |   |            |    | <--Packets  | Kermit's
|    |   |            +----+-------------+-----------------> Remote
|    |   +------------+    | Packets-->  | Assigned TTY      System
|    |                     |             |
|    +---------------------+             |
|                                        |
+---------------------------------------+
```

Figure 1-2: DEC-20 Kermit Local Operation

Kermit-20 enters local mode when you issue a SET LINE *n* command, where *n* is the octal TTY number of any line other than your own controlling terminal.

1.4. Conditioning Your Job for Kermit

Kermit-20 does as much as it can to condition your line for file transfer. It saves all your terminal and link settings, and restores them after use. However, there are some sources of interference over which Kermit-20 can have no control. In particular, messages issued by superior or parellel forks could become mingled with Kermit packets and slow things down or stop them entirely. For this reason, before using Kermit-20 for any extended period, you should:

- Type the Exec commands SET NO MAIL-WATCH and SET NO ALERTS

- Make sure you don't have any print or batch jobs pending that were submitted with the /NOTIFY option.
- Make sure you don't have any superior or parallel forks that have enabled terminal interrupts on Control-A; these could prevent Kermit packets (which start with Control-A) from getting through.

After running Kermit, you can restore your mail-watch and alerts by hand. Alternatively, you could have an Exec command file for invoking Kermit like this:

```
set no alerts
set no mail-watch
kermit
set mail-watch
set alert  1:00PM Go to lunch
set alert  6:00PM Go to dinner
set alert 11:30PM Go to sleep
```

1.5. Kermit-20 Commands

This section describes the Kermit-20 commands -- in detail where they differ from the "ideal" Kermit, briefly where they coincide. Kermit-20 has the following commands:

BYE	to remote server.
CLEAR	a stuck connection
CLOSE	log file and stop logging remote session.
CONNECT	as terminal to remote system.
CWD	change local working directory.
DEFINE	macros of Kermit-20 commands.
DELETE	local files.
DIRECTORY	listing of local files.
ECHO	a line of text.
EXIT	from Kermit-20.
FINISH	Shut down remote server.
GET	remote files from server.
HELP	about Kermit-20.
INPUT	characters from communication line.
LOCAL	prefix for local file management commands.
LOG	remote terminal session.
OUTPUT	characters to communication line.
PAUSE	between commands.
PUSH	to TOPS-20 command level.
QUIT	from Kermit-20
RECEIVE	files from remote Kermit.
REMOTE	prefix for remote file management commands.
RUN	a DEC-20 program.
SEND	files to remote Kermit.
SERVER	mode of remote operation.
SET	various parameters.
SHOW	various parameters.
SPACE	inquiry.
STATISTICS	about most recent file transfer.
TAKE	commands from a file.
TRANSMIT	a file "raw".
TYPE	a local file.

1.5.1. Commands for File Transfer

Kermit-20 provides the standard SEND, RECEIVE, and GET commands for transferring files using the Kermit protocol.

The SEND Command

Syntax:

Sending a single file:

`SEND` *nonwild-filespec1* `(AS) [`*filespec2*`]`

Sending multiple files:

`SEND` *wild-filespec1* `(INITIAL) [`*filespec2*`]`

The SEND command causes a file or file group to be sent from the DEC-20 to the other system. There are two forms of the command, depending on whether *filespec1* contains wildcard characters ("`*`" or "`%`"). Kermit-20 automatically recognizes the two cases and issues the appropriate guide word, (AS) or (INITIAL), depending on the form of *filespec1*.

Sending a File Group

If *filespec1* contains wildcard characters then all matching files will be sent, in alphabetical order (according to the ASCII collating sequence) by name. If a file can't be opened for read access, it will be skipped. The initial file in a wildcard group can be specified with the optional *filespec2*. This allows a previously interrupted wildcard transfer from where it left off, or it can be used to skip some files that would be transmitted first.

Sending a Single File

If *filespec1* does not contain any wildcard characters, then the single file specified by *filespec1* will be sent. Optionally, *filespec2* may be used to specify the name under which the file will arrive at the target system; *filespec2* is not parsed or validated in any way by Kermit-20, but lower case letters are raised to upper case, and leading "whitespace" (blanks and tabs) are discarded. If *filespec2* is not specified, Kermit-20 will send the file with its own name.[2]

SEND Command General Operation:

Files will be sent with their DEC-20 filename and filetype (for instance `FOO.BAR`, no device or directory field, no generation number or attributes). If you expect to be sending files whose names contain characters that would be illegal in filenames on the target system, and you know that the Kermit on the target system does not have the ability to convert incoming filenames, you can issue the SET FILE NAMING NORMAL-FORM command to have Kermit-20 replace suspect characters by `X`'s.

Each file will be sent according to its bytesize and byte count from the directory unless you specify otherwise using SET FILE BYTESIZE, or unless the file has an "ITS Binary" header. If the bytesize is 8, then four 8-bit bytes will be sent from each DEC-20 36-bit word, and the low order four bits will be skipped. If other than 8, then five 7-bit bytes will be sent from each word, with the 8th bit of the 5th character set to the value of the remaining bit ("bit 35") from the word.[3]

---

[2] Control-V's, which are used to quote otherwise illegal characters in DEC-20 file specifications, are stripped.

[3] This is the same method used by the DEC-20 to encode 36-bit data on "ANSI-ASCII" tapes. It allows not only DEC-20 binary files, but also the line-sequence-numbered files produced by EDIT, SOS, or OTTO, which use bit 35 to distinguish line numbers from text, to be sent and retrieved correctly.

If communication line parity is being used (see SET PARITY), Kermit-20 will request that the other Kermit accept a special kind of prefix notation for binary files. This is an advanced feature, and not all Kermits have it; if the other Kermit does not agree to use this feature, binary files cannot be sent correctly. This includes executable programs (like DEC-20 `.EXE` files, CP/M `.COM` files), relocatable object modules (`.REL` files), as well as text files with line sequence numbers.

Kermit-20 will also ask the other Kermit whether it can handle a special prefix encoding for repeated characters. If it can, then files with long strings of repeated characters will be transmitted very efficiently. Columnar data, highly indented text, and binary files are the major beneficiaries of this technique.

If you're running Kermit-20 locally, for instance dialing out from the DEC-20 to another system using an autodialer, you should have already run Kermit on the remote system and issued either a RECEIVE or a SERVER command. Once you give Kermit-20 the SEND command, the name of each file will be displayed on your screen as the transfer begins; a "`.`" will be displayed for every 5 data packets sucessfully sent, and a "`%`" for every retransmission or timeout that occurs (you may also elect other typeout options with the SET DEBUG command). If the file is successfully transferred, you will see "`[OK]`", otherwise there will be an error message. When the specified operation is complete, the program will sound a beep. If you see many "`%`" characters, you are probably suffering from a noisy connection. You may be able to cut down on the retransmissions by using SET SEND PACKET-LENGTH to decrease the packet length; this will reduce the probability that a given packet will be corrupted by noise, and reduce the time required to retransmit a corrupted packet.

During local operation, you can type Control-A at any point during the transfer to get a brief status report. You may also type Control-X or Control-Z to interrupt the current file or file group.

The RECEIVE Command

Syntax: `RECEIVE [`*filespec*`]`

The RECEIVE command tells Kermit-20 to receive a file or file group from the other system. If only one file is being received, you may include the optional *filespec* as the name to store the incoming file under; otherwise, the name is taken from the incoming file header. Even if the name in the header is not a legal TOPS-20 file name, Kermit-20 will store it under that name, in which case you can refer to it later only by quoting each illegal character (spaces, control characters, etc) with Control-V. If for some reason an incoming filename simply cannot be converted to legal form, the file will be saved as `-UNTRANSLATABLE-FILENAME-.KERMIT` (new generation). You may also use SET FILE NAMING NORMAL-FORM to have Kermit-20 choose more conventional names for incoming files.

If an incoming file has the same name as an existing file, Kermit-20 just creates a new generation of the same name and type, for instance `FOO.BAR.3, FOO.BAR.4`. The oldest generation will be automatically deleted, but you can still UNDELETE it.

Incoming files will all be stored with the prevailing bytesize, 7 by default, which is appropriate for text files. If you are asking Kermit-20 to receive binary files from a microcomputer or other 8-bit system, you must first type SET FILE BYTESIZE 8. Otherwise, the 8th bit of each byte will be lost and the file will be useless when sent back to the system of origin.

If you have SET PARITY, then 8th-bit prefixing will be requested. If the other side cannot do this, binary files cannot be transferred correctly. In all cases, Kermit-20 will request the other Kermit to compress repeated characters; if the other side can do this (not all Kermits know how) there may be a significant improvement in transmission speed.

If an incoming file does not arrive in its entirety, Kermit-20 will normally discard it; it will not appear in your directory. You may change this behavior by using the command SET INCOMPLETE KEEP, which will cause as much of the file as arrived to be saved in your directory.

If you are running Kermit-20 locally, you should already have issued a SEND command[4] to the remote Kermit, and then escaped back to DEC-20 Kermit. As files arrive, their names will be displayed on your screen, along with "." and "%" characters to indicate the packet traffic; you can type Control-A during the transfer for a brief status report.

If a file arrives that you don't really want, you can attempt to cancel it by typing Control-X; this sends a cancellation request to the remote Kermit. If the remote Kermit understands this request (not all implementations of Kermit support this feature), it will comply; otherwise it will continue to send. If a file group is being sent, you can request the entire group be cancelled by typing Control-Z.

The GET Command

Syntax: `GET [`*remote-filespec*`]`

The GET command requests a remote Kermit server to send the file or file group specified by *remote-filespec*. This command can be used only when there is a Kermit server on the other end of the line. This means that you must have CONNECTed to the other system, logged in, run Kermit there, issued the SERVER command, and escaped back to the DEC-20, or else you Kermit-20 is in remote mode, TAKEing commands from a file, and interacting with a local Kermit server.

The remote filespec is any string that can be a legal file specification for the remote system; it is not parsed or validated locally. You should not put a trailing comment on the GET command, since this will be sent as part of the remote filespec.

If you need to include otherwise illegal characters such as "!" or ";" (the normal command comment delimeters), "?" (the command help character), "@" (the indirect command file indicator), or certain control characters, then you should precede each such character by a Control-V. Kermit-20 will discard these Control-V quoting prefixes before sending the file specification to the remote host.

If you want to store the incoming file name with a different name than the remote host sends it with, just type GET alone on a line; Kermit-20 will prompt you separately for the source (remote) and destination (local) file specification. If more than one file arrives, only the first one will be stored under the name given; the rest will be stored under the names they are sent with. Example:

```
Kermit-20>get
 Remote Source File: profile exec a1
 Local Destination File: profile.exec
```

As files arrive, their names will be displayed on your screen, along with "." and "%" characters to indicate the packet traffic. As in the RECEIVE command, you may type Control-A to get a brief status report, ^X to request that the current incoming file be cancelled, ^Z to request that the entire incoming batch be cancelled.

If the remote Kermit is not capable of server functions, then you will probably get an error message back from it like "Illegal packet type". In this case, you must connect to the other Kermit, give a SEND command, escape back, and give a RECEIVE command.

---

[4] *not* SERVER -- use the GET command to receive files from a Kermit server.

The STATISTICS Command

Give statistics about the most recent file transfer. For instance, here's what Kermit-20 displayed after transmitting a short binary file, using repeated-character compression:

```
Maximum number of characters in packet:  80 received; 80 sent
Number of characters transmitted in 2 seconds
       Sent:      34          Overhead:      34
       Received:  107         Overhead:      -408
 Total received:  141         Overhead:      -374
Total characters transmitted per second:     70
Effective data rate:   2570 baud
Efficiency:            214.1667 per cent
Interpacket pause in effect: 0 sec

Timeouts: 0
NAKs:     0
```

Note that the data compression allowed the effective baud rate to exceed the actual speed of the communication line, which in this case happened to be 1200 baud. The efficiency is displayed only if the actual baud rate is known.

1.5.2. Server Operation

The SERVER Command

The SERVER command puts a remote Kermit-20 in "server mode", so that it receives all further commands in packets from the local Kermit. The Kermit-20 server is capable (as of this writing) of executing the following remote server commands: SEND, GET, FINISH, BYE, REMOTE DIRECTORY, REMOTE CWD, REMOTE SPACE, REMOTE DELETE, REMOTE TYPE, REMOTE HELP.

Any nonstandard parameters should be selected with SET commands before putting Kermit-20 into server mode, in particular the file bytesize. The DEC-20 Kermit server can send most files in the correct manner automatically, by recognizing the DEC-20 file bytesize. However, if you need to ask the DEC-20 Kermit server to receive binary files from an 8-bit-byte system (that is, from almost any system that's not a DEC-10 or DEC-20) you must issue the SET FILE BYTESIZE 8 command before putting it into server mode, and then you must only send 8-bit binary files. You cannot send a mixture of text files and 8-bit binary files to a Kermit-20 server.

Commands for Servers

When running in local mode, Kermit-20 allows you to give a wide range of commands to a remote Kermit server, with no guarantee the that the remote server can process them, since they are all optional features of the protocol. Commands for servers include the standard `SEND`, `GET`, `BYE`, and `FINISH` commands, as well as the `REMOTE` command.

These commands are generally issued when Kermit-20 is in local mode, i.e. you have already connected to another system, run Kermit there and put it into server mode, and escaped back to Kermit-20. However, Kermit-20 also allows you to operate in the opposite direction, i.e. Kermit-20 is the remote Kermit, and the local Kermit is in server mode. This is handy when, for instance, you want to transfer a disparate collection of files that can't be readily specified by a wildcard group, all in a single, unattended operation. In this case, you can create a TAKE command file for Kermit-20 that SENDs and/or GETs the desired files, and then shuts down local server when done, e.g.:

```
set delay 0             ; No need to pause before sending
; Connect to own directory, leave a blank line for password.
cwd me:

log transactions        ; Keep a log
```

```
; Change directories on the PC.
remote cwd \kermit

send ker:mskerm.doc     ; Send the MS-DOS Kermit manual
send ker:mskerm.bwr     ; Send the MS-DOS Kermit "beware file"
; Now change to the MS-DOS binaries area
remote cwd \bin

send kb:msvibm.exe      ; Send the executable DOS Kermit program
; Put DOS back in default directory
remote cwd \chris

; Connect back to default directory on the DEC-20
cwd me:

close transactions      ; Close transaction log
send transaction.log    ; Send it
finish                  ; Shut down DOS Kermit server
```

Commands to servers (GET, BYE, FINISH, REMOTE) can be issued from a remote Kermit-20 only by means of a TAKE file. When Kermit-20 is local (i.e. after SET LINE), you can issue these commands interactively as well.

The REMOTE Command

Send the specified command to the remote server. If the server does not understand the command (all of these commands are optional features of the Kermit protocol), it will reply with a message like "Unknown Kermit server command". If does understand, it will send the results back, and they will be displayed on the screen. The REMOTE commands are:

CWD [*directory*] Change Working Directory. If no directory name is provided, the server will change to the default or home directory. Otherwise, you will be prompted for a password, and the server will attempt to change to the specified directory. The password is entered on a separate line, and does not echo as you type it. If access is not granted, the server will provide a message to that effect. Do not put trailing comments after a REMOTE CWD command, or after the password.

DELETE *filespec* Delete the specified file or files. The names of the files that are deleted will appear on your screen.

DIRECTORY [*filespec*]
The names of the files that match the given file specification will be displayed on your screen, perhaps along with size and date information for each file. If no file specification is given, all files from the current directory will be listed.

HELP Provide a list of the functions that are available from the server.

HOST [*command*] Pass the given command to the server's host command processor, and display the resulting output on your screen.

SPACE Provide information about disk usage in the current directory, such as the quota, the current storage, the amount of remaining free space.

TYPE *filespec* Display the contents of the specified file on your screen.

1.5.3. Commands for Local File Management

Syntax: `LOCAL [`*command*`]`

Execute the specified command on the local system -- on the DEC-20 where Kermit-20 is running. These commands provide some local file management capability without having to leave the Kermit-20 program.

CWD [*directory*] Change working directory, or, in DEC-20 terminology, CONNECT to the specified directory. If a password is required, you will be prompted for one. Do not include a trailing comment after the password.

DELETE *filespec* Delete the specified file or files, but do not expunge them (unless you have SET EXPUNGE ON).

DIRECTORY [*filespec*]
Provide a directory listing of the specified files.

RUN [*filespec*] Attempts to run the specified file, which must be in ".EXE" format (.EXE is the default filetype), in an inferior fork. Control returns to Kermit-20 when the program terminates. Once you have used this command, you can restart the same program by issuing a RUN command with no arguments. If you RUN SYSTEM:EXEC, then you will be able to issue TOPS-20 commands without leaving Kermit; you can get back to Kermit from the EXEC by typing the EXEC POP command.

SPACE Show how much space is used and remaining in the current directory.

TYPE Display the contents of the specified file or files at your terminal. This works like the DEC-20 TYPE command, except that if a file has a bytesize of 8, Kermit-20 will do 8-bit input from it rather than 7-bit. Also, the DEC-20 Control-O command discards output only from the file currently being displayed; if multiple files are being typed, then output will resume with the next file.

The LOCAL commands may also be used without the "LOCAL" prefix.

1.5.4. The CONNECT Command

Syntax: CONNECT [*number*]

Establish a terminal connection to the system connected to the octal TTY number specified here or in the most recent SET LINE command, using full duplex echoing and no parity unless otherwise specified in previous SET commands. Get back to Kermit-20 by typing the escape character followed by the letter C. The escape character is Control-Backslash (^\) by default. When you type the escape character, several single-character commands are possible:

C Close the connection and return to Kermit-20.
S Show status of the connection; equivalent to SHOW LINE.
P Push to a new Exec. POP from the Exec to get back to the connection.
Q If a session log is active, temporarily Quit logging.
R Resume logging to the session log.
B Send a simulated BREAK signal.
? List all the possible single-character arguments.
^\ (or whatever you have set the escape character to be):
Typing the escape character twice sends one copy of it to the connected host.

You can use the SET ESCAPE command to define a different escape character, and SET PARITY, SET DUPLEX, SET HANDSHAKE, SET FLOW, and SET SPEED to change those communication-line-oriented parameters. In order for the simulated BREAK signal to work, TOPS-20 must know the speed of the terminal. If it does not, you may use the SET SPEED command. Type the SHOW LINE command for information about your current communication settings.

Kermit-20 does not have any special autodialer interface. It assumes that the connection has already been made and the line assigned.

1.5.5. The SET, SHOW, and DEFINE Commands

SET is used for establishing or changing parameters, DEFINE lets you group several SET commands together into a single "macro" command, and SHOW lets you examine current settings or macro definitions.

The SET Command

Syntax: `SET` *parameter* `[`*option* `[`*value*`]]`

Establish or modify various parameters for file transfer or terminal connection. You can examine their values with the SHOW command. The following parameters may be SET:

BREAK	Adjust the BREAK simulation parameter
BLOCK-CHECK	Packet transmission error detection method
DEBUGGING	Record or display state transitions or packets
DELAY	How long to wait before starting to send
DUPLEX	For terminal connection, FULL or HALF
ESCAPE	Character for terminal connection
FILE	For setting file parameters like byte size
FLOW-CONTROL	For enabling or disabling XON/XOFF flow control
HANDSHAKE	For turning around half duplex communication line
IBM	For communicating with an IBM mainframe
INCOMPLETE	What to do with an incomplete file
INPUT	For specifying behavior of the INPUT command
ITS-BINARY	For recognizing a special 8-bit binary file format
LINE	TTY line to use for file transfer or CONNECT
PARITY	Character parity to use
PROMPT	Change the program's command prompt
RECEIVE	Various parameters for receiving files
RETRY	How many times to retry a packet before giving up
SEND	Various parameters for sending files
SPEED	Baud rate of communication line
TVT-BINARY	For negotiating binary mode on ARPANET

The DEFINE command may be used to compose "macros" by combining SET commands. Those SET commands which differ from the "ideal" Kermit are now described in detail.

SET BREAK

Syntax: `SET BREAK` *n* Specify the number of nulls to be sent at 50 baud to simulate a BREAK signal when connected to a remote host via SET LINE and CONNECT.

SET DEBUG

Syntax: `SET DEBUG` *options*

Record the packet traffic, either on your terminal or in a file. Some reasons for doing this would be to debug a version of Kermit that you are working on, to record a transaction in which an error occurred for evidence when reporting bugs, or simply to vary the display you get when running Kermit-20 in local mode. Options are:

STATES
: Show Kermit state transitions and packet numbers (brief).

PACKETS
: Display each incoming and outgoing packet (lengthy).

OFF
: Don't display or record debugging information (this is the normal mode). If debugging was in effect, turn it off and close any log file.

The debugging information is recorded in the file specified by the most recent LOG DEBUGGING command, `DEBUGGING.LOG` by default.

SET ESCAPE

`SET ESCAPE` *octal-number*

Specify the control character you want to use to "escape" from remote connections back to Kermit-20. The default is 34 (Control-\). The number is the octal value of the ASCII control character, 1 to 37 (or 177), for instance 2 is Control-B. After you type the escape character, you must follow it by a one of the single-character "arguments" described under the CONNECT command, above.

SET EXPUNGE

`SET EXPUNGE ON` *or* `OFF`

Tell whether you want a DELETE command (either the LOCAL DELETE command or a REMOTE DELETE command sent to a Kermit-20 server) to expunge files as it deletes them. On the DEC-20, a deleted file continues to take up space, and may be "undeleted" at a later time in the same session. To expunge a deleted file means to remove it completely and irrevocably, freeing its space for further use. EXPUNGE is OFF by default; deleted files are not automatically expunged. SET EXPUNGE applies only to files that are deleted explicitly by Kermit-20, and not to files that are implicitly deleted when new generations of existing files are created.

SET FILE

Syntax: `SET FILE` *parameter keyword*

Establish file-related parameters:

BYTESIZE *keyword or number*
: Byte size for DEC-20 file input/output. The choices are SEVEN (7), EIGHT (8), and AUTO.

SEVEN
: (or 7) Always store or retrieve five 7-bit bytes per word. When sending a file, ignore the file bytesize and do 7-bit input from the file. There would be no reason to use this option except to explicitly force an 8-bit file to be treated as a 7-bit file.

EIGHT
: (or 8) Always store or retrieve four 8-bit bytes per word. When sending a file, ignore the file bytesize and do 8-bit input from the file. This command is necessary when receiving binary files from 8-bit-byte systems, such as most microcomputers.

AUTO
: Equivalent to SEVEN for incoming files, and for outgoing files means to use EIGHT if the DEC-20 file bytesize (as shown by the Exec VDIR command) is 8, otherwise use SEVEN. The default is AUTO.

The DEC-20 can send any mixture of file types in the correct way automatically, but you *must* set the file bytesize to 8 for any incoming 8-bit binary files, and to AUTO (i.e. 7) for any incoming text files or DEC-20 binary files.

NAMING UNTRANSLATED *or* NORMAL-FORM
: If NORMAL-FORM the names of incoming or outgoing files will be converted to contain only uppercase letters, digits, and at most one period; any other characters will be translated to "`X`". If UNTRANSLATED, filenames will be sent and used literally. UNTRANSLATED is the default.

SET IBM

Syntax: `SET IBM ON` *or* `OFF`

SET IBM is really a predefined SET macro rather than a "hardwired" SET command; it can be redefined or undefined (see DEFINE); as distributed from Columbia, Kermit-20 defines IBM to be "parity mark, handshake XON, duplex half".

SET IBM should be used when running Kermit-20 in local mode, connected to an IBM or similar mainframe. If you have redefined the SET IBM macro, then your parameters will be used instead.

SET ITS-BINARY

Syntax: `SET ITS-BINARY ON` *or* `OFF`

Specify whether ITS-Binary file headers are to be recognized or ignored. By default, they are recognized. ITS binary format is a way (devised at MIT) of storing foreign 8-bit binary data on a 36-bit machine to allow automatic recognition of these files when sending them out again, so that you don't have to depend on the file byte size, or to issue explicit SET FILE BYTESIZE commands to Kermit.

An ITS format binary file contains the sixbit characters "DSK8" left-adjusted in the first 36-bit word. If ITS-BINARY is ON, then Kermit-20 will send any file starting with this "header word" using 8-bit input from the file even if the file bytesize is not 8, and will not send the header word itself. Kermit-20 will also store any incoming file that begins with that header word using 8-bit bytesize, again discarding the header word itself. If ITS-BINARY is OFF, then the header word, if any, will be sent or kept, and i/o will be according to the setting of FILE BYTESIZE.

This facility is provided for compatibility with the file formats used on certain public-access CP/M libraries.

SET INPUT

Syntax: `SET INPUT` *parameter value*

The INPUT command is used in TAKE command files or DEC-20 Batch control files as part of the login script facility, which is explained in greater detail later. SET INPUT controls the behavior of the INPUT command. The parameters are as follows:

SET INPUT DEFAULT-TIMEOUT *n*

n is the number of seconds for an INPUT command to time out after not receiving the requested input, when no interval is explicitly given in the INPUT command. For instance, if the default timeout interval is 10 seconds, then the command

```
INPUT login:
```

will look for the "`login:`" prompt for 10 seconds. The default may be overriden by including an explicit interval in the INPUT command:

```
INPUT 15 login:
```

The default timeout interval is 5 seconds.

SET INPUT TIMEOUT-ACTION PROCEED *or* QUIT

If the INPUT command comes from a Kermit-20 command file (see TAKE command) or a TOPS-20 Batch control file, then use this command to specify whether processing of the command file should proceed or quit after a timeout occurs. For TAKE files, the current command file is terminated and control returns to the invoking level (Kermit-20 prompt level, or a superior TAKE file). The default action is PROCEED.

SET INPUT CASE IGNORE *or* OBSERVE

Specify whether alphabetic case should be ignored ("a" matches "A") or observed ("a" does not match "A") when scanning the input for the specified search string. By default, aphabetic case is ignored.

SET INPUT commands are "global"; the settings are not "pushed" and "popped" when entering or leaving TAKE command files.

SET LINE

Syntax: `SET LINE [`*octal-number*`]`

Specify the octal TTY number to use for file transfer or CONNECT. If you issue this command, you will be running Kermit-20 *locally*, and you must log in to the remote system and run Kermit on that side in order to transfer a file. If you don't issue this command, Kermit-20 assumes it is running *remotely*, and does file transfer over its job's controlling terminal line. You can also select the line directly in the CONNECT command; the command

```
CONNECT 12
```

is equivalent to

```
SET LINE 12
CONNECT
```

If you type SET LINE with no number argument, you will deassign any previous assigned line and revert to remote mode.

The SHOW LINE command will display the currently selected communication line and its charactistics, including parity, duplex, handshake, flow control, the speed if known, whether carrier is present (if it is a modem-controlled line), and whether Kermit-20 is in local or remote mode.

SET RECEIVE

In addition to the full complement of SET RECEIVE commands described in the main part of the Kermit User Guide, you may also SET RECEIVE SERVER-TIMEOUT to a value between 0 and 94. This specifies the number of seconds between timeouts during server command wait, 0 specifies that no timeouts should occur during server command wait. When a Kermit server times out, it sends a NAK packet. Some systems cannot clear piled-up NAKs from their input buffers; if you're using such a system to communicate with a Kermit-20 server, and you expect to be leaving the server idle for long periods of time, you should use this command to turn off server command-wait timeouts.

SET SPEED

Syntax: `SET SPEED` *n*

Set the baud rate of the currently selected communication to *n*, the decimal baud rate, for instance 300, 1200, 4800. When operating in local mode, it may be necessary to issue this command in order to enable BREAK simulation.

SET TVT-BINARY

Syntax: `SET TVT-BINARY ON` *or* `OFF`

Only for users running Kermit-20 on an ARPANET DEC-20, signed on to an ARPANET virtual terminal (TVT) from another host or through an ARPANET TAC. SET TVT ON causes Kermit-20 to negotiate binary mode (8-bit) communication with the ARPANET during file transfer. Without this command, file transfer through a TVT would not work in most cases.

TVT-BINARY is OFF by default. If you normally use Kermit-20 through the ARPAnet, you can put the command SET TVT-BINARY ON into your `KERMIT.INI` file.

CAUTION: This facility requires certain features in the Release 5 TOPS-20 ARPANET monitor, which may not be present in releases distributed by DEC. See the Kermit-20 source code for details.

The DEFINE Command

Syntax: `DEFINE` *macroname* `[`*set-option* `[,` *set-option* `[...]]]`

The DEFINE command is available in Kermit-20 for building "macros" of SET commands. The macro name can be any keyword-style character string, and the set options are anything you would type after SET in a SET command; several set options may be strung together, separated by commas. Example:

```
define notimeout send timeout 0, receive timeout 0, receive server 0
```

Macro definitions may not include macro names. You can list all your macros and their definitions with the SHOW MACROS command. You can list a particular macro definition with HELP SET *macroname*.

The SHOW Command

Syntax: `SHOW` [*option*]

The SHOW command displays various information:

DAYTIME	Current date, time, phase of moon.
DEBUGGING	Debugging mode in effect, if any.
FILE-INFO	Byte size for DEC-20 file i/o, incomplete file disposition.
INPUT	INPUT command parameters.
LINE	TTY line, parity, duplex, flow control, handshake, escape character, speed (if known), and session loggin information. Note that before release 6.0 of TOPS-20, the DEC-20 does not keep a record of the actual baud rate of a modem-controlled or "remote" TTY line.
MACROS	Definitions for SET macros.
PACKET-INFO	For incoming and outbound packets. Items under RECEIVE column show parameters for packets Kermit-20 expects to receive, under SEND shows parameters for outgoing packets.
TIMING-INFO	Delays, retries, server NAK intervals.
VERSION	Program version of Kermit-20. This is also displayed when Kermit-20 is initially started.
ALL	(default) All of the above.

1.5.6. Program Management Commands

The TAKE Command

Syntax: `TAKE` *filespec*

Execute Kermit-20 commands from the specified file. The file may contain contain any valid Kermit-20 commands, including other TAKE commands; command files may be nested up to a depth of 20. Default file type for the command file is `.CMD`. Most commands may have trailing comments, beginning by semicolon, but these should be avoided in REMOTE commands, GET commands, and the passwords that are prompted for after CWD and REMOTE CWD commands.

The ECHO Command

Syntax: `ECHO` *line of text*

The line of text is echoed at the terminal. This is useful when issued from within TAKE command files, to report progress or issue instructions.

The HELP Command

Syntax: `HELP [`*topic* `[`*subtopic*`]]`

Typing HELP alone prints a brief summary of Kermit-20 and its commands. You can also type

`HELP` *command*

for any Kermit-20 command, e.g. "help send" or "help set parity" to get more detailed information about a specific command. Type

`HELP ?`

to see a list of the available help commands.

The EXIT and QUIT Commands

Syntax: `EXIT`

Exit from Kermit-20. You can CONTINUE the program from the TOPS-20 Exec, provided you haven't run another program on top of it. You can also exit from Kermit-20 by typing one or more control-C's, even if it's in the middle of transferring a file. Kermit-20 will always restore your terminal to its original condition, and you will be able to CONTINUE the program to get back to "`Kermit-20>`" command level with current settings intact.

`QUIT` is a synonym for EXIT.

The LOG Command

Syntax: `LOG` [*option* [*filespec*]]

Log the specified option to the specified file:

SESSION During CONNECT or execution of a login script, log all characters that appear on the screen to the specified file. During CONNECT, the session log can be temporarily turned off during the remote session by typing the escape character followed by Q (for Quit logging), and turned on again by typing the escape character followed by R (for Resume logging). Default log is `SESSION.LOG` in the current directory.

TRANSACTIONS During file transfer, log the progress of each file. The DEC-20 transaction log file looks like this:

```
Kermit-20 Transaction Log File, Monday 27-Feb-1984

18:40:13: Opened Log: PS:<TIMREK>SAMPLE.LOG.1
18:40:31: -- Send Begins --
   8th bit prefixing: Off
   Block check type: 1
18:40:31: Opened File: PS:<SY.FDC>LOGIN.CMD.6
   Sending As "LOGIN.CMD"
   Sent: 547 7-bit bytes
18:40:34: Closed PS:<SY.FDC>LOGIN.CMD.6
```

```
18:40:34: Send Complete
18:40:50: -- Receive Begins --
   8th bit prefixing: Off
   Block check type: 1
18:40:50: Opened: PS:<TIMREK>AUTOEXEC.BAT.1
   Written: 186 7-bit bytes
18:40:51: Closed: PS:<TIMREK>AUTOEXEC.BAT.1
18:40:56: Closed Transaction Log
```

Transaction logging is recommended for long or unattended file transfers, so that you don't have to watch the screen. The log may be inspected after the transfer is complete to see what files were transferred and what errors may have occurred. Default log is `TRANSACTION.LOG` in the current directory.

DEBUGGING Log STATES or PACKETS, as specified in the most recent SET DEBUGGING command, to the specified file. If log file not specified, then use TTY if local, or `DEBUGGING.LOG` in the current directory if remote. If no SET DEBUGGING command was previously issued, log STATES to the specified file. Also allow specification of bytesize for the log file, 7 (normal, default), or 8 (for debugging binary transfers when the parity bit is being used for data), for instance

```
LOG DEBUGGING BINARY.LOG 8
```

A 7-bit log file can be typed, printed, or examined with a text editor or searching program. An 8-bit log file can only be examined with a system utility like FILDDT. When logging packets, each packet is preceded by a timestamp, the current timeout interval (preceded by a slash), and "`R:`" or "`S:`" to indicate data being received and sent, respectively. Packet format is described in the *Kermit Protocol Manual*.

SESSION is the default option. Thus the command "LOG" alone will cause CONNECT sessions to be logged in `SESSION.LOG` in the current directory. Any log files are closed when you EXIT or QUIT from Kermit, and are reactivated if you CONTINUE the program. You may explicitly close a log file and terminate logging with the CLOSE command.

The CLOSE Command

Syntax: `CLOSE` *option*

Close the specified log file, SESSION, TRANSACTION, or DEBUGGING, and terminate logging to that file.

1.6. Login Scripts: The INPUT, OUTPUT, CLEAR, and PAUSE Commands

When running Kermit-20 in local mode, connecting from the DEC-20 to another system via an external TTY line (for instance, through an autodialer), you may use the Kermit-20 INPUT, OUTPUT, CLEAR, and PAUSE commands to carry on a dialog with the remote system. When combined into a "script" in a Kermit-20 TAKE command file, or included in a Batch control file, these commands provide the ability to initially connect and log in to a remote system, and to set it up for file transfer. During script execution, session logging may be used to record the dialog.

The CLEAR Command

Syntax: `CLEAR`

Clear the input and output buffers of the currently selected line, and attempt to clear any XOFF deadlock.

The PAUSE Command

Syntax: `PAUSE` [*interval*]

Pause the specified number of seconds before executing the next command. The default interval is one second.

The INPUT Command

Syntax: `INPUT` [*interval*] [*string*]

On the currently selected communication line, look for the given string for the specified interval of time, which is specified in seconds. If no interval is specified, then wait for the default interval, which may be specified by SET INPUT DEFAULT-TIMEOUT, and is normally 5 seconds. Specifying an interval of 0 (or less) means no timeout -- wait forever for the specified string. An INPUT command can by interrupted by typing one or more Control-C's, which will return you to `Kermit-20>` prompt level.

Characters coming in from the line will be scanned for the search string, and when a match is found, the command will terminate successfully; if the string is not found within the given interval, the command will terminate unsuccessfully. While the INPUT command is active, all incoming characters will appear on your screen.

The search string may contain any printable characters. Control or other special characters that you could not normally type as part of a command may be included by preceding their octal ASCII values with a backslash, for instance `foo\15` is "foo" followed by a carriage return (ASCII 15, octal). A backslash alone will be taken as is, unless it is followed by an octal digit (0-7); if you want to actually specify a backslash in this context, double the backslash (`\\5` will be taken as `\5`).

The behavior of the INPUT command is governed by the SET INPUT CASE, SET INPUT DEFAULT-TIMEOUT, and SET INPUT TIMEOUT-ACTION commands, as described in the Kermit Commands section of the User Guide, or in the Kermit book.

In addition to normal use, Kermit-20 scripts can also be used in DEC-20 batch control files. Failure to match an input string in the timeout interval will result in a message starting with ‘‘`?`’’, which signals the Batch controller to detect an error. If INPUT TIMEOUT-ACTION is SET to PROCEED, any timeout error messages will be issued starting with a "`%`", which does not signal an error to Batch.

In addition to otherwise untypable control characters (like Control-C), certain printable characters in the search string may need to be "quoted" using the backslash mechanism:

`@` (ASCII 100) If it is the first character in the string, atsign tells TOPS-20 that the following characters will be the name of an indirect command file, for instance

```
input 10 @foo.txt
```

tells Kermit to spend 10 seconds scanning the communication line input for the string which is contained in the file `FOO.TXT`. If you need to specify a string that starts with "`@`", use `\100` instead.

`?` (ASCII 77) A question mark tells TOPS-20 to provide a brief help message about this part of the command; use `\77` instead.

`!` (ASCII 41) If it is the first character in the string, an exclamation point will cause TOPS-20 to ignore the rest of the string, i.e. treat it as a comment, use `\41`.

`;` (ASCII 73) Same as exclamation mark, use `\73`.

`(` (ASCII 50) In first position, TOPS-20 will think this marks the beginning of a "guide word"; use `\50`.

`-` (ASCII 55) In *final* position, a dash tells TOPS-20 that the command line is to be continued, concatenated with the following line. Use `\55` instead of a final dash. For instance, to specify the string "`More?--`", use "`More\77-\55`".

The OUTPUT Command

Syntax: `OUTPUT` *string*

The given string is sent out the currently selected communication line. The string is in the same form as the INPUT string; control or special characters may be included by prefacing their octal ASCII value with a backslash. Note that any terminating carriage return must be included explicitly as `\15`. The string will also be echoed at your terminal.

Login Script Hints

It is not a good idea to store passwords in plain text in a file. The facilities of the TOPS-20 command parser make this unnecessary, so long as you are sitting at your terminal. Suppose you have a script that looks for the string "`Password: `" and then outputs your password using a command like

```
out mypassword\15
```

If you change this line to

```
out @tty:
```

you may enter the password from your terminal as follows:

```
login: myuserid
Password: mypassword\15^Z
```

That is, you type the password, a backslash-encoded carriage return, and then Control-Z. This may be done even when executing commands from a TAKE file; after the `^Z`, control returns to the TAKE file. In the OUTPUT command, "`@TTY:` designates `TTY:` (your job's controlling terminal) to be an indirect command file; the `^Z` is the "end of file" for a terminal. This same technique could have been used in the first script example to allow you to supply from the terminal the name of the file to be sent. It might be a good idea to for you to include an ECHO command in your script file to remind you to do this, for instance:

```
input password:
echo ^GType your password, followed by "\15" and then a CTRL-Z
output @tty:
```

The `^G` is a Control-G, which should get your attention by sounding a beep at your terminal.

One might expect to be able to use the same indirect file mechanism with the OUTPUT command to provide a crude "raw upload" facility, as in

```
output @foo.bar
```

to send the contents of the file `FOO.BAR` to the remote system, with *no* synchronization or error checking. Unfortunately, there are two problems with this approach: first, TOPS-20 converts all carriage return / linefeeds in an indirect command file to spaces, and second, only very short files may be treated this way, because they must fit within TOPS-20's command "atom" buffer. The Kermit-20 TRANSMIT command provides a synchronized raw uploading of files.

1.7. Raw Download and Upload

"Raw Download" is the term commonly used to describe the capture of a remote file on the local system, without any kind of error detection or correction. This allows you to obtain files from remote systems that do not have Kermit, but with the risk of loss or corruption of data.

Kermit-20 provides raw downloading via the LOG SESSION command during CONNECT to a remote system. The session log is described above. To use session logging to capture a file:

1. Run Kermit on the DEC-20.

2. SET LINE to the TTY number through which you will be connected to the remote system.

3. Perform any required SET commands to condition Kermit for communication with the remote system. You may need SET PARITY, SET DUPLEX, SET FLOW, SET HANDSHAKE, etc., depending on the characteristics of the remote system and the communication medium.

4. CONNECT to the remote system and log in.

5. Condition your job on the remote system not to pause at the end of a screenful of text, and give whatever commands may be necessary to achieve a "clean" terminal listing -- for instance, disable messages from the system or other users.

6. Type the appropriate command to have the desired file displayed at the terminal, *but do not type the terminating carriage return*. On most systems, the command would be "type", on Unix it's "cat".

7. Escape back to Kermit to the DEC-20 and give the LOG SESSION command.

8. CONNECT back to the remote system and type a carriage return. The file will be displayed on your screen and recorded in the session log file.

9. Escape back to Kermit on the DEC-20 and give the CLOSE SESSION command.

The file will be in `SESSION.LOG` in your connected directory, unless you gave another name for it in your LOG SESSION command. You will probably find that some editing necessary to remove extraneous prompts, messages, padding characters, or terminal escape sequences, or to fill in lost or garbled characters. Here's an example showing how to capture a file `foo.bar` from a remote Unix system:

```
@kermit
Kermit-20>set line 23
Kermit-20>connect
[KERMIT-20: Connecting to remote host over TTY23:,
 type <CTRL-\>C to return.]
4.2 BSD UNIX

login: myuserid
Password: mypassword
% cat foo.bar^\C
[KERMIT-20: Connection Closed]
Kermit-20>log session foo.bar
Kermit-20>connect
[KERMIT-20: Connecting to remote host over TTY23:,
 type <CTRL-\>C to return.]
[KERMIT-20: Logging to File FOO.BAR.1]
(Type carriage return now.)
This is the file foo.bar.
It has three lines.
This is the last line.
% ^\
[KERMIT-20: Closing Log File FOO.BAR.1>
[KERMIT-20: Connection Closed]
```

```
Kermit-20>close session
```

Note that in this case, the Unix "`% `" prompt at the end of the text will have to be edited out.

Raw Upload

"Raw Upload" means sending a file from the local system to a remote one, again without error detection or correction. This allows you to send files from the DEC-20 to remote systems that don't have Kermit. Kermit-20 provides the TRANSMIT command for this purpose.

Syntax: `TRANSMIT` *filespec* [*prompt*]

For use in local mode only. Sends the specified text file a line at a time, "raw" (as is, *without* using Kermit protocol), to the remote system, waiting for the specified prompt for each line. Only a single file may be sent with the TRANSMIT command; wildcards are not allowed in the filespec. The file should be a text file, not a binary file. Since protocol is not being used, no assurance can be given that the file will arrive at the destination correctly or completely.

The *prompt* is any string, for instance the prompt of a line editor in text insertion mode. The prompt string may include special characters by preceding their octal ASCII values with a backslash, e.g. `\12` for linefeed, `\21` for XON (`^Q`). The syntax of the prompt string is explained in greater detail above, with the INPUT command.

If a prompt string is supplied, alphabetic case will be ignored in searching for it unless you SET INPUT CASE OBSERVE. If a prompt string is not supplied, then linefeed will be used by default, unless you have performed a SET HANDSHAKE command, in which case the current handshake character will be used. If you really want to send the entire file without waiting for any prompts, specify a prompt of "\0" (ASCII zero, null) (this is not advised).

The file will be sent using the current settings for duplex, parity, and flow control. There are no timeouts on input, as there are with the INPUT command. The TRANSMIT command waits forever for the prompt to appear. However, if you observe that the transfer is stuck, there are three things you can do:

- Type a Carriage Return to transmit the next line.
- Type a Control-P to retransmit the line that was just transmitted.
- Type two Control-C's to cancel the TRANSMIT command and get back to `Kermit-20>` command level.

TRANSMIT should be used as follows: CONNECT to the remote system, login, and start up some kind of process on the remote system to store input from the terminal into a file. On a DEC-20 (that doesn't have Kermit), you could do

```
copy tty: foo.bar
```

or you could start a line editor like EDIT or OTTO and put it into text insertion mode. On a Unix system, you could

```
cat /dev/tty > foo.bar
```

or you could run "ed" and give it the "a" command.

The Kermit-20 TRANSMIT command will send the first line of the file immediately. Then it will wait for a "prompt" from the remote system before sending the next line. When performing a copy operation from the terminal to a file, the "prompt" will probably be a linefeed, "`\12`" which is the default prompt -- most full duplex systems expect you to type a line of text terminated by a carriage return; they echo the characters you type and then output a linefeed. Half duplex systems, on the other hand, use some kind of line turnaround handshake character, like XON (Control-Q), to let you know when they are ready for the next line of input. Line editors like EDIT and OTTO prompt you with a line number followed by a tab; in that case your prompt character would be "`\11`" (be careful -- if the remote DEC-20 doesn't think your terminal has hardware tabs, it will simulate them by outputting spaces). In any case, to assure synchronization, it is your responsibility to set up the target system to accept line-at-a-time textual input and to determine what the system's prompt will be when it is ready for the next line.

Each line is sent with a terminating carriage return; linefeeds are not sent, since these are supplied by the receiving system if it needs them. The TRANSMIT command continues to send all the lines of the file in this manner until it reaches the end, or until you interrupt the operation by typing Control-C's.

If you cannot make the TRANSMIT command work automatically, for instance because the remote system's prompt changes for each line, you may TRANSMIT manually by specifying a prompt string that will not appear, and then typing a carriage return at your keyboard for each line you want to send.

If the TRANSMIT command completes successfully (you'll get a message to the effect that the transmission is complete), then you must connect back to the remote system and type whatever command it needs in orderto save and/or close the file there.

1.8. Kermit-20 Examples

Here are a few examples of the use of Kermit-20. Text entered by the user is underlined.

Remote Operation

The following example shows use of Kermit-20 as a server from an IBM PC. In this example, the user runs Kermit on the PC, connects to the DEC-20, and starts Kermit-20 in server mode. From that point on, the user need never connect to the DEC-20 again. In this example, the user gets a file from the DEC-20, works on it locally at the PC, and then sends the results back to the DEC-20. Note that the user can leave and restart Kermit on the PC as often as desired.

```
A>Kermit
Kermit-MS>connect
@
@Kermit
TOPS-20 Kermit version 4.2(262)

Kermit-20>server

 Kermit Server running on DEC-20 host.  Please type your escape
 sequence to return to your local machine.  Shut down the server by
 typing the Kermit BYE command on your local machine.
^[C
Kermit-MS>get foo.txt
      The transfer takes place.

Kermit-MS>exit
A>
A>edit foo.txt ; (or whatever...)
A>
A>Kermit
Kermit-MS>send foo.txt
      The transfer takes place.

Kermit-MS>bye
A>
```

The next example shows the special procedure you would have to use in order to send a mixture of text and binary files from a PC (or an 8-bit-byte system) to the DEC-20. Note that in this case, it's more convenient to avoid server mode.

```
Kermit-MS>connect
```

```
@
@Kermit
TOPS-20 Kermit version 4.2(262)

Kermit-20>receive
^]C
Kermit-MS>send *.txt
```

Textual files are sent.

```
Kermit-MS>connect
Kermit-20>set file bytesize 8
Kermit-20>receive
^]C
Kermit-MS>send *.exe
```

Binary files are sent.

```
Kermit-MS>connect
Kermit-20>exit
@logout
^]C
Kermit-86>exit
A>
```

Local Operation

In this example, a program DIAL is used to direct an autodialer to call another computer (a DECsystem-10); once the connection is made, DIAL starts Kermit with an implicit CONNECT command for the appropriate communication line. DIAL is not part of Kermit; if your system has an autodialer, there will be some site-specific procedure for using it.

```
@dial
Dial>dial stevens
STEVENS, 1-(201) 555-1234, baud:1200
[confirm]
Dialing your number, please hold...
Your party is waiting on TTY11:.
@
@Kermit
TOPS-20 Kermit version 4.2(262)

Kermit-20>connect 11
[Kermit-20: Connecting over TTY11:, type <CTRL-\>C to return]

CONNECTING TO HOST SYSTEM.
Stevens T/S 7.01A(10) 20:20:04 TTY41 system 1282
Connected to Node DN87S1(101) Line # 57

Please LOGIN or ATTACH

.log 10,35
JOB 51 Stevens T/S 7.01A(10) TTY41
Password:
20:20   15-Dec-83       Thur

.r new:Kermit

TOPS-10 Kermit version 2(106)

Kermit-10>server

[Kermit Server running on the DEC host.  Please type your escape
 sequence to return to your local machine.  Shut down the server by
 typing the Kermit BYE command on your local machine.]
^YC

[Kermit-20: Connection Closed.  Back at DEC-20.]
```

```
Kermit-20>set file bytesize 8
Kermit-20>get setdtr.cmd
^A for status report, ^X to cancel file, ^Z to cancel batch.
SETDTR.CMD.7 ^A
 Receiving SETDTR.CMD.7, file bytesize 8
 (repeated character compression)
 At page 1
 Files: 0, packets: 1, chars: 66
 NAKs: 0, timeouts: 0
.[OK]
Kermit-20>bye
Job 51  User F DA CRUZ [10,35]
Logged-off TTY41  at 20:22:58  on 15-Dec-83
Runtime: 0:00:01, KCS:33, Connect time: 0:02:39
Disk Reads:72, Writes:4, Blocks saved:160
....
Hangup? y
Click. Call duration was 193 seconds to area 201.
Dial>exit
```

Note the use of Control-A to get a status report during the transfer.

1.9. Installation of Kermit-20

Kermit-20 is built from a single MACRO-20 source file, `K20MIT.MAC`. It requires the standard DEC-distributed tools MONSYM, MACSYM, and CMD; the following files should be in `SYS:` -- `MONSYM.UNV`, `MACSYM.UNV`, `MACREL.REL`, `CMD.UNV`, and `CMD.REL`. The CMD package is also included with the Kermit distribution as `K20CMD.*`, in case you can't find it on your system.

The program should work on all TOPS-20 systems as distributed, but many customizations are possible. The site manager may wish to change various default parameters on a site-wide basis; this may be done simply by changing the definitions of the desired symbols, under "subttl Definitions", and reassembling.

The most notable site dependency is the definition of "SET IBM". As distributed from Columbia, Kermit-20 defines "SET IBM" in a built-in SET macro definition as "parity mark, duplex half, handshake xon". This definition may be found at MACTAB+1, near the end of the impure data section. It may be changed or deleted, and the program reassembled.

Sites that do not have ARPANET may wish to delete the TVT-BINARY entries from SET command tables, SETABL and SETHLP.

Index

Table of Contents

List of Figures