                        Internet Kermit Service

Status of this Memo

Copyright Notice

ABSTRACT

   This document describes a new file transfer service for the Internet
   based on Telnet Protocol for option negotiation and Kermit Protocol
   for file transfer and management. The Internet Kermit Service
   provides access to both authenticated and anonymous users. The use
   of Kermit protocol over a Telnet connection provides several
   advantages over FTP, including easy traversal of firewalls, transfers
   over multiple transports, and security via a combination of supported
   Telnet authentication and encryption option negotiations, plus
   significant functional benefits. While this document describes a new
   service for the Internet, the clients for this service already exist
   on most platforms in the form of Telnet clients that support the
   Kermit file transfer protocol. These clients are available not only
   from Columbia University's Kermit Project but also numerous third
   parties.

TABLE OF CONTENTS

PREFACE

   This document describes an Internet Kermit Service (IKS) which
   provides an alternative to FTP for the transfer of files. This
   service is based upon both the TELNET protocol and the Kermit file
   transfer protocol.

1. INTRODUCTION

   The Internet Kermit Service:

   1. Provides direct access to Kermit file transfer and management
      services without requiring the user to first login to a shell
      account;

   2. Provides Kermit file transfer and management services to anonymous
      users;

   3. Provides services to all Telnet clients that support Kermit file
      transfer protocol via a simple, predictable, scriptable, and
      well-documented textual interface;

   4. Provides direct and tightly-coupled access to a Kermit server when
      requested via the Telnet Kermit Option [TKO].

   This memo assumes knowledge of Transmission Control Protocol, the
   Telnet Protocol [TEL], the Kermit File Transfer Protocol [KER,PRF],
   Telnet Kermit Option [TKO], and the commands and features of Kermit
   software [CKB,CMG,K95].

   The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT",
   "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this
   document are to be interpreted as described in RFC 2119 [BCP].

   Definitions:

   Kermit server
      A software program that is ready to accept and act upon commands
      in the form of well-defined Kermit packets [KER].

   Kermit client
      A software program that receives requests through its user
      interface from a human user (or a script or other source) and
      translates them to command packets, which it sends to a Kermit
      server, thus initiating a Kermit protocol transaction such as the
      transfer of one or more files.

2. BACKGROUND

2.1. History

   "Kermit" is the name of an extensible platform- and medium-
   independent file transfer and management protocol [KER,PRF] and of a
   suite of communications software programs that implement it and
   integrate it with other communications functions [CMG,CKB,K95].

   The Kermit protocol was first developed at Columbia University in New
   York City in 1981 for transferring files without errors between
   diverse types of computers over potentially hostile communication
   links. Since 1981, the Kermit Project at Columbia University has
   expanded the protocol, developed communications software that
   implements it upon key platforms, and worked with volunteer
   programmers at other sites adapting Kermit protocol to other
   platforms or communication methods. The Kermit Project also serves
   as the central point of Kermit software development, support,
   information, and distribution throughout the world.

   Kermit software is now available for nearly every computer and
   operating system in existence. The major features of the most
   popular Kermit programs are:

   - Connection establishment and maintenance for a variety of
     connection methods including direct serial, dialup, TCP/IP, X.25,
     DECnet, and NETBIOS.

   - Terminal emulation.

   - Error-free transfer of both text and binary files, individually or
     in groups.

   - Character-set translation during both terminal emulation and
     text-mode file transfer -- a unique feature of Kermit software.

   - Remote file management through the client/server protocol.

   - A powerful and portable scripting language allowing complete
      automation of any task that can be performed manually.

   Kermit's command and script language is consistent across all
   platforms and communication methods, thus offering a unified method
   for accomplishing a wide range of communication tasks manually or
   under script control.

   A single Kermit program combines the functions of many different
   programs such as uucp, cu, tip, telnet, rlogin, ftp, iconv, and
   expect: it is a Telnet and Rlogin client that can also transfer
   files; it is a file transfer program that can also convert character
   sets; it is a dialout program that can use dialing directories and
   understands country codes and area codes; it is fully scriptable; it
   offers both client/server and interactive modes of operation. In its
   desktop versions (particularly for DOS, Windows, and OS/2) it offers
   all the features of communications software that are usually lacking
   from Internet client software (key mapping, colors, scrollback, mouse
   functions, printer control, etc)

   Kermit software is widely used throughout the academic, government,
   and corporate spheres, both in the USA and internationally.

   In addition to the Kermit software developed and/or distributed by
   the Kermit Project at Columbia University, hundreds of other software
   products -- commercial, shareware, and freeware -- also include some
   level of support for the Kermit protocol. Thus there are hundreds,
   perhaps thousands, of independent and interoperable Kermit protocol
   implementations based upon the open Kermit protocol specification
   [KER].

   The Internet has formed the primary mechanism by which users and
   developers of Kermit software have collaborated to produce feature
   and command sets that continually evolve to meet their needs as
   technology changes.

2.2. Motivation.

   Kermit protocol and software makes connections from one computer to
   another and transfers data between them. Countless people "live" in
   Kermit all day long; as a customizable Telnet or Rlogin (or serial
   communication) client with a wide selection of terminal emulations
   and convenience features, it is their window onto the Internet.

Others use it in more creative ways, including some that involve key
parts of the Internet, e.g. in batch or cron jobs that update news or
Web servers or fetch email, or to monitor routers, terminal servers,
and hubs and dial pagers when faults are detected. It is used by
vendors of telecommunications equipment for remote diagnosis,
patching, and updates. Telecom managers often use Kermit scripts to
configure PBXs, muxes, routers, or terminal servers. In the world of
commerce, Kermit is widely used for financial transactions, EDI,
medical claim submission, and so forth. It is used with mobile
barcode readers in warehousing and inventory applications. It is
found in US Postal Service sorting and scanning equipment. It
connects many of the logistics and supply systems throughout the
military. It is found in fast-food restaurant cash registers,
milling and die-cutting machines, textile looms and cutters, printing
presses, and medical diagnostic equipment. It was the communications
backbone of the 1994 Brazilian national election -- the largest in
history.

And yet there has never been a strong, explicit connection of Kermit
with the Internet. In the early years, Kermit acted as a kind of
do-it-yourself network, enabling ordinary users to make connections
that were not already there, and for some years was the predominant
method of connecting a personal computer to the ARPAnet (e.g. by
dialing a TAC).

Nowadays, however, with so many of the world's computers on the
Internet, the role of Kermit software and protocol is changing.
Kermit users on the network would like to have the features,
functions, and interface they are accustomed to -- especially the
automation features -- available for use in settings where presently
only tools like FTP are available -- and even more so in situations
where standard software like FTP can't be used.

An Internet Kermit Service can fill this role, and augment the data
transfer power and flexibility of other Internet applications such as
Web browsers:

- Like FTP, Kermit provides a service that can be accessed from many
   different platforms with a consistent set of commands, but unlike
   FTP, these commands include programming constructions such as
   variables, arrays, looping and selection mechanisms, and local and
   remote procedure calls.

- Like FTP, Kermit provides both text- and binary-mode data
   transfer, as well as file management capabilities. But Kermit
   also offers numerous features lacking from FTP, such as

          character-set translation, flexible file selection mechanisms,
          attribute preservation, and so on (see Section 5.3 for a longer
          list).

     -    Unlike standard FTP, Kermit can transfer data through multiple
          firewalls, proxies, and network address translators (NATs) on a
          single port.

     -    Unlike FTP, Kermit can transfer data across a combination of
          transports (e.g. dial-up to a terminal server and thence to an
          Internet host).

     -    Authentication and data transfer can take place over secure
          connections (mutually authenticated and encrypted) using
          established Telnet authentication and encryption options.

     -    Unlike traditional Kermit use over Telnet, anonymous access is
          possible, and the considerable overhead of the intervening Telnet
          server and pseudoterminal service is eliminated.

     Until now the primary obstacles to an Internet Kermit Service have
     been:

     -    Issues of authentication, privacy, and anonymous access. These
          have been addressed in our implementation, as described Section 4
          of this document.

     -    Issues of coordination and control. A Kermit software program can
          be in any of several "modes": at its command prompt or menu,
          awaiting commands from the user; in terminal mode, in which the
          user's keystrokes are sent to the remote computer or service; or
          in protocol mode, in which two Kermit programs communicate via
          well-defined Kermit packets [KER]. Commands or operations valid
          in one mode do not necessarily work in another. Until now, it has
          been the user's responsibility to switch modes at one or both ends
          of the connection as needed. A companion document [TKO] to this
          one specifies a mechanism to closely couple the client and server
          via Telnet protocol negotiations, allowing each to know the
          other's state and to switch to the appropriate mode automatically
          so a valid and useful relationship obtains at all times.

     -    Lack of a standard TCP port. The "registered" port 1649 was
          assigned by IANA for this purpose (27 September 1995) and is named
          "Kermit". (renamed from "Inspect".)

3. THE INTERNET KERMIT SERVICE MODEL

   The Internet Kermit Service (IKS) uses a standard Telnet [TEL]
   connection, in which all Telnet rules apply. Unlike FTP, which
   requires additional TCP connections, IKS uses a single channel for
   both signaling and data transfer. The connection is multiplexed via
   (a) Telnet options, and (b) Kermit protocol messages. This allows
   existing Telnet clients that also support the Kermit protocol,
   whether or not they support the Telnet Kermit Option [TKO], to use
   the IKS and take advantage of all relevant Telnet options including
   authentication and encryption.

   The system Internet services daemon (e.g. inetd) waits for a
   connection on the Kermit socket (1649) and then starts the IKS on the
   new connection. The IKS performs the familiar Telnet negotiations
   including the Telnet Kermit option. Unlike a standard Telnet server,
   the IKS does not support the ability to present the user with an
   interactive system shell. The Kermit socket is used only for file
   transfer and management functions provided by Kermit file transfer
   protocol and the Kermit script language.

   Once the connection is established, the Telnet Kermit Option is
   negotiated in both directions. The results determine which of the
   following configurations is used by the Telnet client and Server:

     . Server-side Kermit Server (SKS)
     . Client-side Kermit Server (CKS)
     . No Kermit Server (NKS)

   Different procedures and functions apply to each configuration. The
   configuration may be changed at any time by Telnet Kermit Option
   subnegotiations, which assure that the Telnet client and server are
   always in compatible states.

   The three configurations are described in the following sections.

3.1. Server-Side Kermit Server

   In the Server-Side Kermit Server (SKS) configuration, the Telnet
   server is the Kermit server and the Telnet client is the Kermit
   client. This configuration is used when both Telnet client and IKS
   support the Telnet Kermit Option and the IKS sends WILL KERMIT to the
   Telnet client and receives DO KERMIT from the Telnet client [TKO].

   In this case, the IKS immediately starts a Kermit server and reports
   this to the Telnet client with a Telnet KERMIT START-SERVER
   subnegotiation.

The SKS configuration is appropriate when the user wishes to interact
only with the Telnet client's commands or menus.

If authentication was not performed with one of the Telnet
Authentication Option protocols, the Kermit server rejects all Kermit
protocol operations (except REMOTE LOGIN, REMOTE HELP, REMOTE EXIT,
BYE, or FINISH -- that is, the ones that request help, that log in,
that close the connection, or that change the status of the
connection) until:

- A Kermit REMOTE LOGIN command successfully authenticates the user;

- The login retry limit is reached;

- A Kermit BYE or REMOTE EXIT command is received, which closes the
  connection;

- A Kermit FINISH command or a Telnet KERMIT REQ-KERMIT-STOP
  subnegotiation is received to request the IKS exit from Kermit
  server mode. At this point, the IKS can either exit and close the
  connection or issue an interactive login prompt, depending on how
  it was started or configured by the system administrator.

Once the user is authenticated:

- The Telnet client configures itself for Kermit client/server
  operation, with itself as the Kermit client, communicating with
  the server only by Kermit packets, and optionally adjusting its
  menus or commands to eliminate functions (such as terminal
  emulation) that make no sense in this context.

- The relationship persists until the Telnet client and IKS agree to
  terminate the Kermit server via Kermit protocol commands (BYE,
  FINISH, or REMOTE EXIT), or by Telnet Kermit Option
  subnegotiation, or by closing the connection.

3.2. Client-Side Kermit Server

In the Client-Side Kermit Server (CKS) configuration, the Telnet
server is the Kermit client, and the Telnet client is the Kermit
server. This configuration is used when the IKS has sent WONT KERMIT
or SB KERMIT STOP-SERVER, and the Telnet Client has sent WILL KERMIT
and SB KERMIT START-SERVER, indicating that it is prepared to accept
and process Kermit protocol packets.

In the CKS configuration, the Telnet client assumes the role of
Kermit server by virtue of its ability to recognize and process
Kermit protocol packets in its terminal emulator. Thus the Telnet

client must not send WILL KERMIT or the KERMIT START-SERVER
subnegotiation unless its terminal emulator is capable of recognizing
Kermit packets.

If the IKS is at top command level (as opposed to executing a
script), or when it reaches top level after finishing a script, it
issues its interactive command prompt.

At this point, the user may type commands or send scripted commands
to the IKS command prompt. When a data-transfer command (such as
SEND) is issued by the user at the IKS prompt, a Kermit packet is
transmitted and recognized by the Telnet client, causing it to
automatically perform the requested action (e.g. receive a file), and
then resume its previous mode (terminal emulation or script
execution) when the data transfer is complete.

Thus, in the CKS configuration, data transfers are initiated by the
IKS rather than by the Telnet client. This configuration is useful
when the user prefers the command interface or repertoire of the
server to that of the client.

If the IKS sends a Telnet KERMIT START-SERVER subnegotiation, the
relationship switches automatically to Server-Side Kermit Server
(Section 3.1), in which the Telnet client is the Kermit client and
the Telnet server is the Kermit server.

If the Telnet client sends a KERMIT STOP-SERVER subnegotiation, the
connection switches to No Kermit Server (Section 3.3) and the IKS
issues its command prompt. At this point, neither side is a Kermit
server, and both sides may optionally disable Kermit protocol
commands. Subsequent user action can designate one side or the other
as the Kermit server, as desired.

3.3. No Kermit Server

If both Telnet client and IKS send WONT KERMIT or SB KERMIT STOP-
SERVER, or if the Kermit client and server are connected across
multiple hosts or transports, thus precluding end-to-end Telnet
negotiation, a Kermit server is not known to be available. In the
KERMIT STOP-SERVER case, the Kermit partners can later switch back to
SKS or CKS, but in the other two cases, there is no such signaling
and loose coupling characterizes the entire session.

In the No Kermit Server (NKS) configuration, the IKS presents a
command prompt to the Telnet client. As in the Client-Side Kermit
Server configuration, plain-text commands are issued to the IKS.

In the loosely coupled NKS configuration, the Telnet client does not
know the state of the Telnet server, and so can not automatically
adjust its commands and menus to present only valid choices, or
automatically change its state to complement the server's; it is the
user's responsibility to assure that the "mode" (command prompt,
terminal emulation, server command wait) of each Kermit partner is
appropriate for each action. Thus an Internet Kermit Server appears
as an ordinary remote Kermit program to any Telnet client that does
not implement the Telnet Kermit Option, or in which this feature is
disabled or can not be used.

The NKS configuration allows successful manual operation of the IKS
through Telnet clients that do not support the Telnet Kermit Option.
The Telnet client might or might not support Kermit "autodownload"
and "autoupload"; if it does not, then the user is forced to manually
issue command on both sides of the connection in the traditional and
familiar manner [CKB,CMG,K95].

4. SECURITY CONSIDERATIONS

4.1. AUTHENTICATION

   Authentication is provided via one or more of the following methods:

     - The Telnet AUTHENTICATION option;

     - The Telnet START_TLS option;

     - Plaintext userid/password verification.

4.1.1. Telnet Authentication option

   The use of one of the many Telnet authentication option methods
   removes the need to transmit passwords in plaintext across public
   networks. In addition, the exchange of user authentication
   information often provides a shared secret that can be used with the
   Telnet Encryption Option protocols to encrypt the connection in one
   or both directions.

   Telnet authentication may also be used in conjunction with the Telnet
   START_TLS option to negotiate end user identity over the encrypted
   and host authenticated TLS channel.

   The IKS currently supports Kerberos 4, Kerberos 5, Secure Remote
   Password and Microsoft NTLM authentication methods via the Telnet
   AUTH option.

4.1.2. Telnet over TLS option

   The Telnet START_TLS option provides for the negotiation and
   establishment of a TLS version 1 session after the initial telnet
   connection. The TLS connection provides host to client
   authentication via the use of X.509 certificate chains. TLS also
   supports optional client to host authentication using host verified
   X.509 certificates which may be used to authenticate a userid
   provided by the client or be mapped to a userid based upon properties
   of the certificate.

4.1.3. Plaintext Authentication via Kermit REMOTE LOGIN

   In the Server-Side Kermit Server configuration, if the client is not
   yet authenticated, the client must log in using a REMOTE LOGIN
   command, in which a Kermit packet containing user ID and password in
   clear text is sent from the Telnet client to the Telnet server, which
   then calls upon local mechanisms to authenticate the user. Any
   packets other than login (or REMOTE HELP, REMOTE EXIT, FINISH, or
   BYE) packets are rejected (returned with an error message) until the
   user is authenticated. If the number of unsuccessful login attempts
   exceeds the limit, the connection is closed. Many Kermit client
   programs support this login method already.

   This method should be avoided whenever possible. If plaintext
   passwords are used, they should only be sent after the Telnet START-
   TLS option has been negotiated (see 4.2.2). Otherwise, passwords are
   open to packet sniffing.

4.1.4. Plaintext Authentication via Command Prompt

   In the Client-Side Kermit Server and No Kermit Server configurations,
   the server presents the user with a plain-text interactive interface
   that begins with the server issuing "Username:" and "Password:"
   prompts, just as if the user were logging in to a multiuser
   timesharing system such as VMS or UNIX. When a password is not
   required an empty response can be given. Invalid username-password
   combinations result in a new series of prompts up to the login retry
   limit, and then disconnection.

   This method should be avoided whenever possible. If plaintext
   passwords are used, they should only be sent after the Telnet START-
   TLS option has been negotiated (see 4.2.2). Otherwise, passwords are
   open to packet sniffing.

4.1.5. Anonymous Login

   When the username is "anonymous" or "ftp", the IKS behaves like an
   anonymous ftp server, in a manner appropriate to the underlying
   platform. In UNIX, for example, access is restricted to a designated
   area of the file system. A password might or might not be required,
   according to the preference of the site administrator.

   If privacy is desired the Telnet START-TLS option should be used (see
   4.2.2).

4.2. ENCRYPTION (PRIVACY)

   As the Internet becomes ever more public and susceptible to
   eavesdropping, it becomes increasingly necessary to provide methods
   for private access to services. Telnet provides two such mechanisms:

    . Telnet Encryption option
    . Telnet START-TLS option

4.2.1. Telnet Encryption option

   The Telnet Encryption option, although it has never achieved RFC
   status, has been used for years in conjunction with the Telnet Auth
   option in Telnet clients and servers that support Kerberos 4,
   Kerberos 5, Secure Remote Password, and others. The IKS currently
   supports the following encryption methods under the Telnet Encryption
   option:

    . cast128_ofb64
    . cast5_40_ofb64
    . des_ofb64
    . cast128_cfb64
    . cast5_40_cfb64
    . des_cfb64

4.2.2. Telnet over TLS option

   Transport Layer Security (TLS), the successor to Secure Sockets Layer
   (SSL), provides methods to implement Server authentication, Client
   authentication, and Transport Layer encryption. Unlike Telnet
   Encryption, Start-TLS does not require the use of Telnet
   Authentication in order to provide a private channel. This means
   that it can be used in conjunction with plaintext passwords and
   anonymous connections.

5. SERVICES

   The Internet Kermit Service includes features for both users and
   system administrators. The IKS is incorporated into the 7.0 release
   of Columbia University's C-Kermit software, which is the "master"
   Kermit software program in terms of features and command language.
   An overview of C-Kermit can be found at:

     http://www.columbia.edu/kermit/ckermit.html
     http://www.kermit-project.org/ckermit.html

   When C-Kermit is employed as an Internet Kermit Service, it may offer
   all its functions to "real" users (those who are authenticated as
   specific users), and a safe subset of its functions to anonymous
   users.

   The Internet Kermit Service resembles an FTP server in that it
   performs its own authentication and uses a well-defined protocol to
   communicate with its client, but differs from the FTP server by also
   offering (at the system manager's discretion) an interactive user
   interface to the Telnet client when it is in terminal mode. It also
   differs from FTP in restricting all protocol messages and data
   transfer to a single socket connection.

   An IKS has been deployed at Columbia University for worldwide public
   access to the Kermit FTP site:

     telnet://kermit.columbia.edu:1649/
     telnet://ftp.kermit-project.org:1649/

5.1. Features for System Administrators

   The system administrator can supply IKS configuration parameters as
   command-line options or in a configuration file, or both in
   combination. Such parameters include:

    . Whether anonymous logins are allowed.

    . The file system or root directory to which anonymous users are
      restricted.

    . Specification of permissions and other attributes to be assigned
      to files uploaded by anonymous users.

    . Whether to make session entries in system logs.

. Specific services to disable: reception of files, sending of
  files, sending of email, printing, changing of directories,
  getting directory listings, deleting files, etc (see next
  section).

. Whether access to the interactive command prompt is allowed.

5.2. Features for Users

   The IKS supports a wide range of services, including, but not limited
   to, the following:

. Authentication as a real user or anonymously.

. Transmission of files to which read access is allowed.

. Reception of files into directories or devices to which write
  access is allowed.

. The ability to display a file on the client's screen.

. Ability to list files.

. Ability to change its working (default) directory.

. Ability to delete files to which write or delete access is
  allowed.

. Ability to rename and copy files

. Ability to create and remove directories.

. The ability to route received files to a specified printer, or to
  send them as email to a specified address list.

. Client control of server parameter settings, within limits
  established by the server system administrator.

. Transmission of variables from client to server or vice versa.

. Remote and local script execution.

. Remote and local procedure execution.

File transfer features include:

. Kermit text-mode transfers incorporate not only record-format
  conversion, but also character-set translation;

. Kermit can switch automatically between text and binary mode on a
  per-file basis when sending groups of files.

. A selection of file collision options, including "make backup copy
  of existing file and accept incoming file", "reject incoming
  file", "accept incoming file only if newer than existing file",
  etc.

. Numerous methods for selecting the files to be transferred,
  including pattern matching, lists of filenames (or patterns),
  exception lists, date and/or size ranges, etc.

. Filename conversion and file renaming.

. Automatic directory creation if elected and enabled.

. Standard mechanisms for directory traversal, allowing transmission
  of entire directory trees or other file hierarchies even between
  unlike file systems such as VMS, UNIX, and Windows.

. Atomic file movement: optionally, the source file can be deleted
  (or renamed, or moved) when and only when it has been transferred
  successfully.

. Kermit can retain file attributes including time stamps and
  permissions (at the user's or system administrator's discretion),
  even between unlike platforms;

. Recovery of interrupted transfers from the point of failure.

. File-transfer pipes and filters.

Script programming features include:

. Macros with parameter substitution.

. Built-in and user-defined variables and arrays, with global or
  local scope.

. Built-in and user-defined functions. Built-in functions include:

        - String functions
        - Arithmetic functions
        - Date / time functions
        - File functions

. Input search for multiple simultaneous targets.

. IF-ELSE, WHILE, FOR, SWITCH, GOTO, C-like block structure.

. Every command returns a completion status that may be tested and
  used as a basis for subsequent actions.

5.3. User Interface

   The Internet Kermit Service uses the Kermit command and script
   language, as implemented in Columbia University's C-Kermit
   communication software [CKB]. This program and its command language
   are portable to all known varieties of UNIX, as well as to Windows
   95/98/NT, OS/2, Digital (Open)VMS, Stratus VOS, Data General AOS/VS,
   Plan 9, OS-9, QNX, the Commodore Amiga, and other platforms. The
   C-Kermit command language is a superset of that of other Kermit
   software programs including MS-DOS Kermit for DOS and Windows 3.x,
   IBM Mainframe Kermit for VM/CMS, MVS/TSO, CICS, and MUSIC, PDP-11
   Kermit for RT-11, RSTS/E, RSX-11, and IAS, and dozens of other Kermit
   programs.

   It is far beyond the scope of this document to enumerate, let alone
   describe, the commands and services of C-Kermit; this is the subject
   of a 600-page book [CKB], augmented by hundreds of pages of online
   material. A brief overview is included here.

   Commands are based on English words. There is no plan at present to
   support other natural languages (Italian, Portuguese, Norwegian,
   Russian, Hebrew, Japanese, Cherokee, etc) as alternative bases for
   command words, since this would reduce the portability of scripts.
   However, since the command language includes a macro capability,
   macros may be defined to provide selected commands in different
   languages if desired.

   Certain commands can apply either locally or remotely, for example
   "CD" (Change Directory). The convention is to prefix the command
   with the word REMOTE if it is to apply remotely. Example: "cd foo"
   changes to the "foo" directory on the computer where the command was
   given; "remote cd foo" sends a Kermit packet to the Kermit server
   requesting it to change its directory to "foo". The commands in this
   category include:

```
   ASSIGN <variable> <value>       Assign a value to a variable.
   CD <directory>                  Change working directory.
   COPY <files> <destination>      Copy file(s)
   DELETE <files>                  Delete file(s)
   DIRECTORY [ <pattern> ]         List file(s)
   EXIT                            Exit
   HELP [ <topic-or-command> ]     Display help text
   MKDIR <directory>               Create a directory
   PRINT <files>                   Print file(s)
   PWD                             Print working directory
   RENAME <old> <new>              Rename file(s)
   RMDIR <directory>               Remove a directory
   SET <parameter> <value>         Change a parameter's value
   TYPE <file>                     Display the contents of a file
```

   As a convenience, REMOTE commands also have short synonyms: RASSIGN,
   RCD, RCOPY, RDELETE, and so forth.

   The basic file transfer commands are:

```
   SEND [ modifiers ] <files>    Send file(s) (to server)
   GET [ modifiers ] <files>     Get file(s) (from server)
```

   These commands take a file name, pattern, or list, plus various
   optional modifiers, including transfer mode specifiers (text,
   binary), file selectors (date, size, exception list), aliasing, name
   and path options, disposition specifiers, and so on.

   In addition to the commands listed above, the following commands are
   sent by the client to the server:

```
   REMOTE QUERY                    Get value of variable or procedure
   BYE                             Log out and close the connection
   FINISH                          Request the server leave server mode
```

   Like all Kermit client/server commands, these can be disabled if
   desired.

   Of course there are numerous other commands with purely local effect,
   such as the many scripting commands. These, plus all the commands
   above, are fully documented in [CKB]. The repertoire grows over
   time, but never in a way that invalidates existing scripts.

   The system administrator can allow or forbid access to any of these
   features, and to the command language as a whole. In the latter
   case, the IKS may be accessed only as a Kermit server, by giving
   commands to the client.

6. REFERENCES

   [TKO] Altman, J. and F. da Cruz, "Telnet Kermit Option", RFC 2840,
         May 2000.

   [BCP] Bradner, S., "Key words for use in RFCs to Indicate Requirement
         Levels", BCP 14, RFC 2119, March 1997.

   [KER] da Cruz, Frank, "Kermit, A File Transfer Protocol", Digital
         Press/ Butterworth Heinemann, Newton, MA (1987). 379 pages,
         ISBN 0-932376-88-6.

   [CKB] da Cruz, Frank, and Christine M. Gianone, "Using C-Kermit",
         Second Edition, Digital Press / Butterworth-Heinemann, Woburn,
         MA (1997). 622 pages, ISBN 1-55558-164-1.

   [CMG] Gianone, Christine M., "Using MS-DOS Kermit", Second Edition,
         Digital Press / Butterworth-Heinemann, Woburn, MA (1992). 345
         pages, ISBN 1-55558-082-3.

   [K95] Gianone, Christine M., and Frank da Cruz, "Kermit 95", Manning
         Publications, Greenwich CT, (1996). 88 pages, ISBN 1-884777-
         14-7.

   [PRF] Huggins, James K., "Kermit Protocol - Formal Specification and
         Verification", in Boerger, E., "Specification and Validation
         Methods", Oxford University Press (1995). ISBN 0-19-853854-5.

   [FTP] Postel, J. and J. Reynolds, "File Transfer Protocol (FTP)", STD
         9, RFC 959, October 1985.

   [TEL] Postel, J. and J. Reynolds, "Telnet Protocol Specification",
         STD 8, RFC854, May 1983, et seq.; "Telnet Option
         Specification", STD 8, RFC855, May 1983, et seq.

   [IAN] Internet Assigned Numbers Authority:
         http://www.iana.org/numbers.html
         http://www.iana.org/assignment/port-numbers

7. AUTHORS' ADDRESSES

Frank da Cruz

EMail: fdc@columbia.edu

Jeffrey E. Altman

EMail:jaltman@columbia.edu

The Kermit Project
Columbia University
612 West 115th Street
New York NY 10025-7799
USA
http://www.columbia.edu/kermit/
http://www.kermit-project.org/

8. Full Copyright Statement

Acknowledgement