Columbia University Academic Information Systems

Kermit News

Number 5, July 1993

This is not the real cover. The real cover shows a three-dimensional bar graph comparing XMODEM, YMODEM, ZMODEM, and Kermit file transfer for ASCII, Binary, and Precompressed files, on three different types of connections.

Contents

Kermit News *(ISSN 0899-9309) is published periodically free of charge by Kermit Development and Distribution, Columbia University Academic Information Systems, 612 West 115th Street, New York, NY 10025, USA. Contributed articles are welcome.*

Editor: *Christine M. Gianone*
E-Mail: `cmg@columbia.edu` *or* `kermit@cuvma.bitnet`

Editor's Notes

Welcome to *Kermit News* Number 5. This issue announces new releases of **MS-DOS Kermit** for DOS and Windows; **C-Kermit** for UNIX, OpenVMS, OS/2, AOS/VS (and several other operating systems); and **IBM Mainframe Kermit** for VM/CMS, MVS/TSO, MUSIC, and now CICS.

The new generation of Kermit communications software offers improved efficiency, networking capabilities, automation features, and an understanding of national and international character sets. Together, the three major Kermit software programs —MS-DOS Kermit, C-Kermit, and IBM Mainframe Kermit —form a high-quality, powerful, and fully interopable suite of communication programs for the industry's most popular computers.

Credit Cards Accepted

We are also pleased to announce our new ability to accept Visa and MasterCard orders for Kermit software and documentation. This should simplify the ordering process for everyone.

It's KermitWare!

While Kermit software is not a commercial product, it is not in the public domain either, and never has been. It is not ''shareware.'' It's not ''freeware.'' It *is* copyright by Columbia University. See page 24 for our terms and conditions.

As with any software, high-quality documentation is essential. It makes the user self-sufficient, it reduces the burden on the organizational help desk (and our own!), and documentation sales help supply us with the income we need to continue our efforts. In this issue of *Kermit News*, I am pleased to announce the publication of a new edition of *Using MS-DOS Kermit* and the new book, *Using C-Kermit*.

Kermit Saves You Money

In today's economy, it has become increasingly important to obtain high quality at low cost. In ever-increasing numbers, government contractors are proposing Kermit software and documentation in their bids instead of commercial communication software. In the health care industry, Kermit software is rapidly becoming the standard link from the doctor's office, pharmacy, or hospital to the claims clearinghouse or insurance company. Electronic claims submission can speed the process and eliminate billions of dollars in paperwork each year.

In countless cases, Kermit is chosen over other alternatives —ranging from public-domain and shareware packages, to expensive commercial software, to in-house development efforts —because the *quality is high* and the *cost is low.*

- If you are a vendor of modems, PCs, PC LANs, servers, laptops, notebooks, or palmtops, contact us about the benefits of —and easy terms for —bundling MS-DOS Kermit software with your product.
- If you are a vendor of UNIX software and/or UNIX-based computers, ask us about bundling C-Kermit.
- If you are a systems integrator, government contractor, or vendor of any kind of computer hardware or software product or service, ask us how you can use Kermit software to lower your costs, improve your operation, and get the edge on your competitors.
- If your organization is an end-user of computer hardware and software —a corporation, a government agency, a hospital, a university or other nonprofit or public institution —Kermit software can meet your connectivity needs and stay within your budget.

And if you are simply a private individual who wants to communicate from home to office, to get online with commercial data services like MCI Mail or CompuServe, to hook up with BBSs, to transfer data efficiently and reliably between almost any conceivable pair of computers, Kermit software is for you!

Sprechen Sie Français?

Using MS-DOS Kermit, second edition, has been translated into German by Gisbert W. Selke, and published in Germany (see story, page 17):

> Christine M. Gianone, *MS-DOS Kermit, das universelle Kommunikationsprogramm*, Verlag Heinz Heise, Hannover (1991).

And into French by Jean Dutertre, and published in France:

> Christine M. Gianone, *Kermit MS-DOS Mode d'Emploi*, Heinz Schiefer & Cie., Versailles (1993).

And there is also a new Japanese book about MS-DOS Kermit:

> Hirofumi Fujii and Fukuko Yuasa, *MS-Kermit Nyumon*, Computer Today Library 6, Saiensu-Sha Co., Ltd., Tokyo (1993).

Announcing MS-DOS Kermit 3.13

MS-DOS Kermit is widely recognized as one of the most powerful, efficient, and flexible of all PC communication software packages for DOS and Microsoft Windows.

While MS-DOS Kermit's pricey competitors focus on frills like sound effects, elaborate startup screens, and technicolor pop-up exploding menus that consume your PC's memory, disk, and processor capacity, MS-DOS Kermit stresses *substance*: fast, reliable, high-quality terminal emulation and file transfer in a wide variety of communication, computing, and language environments. Small size and efficient operation. Easy setup and configuration. Powerful, easy-to-use key mapping, macros, and script programming. And low cost.

Now MS-DOS Kermit extends its reach even further with *more terminal emulations*, *more communication methods*, *faster file transfer*, and *more languages*.

Version 3.13 of MS-DOS Kermit for the IBM PC, PS/2, and compatibles is now available. It was prepared by Professor Joe R. Doupnik of the Center for Atmospheric and Space Sciences and Department of Electrical Engineering of Utah State University in Logan, Utah, USA, in cooperation with Columbia University in New York City and Waterloo University in Ontario, Canada.

Built-in TCP/IP Networking

A major new feature of version 3.13 is its built-in support for TCP/IP networking, adapted from parts of Erick Engelke's Waterloo TCP package and expanded considerably to include TELNET protocol, **multiple simultaneous sessions**, and more. TCP/IP (Transmission Control Protocol / Internet Protocol) is the worldwide standard for open networking.

Why add TCP/IP to Kermit? Until now, people who use both network and serial connections have had to switch between a TCP/IP TELNET program (which doesn't support serial connections) and Kermit (which didn't support TELNET connections). No more! Now you you can use all of Kermit's powerful features in the TCP/IP environment in place of TELNET and FTP, exactly as you use them now on serial connections: script programming, modem dialing scripts (when dialing out from TCP/IP terminal servers), flexible key mapping, keyboard and command macros, fast and accurate text and graphics terminal emulation, and international character-set translation in both file transfer and terminal emulation.

Perhaps most important of all, now you can have a single application program, a single configuration file, and a common user interface for both serial and network communication.

Kermit's TCP/IP and TELNET support works over Ethernet- or SLIP-class packet drivers available from your network board vendor or from us, as well as over ODI network drivers or on a serial port via Novell's new SLIP_PPP ODI driver, and also (via a ''shim,'' which we also supply) over NDIS drivers.

Other Networks

MS-DOS Kermit supports other networks too: AT&T StarLAN/StarGROUP, Digital Equipment Corporation PATHWORKS (both LAT and CTERM protocols), IBM EBIOS/LANACS, NETBIOS, Intel OpenNET, Novell NASI/NACS, Novell LAN Workplace for DOS TELAPI, InterConnections Inc. TES, 3Com BAPI, and Ungermann-Bass Net/One, plus any BIOS Interrupt 14 interceptor for other network services, including external TCP/IP packages such as those from Beame & Whiteside, FTP Software, or Wollongong.

Compare MS-DOS Kermit with commercial PC communication software packages. How many of them support such a wide array of networks? *And how many offer this support at no extra cost?*

Character Sets for Many Languages

Since version 3.0 was released in 1990, MS-DOS Kermit has been capable of preserving the national and international character sets used for **Western European** languages such as Italian, Norwegian, Portuguese, French, German, Spanish, Dutch, Swedish, Danish, Finnish, and Icelandic during both terminal emulation and file transfer.

Character-set conversion is essential in the international market. Different computers use different encodings for the ''special characters'' found in these languages; Kermit software can reconcile the differences.

Now MS-DOS Kermit also handles **Eastern European** languages like Czech, Polish, Romanian, and Hungarian. And languages written in the **Cyrillic** alphabet such as Russian, Bulgarian, Byelorussian, and Ukrainian. It handles **Hebrew** too, including right-to-left screen writing and a full range of Hebrew VT100 and VT420 terminal features. MS-DOS Kermit 3.13 can even convert **Japanese Kanji** character sets during file transfer.

MS-DOS Kermit can transfer text files written in over 30 different languages with other computers, even when they use completely different encodings, if the other computer is equipped with a Kermit program that uses this technique. IBM mainframe Kermit and C-Kermit 5A do. This is a *unique feature* of the Kermit file transfer protocol, and you won't find it in any other communications software.

Terminal Emulation

MS-DOS Kermit's DEC VT terminal emulation is widely recognized as an industry leader: fast, accurate, powerful, and full-featured. Now, thanks to a development grant from Data General Corporation, MS-DOS Kermit 3.13 adds **Data General DASHER** D463 and D470 text and graphics terminal emulations to its repertoire, suitable for use with DG's AOS/VS-based CEO system. Special capabilities include horizontal scrolling, compressed text, support for the DG International character set, and mouse support for CEO Draw.

The new release includes other text and graphics terminal emulation improvements, too: **132-column** compressed text and horizontal scrolling in VT terminal emulation on EGA and VGA video adapters; a **compose key** for entering accented letters; screen rollback buffers and graphics images are now stored in **expanded memory** (EMS), if available, allowing thousands of rollback screens while freeing conventional memory for other uses. Terminal emulations offered by MS-DOS Kermit now include:

DEC VT52, VT100, VT102, VT220, VT320
DEC Sixel Color Graphics
DG DASHER D463, D470 Text *and* Graphics
Heath/Zenith 19
Tektronix 4014 Graphics with extensions
NONE (for external console drivers, e.g. ANSI)

Faster File Transfer

MS-DOS Kermit 3.13 includes important new file transfer efficiency improvements. The limit on packet buffers has been increased from 2K to 280K to attain the theoretical maximum of 31 window slots of 9024-byte packets for faster transfers on long-distance and network connections. Packet lengths now adapt dynamically to noise conditions, and parity is detected automatically during packet operations. For extra speed, selected control characters can be ''unprefixed'' during file transfer (see the article on page 10). The file transfer display has been expanded and improved and, finally, a new capability has been added for transferring files with IBM mainframes through 3270 protocol converters that lack a transparent mode (see page 6).

Modems and Dialing

Kermit's new **dialing directory** is a plain-text file that Kermit searches automatically whenever you give a DIAL command. Each entry contains a name, the associated phone number, the dialing speed, and the parity. It can be as long as you like. Here is a short (fictional) sample:

```
sprintnet   7654321   2400  mark
tymnet      876-5432  1200  even
mcimail     987-6543 19200  none
compuserve  555-1212  9600  space
```

Just type ''dial sprintnet'' and Kermit does the rest: sets your speed and parity appropriately, commands your modem to place the call, and tells you (or your script program) whether the call succeeded or failed.

As always, MS-DOS Kermit's dialing is accomplished via script programs. In addition to the standard Hayes script, new scripts are furnished for Telebit, US Robotics, Multitech, Penril, Practical Peripherals, Supra, Vadic, and other modems, and for Rolm (Siemens) CBX data phones.

For high-speed modems, MS-DOS Kermit now offers bidirectional RTS/CTS **hardware flow control**, and incorporates new defensive techniques required for the new breed of low-cost high-speed internal modems, and new controls for using them as COM3 or COM4 devices. 14400 bps is now supported as an interface speed, for use with V.32*bis* modems, and to comply with PTT regulations in many countries MS-DOS Kermit also now supports 75/1200 bps split-speed operation.

Scripts and Macros

Kermit's script programming and macro features have been significantly extended. Macros can be longer. Command macros can now be assigned to keystrokes. Long variable names are allowed. New built-in variables represent the current date, time, directory, etc. DOS files and environment variables can be accessed by Kermit commands. And escape sequences sent by the host can run MS-DOS Kermit commands, for fully host-driven operation.

New Documenation

The user manual, *Using MS-DOS Kermit*, was updated and a second edition published in 1992. It describes the new script programming and key mapping capabilities, and includes new character-set tables, a new chapter on TCP/IP and other networks, a complete specification of the VT and Tektronix terminal emulators, and much more. Now available in French and German (see page 1) as well as English.

C-Kermit 5A(189) for UNIX, VMS, OS/2, AOS/VS, . . .

Quite possibly the world's most portable communication software package, C-Kermit 5A(189) for UNIX, DEC VMS and OpenVMS, IBM OS/2, Data General AOS/VS, Apollo Aegis, MicroWare OS-9, the Commodore Amiga, and the Atari ST, offers:

- Portability and consistency across hundreds of hardware and software platforms
- Efficient terminal connection and file transfer
- International appeal
- Automation features
- Network support

Portability

The new C-Kermit release delivers high-quality communications in all eight major operating systems it supports. It is distributed in C-language source code form and also in selected binary formats. It is available for computers ranging from desktop PCs and workstations to large central systems and supercomputers.

C-Kermit's modular design has promoted its adaptation to a diverse collection of computers and communication methods, making it a premiere example of open and portable software. C-Kermit's user interface is easy to learn and use, helpful to the novice without getting in the way of the expert, and consistent throughout a wide range of operating systems and hardware platforms. Users of many types of computers now have a single software package to meet both serial and network communication needs.

UNIX C-Kermit

C-Kermit 5A is available for the major UNIX variants, new and (within reason) old, including AT&T System III and V (R2, R3, R4), BSD (4.1–4.4 as well as 2.11), OSF/1, and POSIX, plus most popular commercial UNIX products, including AIX, A/UX, BSDI, DG/UX, DNIX, DYNIX, DYNIX/PTX, DRS/NX, ESIX, HP-UX, IRIX, NeXTSTEP, QNX, SCO (XENIX, UNIX, and ODT), Solaris, SunOS, ULTRIX, UMAX, UMIPS, UNIPLUS, UnixWare, UNOS, and many more (see our version list on page 25).

VMS C-Kermit

Users of DEC (Open)VMS can bid a fond farewell to Kermit-32. Support disappeared years ago, never to be fully replaced because Kermit-32 is written in a language (Bliss) found at very few sites. C-Kermit 5A is available for both VAX and Alpha AXP versions of (Open)VMS, on processors ranging from desktop workstations to large enterprise mainframes, and offers all the capabilities of Kermit-32 plus all the new capabilities of C-Kermit.

Unlike its predecessors, C-Kermit 5A has a comprehensive understanding of the VMS file system. When sending files, C-Kermit automatically selects the appropriate mode, text or binary, based on each file's record format. A new feature also allows more complex VMS files to be transferred in a way that preserves all of their RMS attributes.

OS/2 C-Kermit

C-Kermit 5A is a full-function communications software package for OS/2 2.0 and 2.1 as well as for OS/2 1.3. In addition to all of C-Kermit's regular features, the OS/2 version includes its own built-in VT102 terminal emulator complete with a sizeable screen, screen rollback, color control, printer control, key mapping, and other features familiar to users of MS-DOS Kermit. When run in an OS/2 window, C-Kermit also allows cutting and pasting, background file transfer, and easy switching among applications. Early reports rate C-Kermit's VT102 emulator among the best available for OS/2.

AOS/VS C-Kermit

C-Kermit 5A is also available for Data General Eclipse MV-Series minicomputers running AOS/VS and AOS/VS II. This full-featured C-Kermit version is already seeing heavy service at NASA (where, for example, it is installed in the Spacelink information server) and the US Forest Service.

File Transfer Performance

Since its previous release, 4E(072) in 1989, C-Kermit's file transfer efficiency has been improved dramatically by the addition of control-character ''unprefixing,'' sliding windows (up to 31 window slots), and long packets (up to 9024 bytes), bringing file transfer efficiency —even over long-delay satellite and/or public network connections —into the 95%–100% range, or better, with excellent error recovery characteristics on noisy connections.

The sliding window transport, perfected over three years of field testing, uses selective retransmission to minimize overhead on noisy connections, and the packet length adapts automatically to noise characteristics. Errors are detected by 16-bit CRC and other methods. For more about efficiency, plus real-life benchmarks comparing Kermit with other protocols, see the article on page 10.

International Appeal

The Kermit protocol stands alone in its ability to convert a file's character set during transfer in a mixed computing environment. For example, an Italian-language document written in Code Page 437 on a PC is correctly translated during transmsission to (say) a DEC workstation using the DEC Multinational Character Set. Despite their differing internal representations, the accented letters come out the same on both ends, rather than the gibberish often seen when transmitting such text across vendor, language, or character-set boundaries.

C-Kermit's character-set conversion capabilities are not confined to **West-European** languages like Italian, French, German, Spanish, Norwegian, and Portuguese, but also extend to **East-European** languages like Hungarian, Czech, Polish, and Romanian, as well as to languages written in the **Cyrillic** alphabet, like Russian and Ukrainian, and also to **Hebrew** and to **Japanese Kanji**.

Most of the same conversions are also available during terminal connection, screen capture, and ''ASCII upload'', and can also be used to change a local file's character-set ''in place''. Language-specific rules (e.g. ''ä'' to ''ae'') can be applied when translating text written in languages like German, Dutch, Swedish, or French into ASCII.

In the 7-bit communication environment —an area neglected or ignored by most other protocols —efficient transfer of 8-bit textual data (such as Cyrillic or Kanji) is achieved using a new locking-shift mechanism, discussed on pages 11 and 14.

The advanced Kermit protocol features of C-Kermit 5A can be used to full advantage with MS-DOS Kermit on PCs with DOS or Windows, IBM Mainframe Kermit-370 for VM/CMS, MVS/TSO, or CICS, and, of course, with another copy of C-Kermit 5A itself on UNIX, VMS, OS/2, AOS/VS, or any of the other operating systems where it runs.

Automation Features

C-Kermit's automation features include key mapping and keystroke macros for use during terminal connection, plus command and initialization files, command macros, and a fully functional **script programming language**. Any kind of routine communication task can be fully automated, from connection establishment, to logging in, to interacting with a remote host or service, to file transfer, to logging out and connection release. C-Kermit's script language is almost fully compatible with MS-DOS Kermit's, and script programs can be written that work with both.

Communication Features

Dialing is accomplished with C-Kermit's expanded built-in support for a wide variety of modems, plus a new text-based dialing directory compatible with MS-DOS Kermit's, and an even more powerful text-based services directory that not only establishes your connection but also logs you in automatically. The OS/2 and many of the UNIX C-Kermit versions (depending on the capabilities of the underlying operating system) also support hardware flow control for use with terminal servers and high-speed modems.

C-Kermit 5A supports not only serial (direct and dialed) connections, but also **TCP/IP connections** in most UNIX versions, in the AOS/VS version, in the OS/2 version, and also for VMS systems equipped with DEC TCP/IP (UCX), TGV MultiNet, Wollongong WIN/TCP or PathWay, or with Process Software TCPware.

On Sun computers equipped with SunLink X.25, C-Kermit 5A also supports **X.25 connections**. And on OS/2 systems equipped with DEC PATHWORKS, C-Kermit can make **DECnet LAT** connections.

On TCP/IP networks, C-Kermit can be used in place of TELNET and FTP with several advantages. C-Kermit's TELNET connections handle character-set conversion, key mapping and macros, session logging, and other functions lacking from normal TELNET software. C-Kermit's DIAL command command can be used to place calls using modems that are connected to network-accessible reverse terminal servers.

Kermit file transfer offers features lacking from FTP: correct handling of file size and date, character-set conversion, an update feature, numerous options for handling filename collisions, convenient methods of transfer interruption, and so on. And, unlike traditional TELNET and FTP programs, C-Kermit's network operations can be fully automated.

Documentation

C-Kermit 5A comes with the new Digital Press book, *Using C-Kermit*, by Frank da Cruz (author of *Kermit, A File Transfer Protocol*) and Christine M. Gianone (author of *Using MS-DOS Kermit*), which is geared towards both the beginner and the expert. The book includes tutorials, numerous illustrations and tables, hundreds of examples, as well as easy-to-use and comprehensive reference features.

A German-language edition, *C-Kermit, Einführung und Referenz*, will be available in October from Verlag Heinz Heise in Hannover, Germany (see article on page 17).

IBM Mainframe Kermit-370 for VM/CMS, MVS/TSO, CICS

John F. Chandler
Harvard/Smithsonian Center for Astrophysics
Cambridge, Massachusetts

There have been several new releases of IBM mainframe Kermit-370 since version 4.2 was announced in the last issue of *Kermit News*. Most notable among these is a brand-new version for CICS, which can be used under either MVS or DOS/VSE. The new releases run on all the major operating systems found on the IBM System/370 (390, 9000) architecture, including XA and ESA:

CICS Kermit 4.2.4
For CICS versions 1.6–2.1 under MVS, DOS/VSE, and probably also VM (this has not yet been tested), supporting the full range of linemode and fullscreen environments that are supported by the other Kermit-370 versions.

TSO Kermit 4.2.4
For MVS/TSO, MVS/XA/TSO, and MVS/ROSCOE/ETSO.

CMS Kermit 4.2.5
For VM/CMS, VM/ESA/CMS, VM/XA/CMS, VM/HPO/CMS, VM/IS/CMS, and VM/SP3–SP6.

MUSIC Kermit 4.2.3
For the MUSIC/SP operating system, with MUSIC specifics by Pierre Goyette of McGill University.

Major new features include:

- Support for more communication environments and front ends, notably the IBM 3174 AEA B2.0. Support for 8-bit no-parity file transfer through front ends that support it, and automatic detection of a wider variety of front ends and 3270 protocol converters.
- Support for more character sets. In addition to the ISO-based Roman, Cyrillic, Greek, Hebrew, and Japanese Katakana character sets supported by version 4.2, the new releases also support Latin-2 and Latin-3 based character sets for the East European Roman-alphabet languages, and full Japanese Kanji. Kermit-370 now supports character-set conversion for the following languages: Afrikaans, Albanian, Basque, Breton, Bulgarian, Byelorussian, Catalan, Croatian, Czech, Danish, Dutch, English, Esperanto, Faeroese, Finnish, French, Frisian, Gaelic, Galician, German, Greek, Hebrew, Hungarian, Icelandic, Italian, Japanese (both Katakana and Kanji), Latin, Macedonian, Maltese, Norwegian, Polish, Portuguese, Quechua, Rhaetian, Romanian, Russian, Serbian, Slovak, Slovenian, Spanish, Swahili, Swedish, Turkish, Ukrainian, and Volapuk.
- Support for the new Kermit protocol locking-shift mechanism (see page 11).
- Better recovery from communication I/O errors, improved on-line help, carriage-control conversion, improved support for Generation Data Groups in TSO, and improved installation procedures.
- Support for a new method of Kermit file transfer through 3270 protocol emulators, such as the 3708, that do not offer transparent mode.

CICS Kermit is already serving diverse applications:

- The CICS-based NOTIS bibliographic database system has been equipped with a mechanism for downloading search results with Kermit.
- Washington University has a CICS-resident e-mail system that receives electronic mail messages from the PC via Kermit. Other forms of special-purpose uploads and downloads with Kermit acting as a ‘‘subroutine’’ are also in use.
- The library of the University of New Brunswick has an automated circulation system implemented under CICS. Each location has at least one PC signed on to the system. When the mainframe is down, transactions are recorded on the PC and uploaded and applied when the mainframe comes back up.
- Miami University has a network that collects data by day for nightly retrieval. The collection is automatically uploaded using Kermit-CICS.

3270 Protocol Converters

IBM 370 and compatible mainframes generally support both linemode (TTY) and fullscreen (3270) sessions. Both are marked by the need for communications front ends that variously translate between EBCDIC and ASCII and try to make ASCII terminals look like IBM terminals.

Linemode connections generally do not pose a serious problem for Kermit file transfer; Kermit needs only to undo the front end's ASCII/EBCDIC translation by performing reverse translations in each direction. Fullscreen connections, however, go through 3270 protocol converters that perform complicated functions like screen formatting and optimization that interfere with Kermit packets. Many, but not all, 3270 protocol converters offer a *transparent mode* that can be used to disable these functions, allowing Kermit packets to pass through without modification.

The welter of competing and often incompatible communications devices would cause a major headache for the poor Kermit user, except for three circumstances. First, Kermit-370 has routines to automatically detect which kind of front end is controlling the current session; second, the Kermit installer is encouraged to tailor Kermit to force the correct `SET CONTROLLER` default whenever those routines don't work properly; and, third, Kermit now offers a last-resort mode of operation that will work with protocol converters that do not include a transparent mode.

What Is Transparent Mode?

Transparent (sometimes known as ''graphics'') mode is a special type of operation in a protocol converter that passes the inbound and outbound data streams straight through (but modifying the parity in many cases), devised largely to allow interactive graphics on non-IBM graphics terminals.

Graphics applications need to transmit escape sequences or other control characters back and forth between the mainframe and the terminal, but a protocol converter normally filters out all control characters in both directions. In practice, the normal Kermit protocol needs just *one* transmittable control character for each direction to synchronize the encoding and decoding of packets.

Although protocol converters are advertised as simulating the behavior of IBM 3270-type terminals, they offer several different approaches to transparency. The device and Kermit must agree on the method; the SET CONTROLLER command can be used to force a particular style of transparent mode. Here are Kermit-370's SET CONTROLLER options, listing the devices they are known (or reported) to work with:

TTY
(Linemode) Amdahl 4705; IBM 37x5, 3708, or 8232; Comten 36xx; STNxx; Jupiter 1000; K200, K310, or K2000.

SERIES1
Yale ASCII system on IBM Series/1 or 4994; IBM 7171 or 937x ASCII subsystem; Hydra II; Commtex Cx-80; SIM3278/TCPIP 2.0 or /VM 5.0; tn3270; Cisco 516-CS.

GRAPHICS
Datalynx 3174 or 3274; Datastar 4025; Datastream/Leedata 8010, 8030, or 874; PCI 1076 or 276; Renex PCM, TMS-x, RPAD, or RTD; KMW S/II 3270.

AEA
IBM 3174 AEA (B2 or higher).

FULLSCREEN-Mode File Transfer

What if your 3270 protocol converter is not supported by one of the SET CONTROLLER commands listed above? Until now, Kermit file transfer through such devices was not possible.

Kermit-370's new FULLSCREEN mode has changed all that. The SET CONTROLLER FULLSCREEN command allows file transfer with no control characters at all and, therefore, without a transparent mode when used with a suitable transfer partner:

- It allows Kermit packets to start with a printable character rather than a control character.
- It frames the packet by its length field rather than depending on a ''line terminator.''
- It ensures there are no strings of repeated blanks in the Kermit packets.
- It ensures that the Kermit packet does not end with a blank.
- It restricts the packet length to fit within the screen width to prevent ''wraparound'' by the protocol converter.

Kermit programs that need to transfer files with Kermit-370 in fullscreen mode must be modified to account for these factors, and also must be prepared to ignore reflections of their own packets, which are echoed back once by the protocol converter and sometimes again by the operating system.

Kermit-370, MS-DOS Kermit 3.13, and C-Kermit 5A incorporate the needed modifications, and successful FULLSCREEN transfers are reported through pre-B2 IBM AEA controllers, the IBM 3708, Micom 7400, Sim3278/VTAM, and various tn3270 implementations, including those found in terminal servers from Xyplex and others.

Use FULLSCREEN mode only as a last resort: the requirement for short packets and the time needed to absorb multiply echoed packets reduce efficiency, and the lack of a unique packet synchronization character complicates error recovery procedures on noisy connections.

To set up a FULLSCREEN mode file transfer, issue the following commands:

Kermit-370	*C-Kermit or MS-DOS Kermit*
`SET CONTROLLER FULL`	
`SET RECEIVE START 62`	`SET SEND START 62`
`SET SEND START 62`	`SET RECEIVE START 62`
`SET BLOCK B`	`SET BLOCK B`
`SET HANDSHAKE 0`	`SET HANDSHAKE NONE`

This sets the packet-start character in both directions to be the greater-than sign (>) (ASCII 62) and enables a new block-check type (a 12-bit checksum containing no blanks) to defeat the trailing-blank-stripping feature found in many protocol converters. Short packets are used automatically.

Other New Kermit Releases

Acorn Archimedes Kermit

From Cosmos Nicolaou and Andrew Brooks, via Lancaster University in the UK, Kermit for the Acorn Archimedes with the RISC_OS operating system, a port of the ACW Panos version. May 1993. *Tape C.*

Apple II Kermit 3.87

From Ted Medin. New features include: screen restore on CONNECT for Apple IIe or later models; Kermit file attributes are now handled; percent of file transferred shown when possible (file attributes required); a new driver for the Ace dual card. Also, many bugs fixed, terminal emulation and printer support improved. December 1990. *Tape A.*

BTOS / CTOS Kermit 2.00

From Evan Arnerich and Doug Drury of ITT Federal Services Corporation, Santa Maria, CA: version 2.00 of CT-Kermit for the Burroughs B20/BTOS and Convergent NGEN/CTOS systems.

This new version adds some of the capabilities of MS-DOS Kermit 3.x and C-Kermit 5A, particularly script programming features (INPUT, OUTPUT, IF, ASK, GOTO, variables, etc), and includes a built-in VT101 terminal emulator. January 1993. *Tape C.*

Chinese DOS Kermit

An adaptation of MS-DOS Kermit 2.32/A to CC-DOS, the Chinese version of MS-DOS, also known as LIANXIAN, STCDOS, CCDOS213, and GWCDOS, written by Quanfang Zhang of Zhezhiang University, Hangzhou, China. See the article on page 19.

Commodore 64/128 Kermit

Updated to include support for the Swiftlink-232 serial interface by Kent Sullivan, Matthew Sorrels, and Ray Moody. September 1992. *Tape C.*

CP/M-80 Kermit 4.11

From Mike Freeman, Bonneville Power Administration, Vancouver, WA, USA.

Features added since version 4.09 (January 1988) include: new filename collision options: BACKUP, DISCARD, OVERWRITE, RENAME; an option to keep or discard incompletely received files; many new REMOTE commands for communicating with Kermit servers; a RENAME command; improved interruption of TAKE, TYPE, and PRINT commands in progress; many bug fixes.

Support was added for the Microbee family of computers (56K, 64K, 128K and 256K) manufactured by Microbee Systems, Ltd, of Australia, by Russell Lang of Monash University, Australia, and for the Ampro Little Board from Jay S. Rouman of Mt. Pleasant, MI, USA. April 1991. *Tape A.*

DEC PDP-8 and PDP-12

From Charles Lasner. Bug fixes and a new encoding format for bootstrapping this program to your PDP-8 or PDP-12. September 1990. *Tape D.*

Gould/SEL MPX Kermit 2.3

From Barry M. Wilson, Queensland Electricity Commision, Australia: a new Kermit server program for the Gould/SEL 32/77 computer with the MPX 1.5E operating system. It supports long packets and handles run-length compression in incoming packets. November 1990. *Tape D.*

Hewlett Packard 3000 MPE Kermit

From Tony Appelget, General Mills, Inc., Minneapolis, MN. New features of this version, written in the SPL language, include long packets, support for 16-bit CRC error checking, improved operation with HP Spectrum machines, a versatile command abbreviation scheme, a new HELP function, and many bug fixes. October 1991. *Tape D.*

A second version of the same program, but written in the C language, was received from Tony in June 1993. *Tape D.*

Honeywell DPS-6 Kermit 2.01

From Frank Dreano, Chesapeake, VA. New features of version 2.01 include wildcard sends; REMOTE commands for servers; ability to transfer foreign binary files both ways; bug fixes. June 1991. *Tape D.*

IBM CS-9000 Kermit

For the circa-1980 IBM laboratory workstation, a send-only Kermit program (so you can get your files off it) from Glenn Howes, University of Wisconsin, written in Pascal. September 1992. *Tape C.*

Luxor Computers

Furnished by Bo Kullmar, chairperson, ABC-Klubben, Stockholm, Sweden.

Protocol fixes for ABC80 and ABC800 from Jörgen Westman of ABC-Klubben. ABC800/802/806 CP/M systems updated to version 4.11, with support added for FACIT DTC and DTC2 Luxor clones, from Mikael Johansson of ABC-Klubben. July 1990. *Tape C.*

NCR 9800 VE/IVS and VE/MCS

From Paul E. Gladden of NCR Corporation, San Diego, California, USA: a new Kermit program for the NCR 9800-4 computer with the NCR VE4.0 operating system.

Separate variations are provided for the IVS and MCS environments. The program is written in C, based on C-Kermit 4E with features selected depending on VRX system capabilities. July 1990. *Tape D.*

Pecan Kermit 1.1

From R. Tim Coslet. Atari Mega ST2 under Pecan Software Systems UCSD p-System Version IV.2.2 with Standard File System (SFS).

Binary transfers now work in both SFS and AFS implementations. Incorrect reporting of file creation time in attribute packet fixed. July 1990. *Tape C.*

Prime Kermit 8.15

From John Horne Polytechnic South West, Plymouth, U.K., with contributions from Matthew Sutter: a new release of Prime Kermit that can initiate outbound connections. There is a new CONNECT command, with accompanying features for logging a terminal session, setting the transmission speed, duplex, and the CONNECT-mode escape character, plus a selection of CONNECT-mode escape functions.

Other new features include: ability to send BYE, FINISH, SEND, GET, and selected REMOTE commands to Kermit servers; script programming features, including new INPUT, OUTPUT, PAUSE, and CLEAR commands; problems with multiple file transfers with a specific file type are corrected.

There are also improvements in sliding windows and other Kermit file transfer protocol features; the exact file length is now sent in the attributes packet; an alternate initialization filename is specifiable on the command line; pound-sign conversion is correctly handled. April 1993. *Tape D.*

TurboDOS Kermit

From Mark Eichen at MIT. This one was written years ago, an adaptation of an earlier release of CP/M-80 Kermit. The source code was lost. The binary executable program is available in hex form. October 1992. *Tape C.*

UNISYS (Burroughs) A-Series Kermit 1.041

From Dave Squire, Computing Services, University of California at Davis. Long packets, alternate block checks, command files. July 1990. *Tape D.*

Kermit Distribution News

Max Evarts, Kermit Distribution
Columbia University, New York City

The Kermit Distribution and Support office has seen many changes since the last issue of *Kermit News*. In June 1990 Ken Suh moved on to law school and Andy Newcomb took his place. Many of you who have called us over the past few years already know Andy.

New Services

We can now accept credit card orders by phone. We can also accept credit card orders and purchase orders by fax. With our optional rush service, you can have your Kermit software within 24 hours! (Rush orders received after 4:00 PM New York time ship the next day.) See the order form for details.

In response to the growing demand for telephone service, we are putting in a call-processing system. If all lines are busy, you will have options to get the information you need from our voice menu or hold to speak to one of us. The system separates order-related calls and technical calls so those of you with a quick ordering question will not have to wait while we debug a software problem.

Technical Support Hints

Speaking of phones and Kermit problems, Andy and I now handle a substantial percentage of the technical support calls that come in. Here are a few pointers for those calling in for tech support:

- Make sure you have the current documentation for your Kermit software; if we are on the phone reading the manual to you, we cannot be helping someone who has a more difficult Kermit problem.
- Know the versions of all the Kermit programs involved before you call; most Kermit programs print the version number when they start up.
- Expect us to ask you to upgrade your Kermit software if you are far behind the times; we can only support up-to-date versions.
- Try to be in a position where you can reproduce your problem while you are on the phone.
- We do not support implementations of the Kermit file transfer protocol that are part of other communications packages.

Our technical support service is a free, but limited, resource. Usually, only one person is available at a time to handle tech support calls; please be considerate of the many other callers —help us to help as many Kermit users as we can.

The Truth about Kermit File Transfer Performance

Frank da Cruz

In the early 1980s, the first generation of Kermit software used the basic Kermit protocol: stop-and-wait exchange of short packets. The basic protocol is easily implemented and highly robust, and led to its rapid proliferation to hundreds of hardware and software platforms where it proved a success in transferring files under even the most difficult conditions.

The new generation of Kermit software improves on the original robust qualities and dramatically boosts performance without sacrificing compatibility with the earlier generation. Protocol capabilities are negotiated automatically so the newest, most advanced versions can still exchange files with the earliest or most minimal versions.

Kermit's performance features include long packets, sliding windows, control-prefixing selection, locking shifts, and compression. The first three have the potential for causing problems, and are not used unless you ask for them. This article describes Kermit's performance features and tests them against other popular protocols. The results might surprise you.

Long Packets and Sliding Windows

The maximum packet length in the basic Kermit protocol is 94, chosen to prevent data loss when the receiver has small input buffers or lacks an adequate method of flow control, and also to reduce vulnerability to noise. But since 1985, Kermit's long-packet extension has allowed packets up to 9024 bytes in length to be used when conditions permit.

Longer packets reduce the ratio of protocol overhead to actual data, increasing the potential file transfer efficiency (the ratio of file characters transferred per second to the actual connection speed) from 86% (for 94-byte packets) to 95% (with 2500-byte packets). When conditions deteriorate on the connection, the packet length is automatically adjusted.

Original, basic Kermit was a stop-and-wait protocol because it had to work on half-duplex as well as full-duplex connections. But connections through satellites or packet-switched networks can have delays that seriously impede the efficiency of a stop-and-wait packet protocol. For example, suppose packets are 90 bytes = 900 bits long, and there is a one-second transmission delay. For one packet and its response, the round-trip delay is 2 seconds. At 300 bits per second (bps), the 3 seconds required to transmit the packet plus the 2-second delay make 5 seconds, so throughput is 180 bps, 60% efficiency. At 9600 bps, it takes only 1/10 second to transmit the same packet, but the delay is still 2 seconds. Throughput is only 428 bps, 4.5% efficiency. When connections have delays, efficiency can be improved by lengthening the packets, but only if the connection is clean. On a noisy connection, longer packets are more likely to be damaged in transmission and take longer to retransmit.

On full-duplex connections, the new generation of Kermit software (IBM mainframe Kermit excluded, which always has a half-duplex connection) can transmit packets in a steady stream, processing the acknowledgements later as they arrive, thus eliminating the effects of transmission delays, and also eliminating the overhead of the acknowledgements themselves, since they are "on the wire" at the same time as the data packets and therefore don't take up any extra transmission time. This technique is called *sliding windows*, because multiple packets are kept in a buffer (window) that "slides" forward whenever the oldest packet in the window is acknowledged.

Using 94-byte packets without sliding windows on a connection that has a 1-second delay results (according to actual measurements) in an efficiency of about 8%. Raising the packet length to 1500 on the same connection increases the efficiency to 63%. Using sliding windows on the same connection raises the efficiency to 82–90%, depending on the packet length.

Optimum performance can be achieved on any given connection by choosing the right combination of packet length and window size.

To see a dramatic speed improvement using MS-DOS Kermit 3.13 and/or C-Kermit 5A, simply give these commands to each Kermit before file transfer:

```
SET WINDOW 3
SET RECEIVE PACKET-LENGTH 1000
```

Adjust as necessary. Longer delays require larger windows; noisier connections (or devices with small input buffers) need shorter packets. MS-DOS Kermit 3.13 and most versions of C-Kermit 5A allow the theoretical maximum sizes, 31 and 9024 respectively, sufficient to overcome any reasonable delay (for example, between the earth and the moon).

Compression

To reduce transmission overhead, the Kermit protocol uses a simple, but often surprisingly effective, compression technique: repeated byte values are represented by a count+byte combination. This technique is easy to program and inexpensive in memory and CPU cycles, and is therefore implemented in most Kermit software, including MS-DOS Kermit, C-Kermit, and IBM mainframe Kermit, and is used automatically when available.

Analysis of large volumes of both textual and binary data shows an average compression of 15–20%. Dramatic savings are achieved in certain types of files, including tabular text (or any other type of text with lots of repeated characters) and executable programs containing large amounts of pre-zeroed data.

Prefixing

To achieve its ability to push data through even the most restrictive types of connections — for example, to mainframes that are sensitive to certain control characters, or on 7-bit connections, or on very noisy ones (one user said recently, ''Kermit will send data over a communication channel that is only slightly better than a pair of tin cans connected with a wet string'') — Kermit formats its packets as lines of printable text. This is done by encoding each control character as a sequence of two printable characters and, *on 7-bit connections only*, encoding 8-bit characters as a sequence of two printable 7-bit bytes.

On some connections it is safe to transmit certain control characters ''bare,'' without prefixing or encoding. ''Unprefixing'' of control characters can speed up the transfer of binary files, particularly precompressed files, which tend to contain a lot of bytes in the control range. MS-DOS Kermit 3.13 and C-Kermit 5A(189) give you the ability to specify which control characters are to be prefixed and which are not. In the benchmarks on pages 12 and 13, only three control characters are prefixed:

```
SET CONTROL UNPREFIXED ALL
SET CONTROL PREFIXED 0 1 129
```

This technique can be used even if the Kermit program on the other end doesn't know anything about it, since well-designed Kermit software will, indeed, accept bare control characters literally. The three exceptions above are NUL (0), which is used internally by C-Kermit for string termination, and SOH (1) and SOH+parity (129), Kermit's normal packet-start indicator. It takes some experimentation to find the maximum safe set. That's why Kermit prefixes all control characters by default: *first make it work, then make it fast*.

On 8-bit connections, Kermit transfers 8-bit data with *no additional overhead*. On 7-bit connections, which are quite common — these are the connections that use even, odd, mark, or space parity, often without the user's knowledge — 8-bit data is encoded using a single-shift technique, a prefix character for each byte whose 8th bit is 1, similar to holding down the Shift key on your keyboard for each 8-bit character. *This allows Kermit to work where most other protocols fail.* The amount of prefixing ranges from 0% up to 100%, depending on the type of file.

> *The Kermit protocol implementations found in many of the popular commercial and shareware PC communication software packages are minimal and perfunctory, usually lacking some or all of the performance features . . .*

Locking Shifts

To avoid the high overhead of transferring 8-bit text, particulary Cyrillic, Hebrew, or Kanji, on 7-bit connections, a new ''locking-shift'' feature works like the Caps Lock key on your PC: a special shift prefix applies to a entire run of 8-bit characters, no matter how long, rather than to a single character. *Locking shifts are used in combination with single shifts to achieve the most compact encoding.*

Locking shifts are supported by MS-DOS Kermit 3.13, C-Kermit 5A, and IBM Mainframe Kermit 4.2.4, and are negotiated automatically when parity is in use (including when parity is detected automatically). They reduce the 8th-bit prefixing penalty anywhere from 0% to 100%, depending on the groupings of the 8-bit characters within the file.

So Why the Bad Reputation?

The Kermit protocol implementations found in many of the popular commercial and shareware PC communication software packages are minimal and perfunctory, usually lacking some or all of the performance features just described. Many of these same packages also include XMODEM, YMODEM, or ZMODEM protocol, which (when they work at all) usually perform better than the *basic* short-packet, stop-and-wait, prefix-everything Kermit protocol. Using a limited Kermit implementation is like filling your bathtup from a dripping faucet instead of turning the water on full blast. It is easy to see why users of such packages might conclude that Kermit file transfers are slow. Nothing could be further from truth; turn the page and see for yourself.

True-Life Benchmarks

Table 1 illustrates the performance of the Kermit protocol implementations found in different PC software packages. These measurements were made on a direct 19200-bps serial connection, downloading a typical ASCII text file (the VM/CMS Kermit-370 manual), 135087 bytes in length, from a Sun SPARCserver-10 with C-Kermit 5A(189) to the hard disk of an IBM PS/2 Model 70.

Table 1 *Kermit Implementations Compared*

PC Software	*Window Size*	*Packet Length*	*Time (secs)*	*Speed (cps)*	*Efficiency*	*Remarks*
Telix	1	94	131	1052	55%	Long packets and sliding windows not available
MTEZ	1	94	119	1158	60%	Long packets and sliding windows not available
Smartcom III	1	94	113	1220	64%	Long packets and sliding windows not available
PROCOMM PLUS	14	1000	77	1790	93%	Window size not selectable
Zstem 340	2	1000	74	1863	97%	Maximum window size is 2
MS-DOS Kermit	3	1000	72	1915	99%	Full control-character prefixing
MS-DOS Kermit	3	1000	69	1999	104%	Only 0, 1, and 129 prefixed

The results speak for themselves.

> ***If you thought Kermit file transfer was slow, you were probably not using real Kermit software!***

The UNIX-resident copy of the file, like all UNIX text files, uses only linefeed (LF) for line termination. During text-mode transfer, each LF becomes carriage return and linefeed (CRLF). There are 2814 lines in the file, so the actual data size during (and after) transfer is 137901. Since the connection runs at 1920 characters per second (10 bits per character), a 100%-efficiency transfer should take 137901 / 1920 = 71.8 seconds. The following PC communications software was used:

MS-DOS Kermit 3.13	Columbia University, New York, NY, USA
MTEZ 1.16	MagicSoft, Inc., Lombard, IL, USA
PROCOMM PLUS 2.0	Datastorm Technologies, Inc., Columbia, MO, USA
Smartcom III 1.0A	Hayes Microcomputer Products, Inc, Norcross, GA, USA
Telix 3.21	deltaComm Development, Cary, NC, USA
Zstem 340 1.0.4	KEA Systems Ltd., Burnaby, BC, Canada

Kermit and X-Y-ZMODEM

XMODEM, YMODEM, and ZMODEM are the file tranfer protocols most commonly compared with Kermit, and which are found in numerous shareware and commercial communication software packages. XMODEM and YMODEM are stop-and-wait protocols; XMODEM uses short blocks (128 data bytes), YMODEM uses longer ones (1024 data bytes). ZMODEM is a streaming protocol.

The tables on page 13 compare XMODEM, YMODEM, ZMODEM, and Kermit transfers between the PC and UNIX. The file transfer software on the UNIX system is sx (XMODEM) / sb (YMODEM) / sz (ZMODEM) 3.24 (June 1993) and C-Kermit 5A(189). On the PC, X-, Y- and ZMODEM transfers were done with Telix and PROCOMM PLUS (which gave exactly the same results). For fairness, four types of files are transferred:

ASCII Text:	`IKCKER.DOC`	137901 bytes	Our original ASCII text file, text mode
UNIX Binary:	`uuencode`	24576 bytes	A Sun SPARC binary executable program, binary mode
PC Binary:	`KERMIT.EXE`	197928 bytes	An MS-DOS binary executable program, binary mode
Precompressed:	`KERMIT.ZIP`	238584 bytes	A compressed ZIP archive,binary mode

Tests were performed on four types of connections and in each trial, Kermit transfers used a window size of 5 and a packet length of 5000, and control prefixing was disabled except for NUL (0), Ctrl-A (1), and 129. As the tables show, **Kermit outperforms the competition every time.**

Table 2 shows the figures for transferring all four files with each of the four protocols on same direct connection used for Table 1. In this and the following tables, the *secs* column shows the elapsed time of transfer in seconds, the *cps* column shows actual file characters transferred per second, and the *eff* column shows the percent efficiency (file characters per second divided by the connection speed).

Table 2 *X- Y- and ZMODEM vs Kermit on a 19200-bps Direct Connection*

	XMODEM			*YMODEM*			*ZMODEM*			*KERMIT*		
File Type	*secs*	*cps*	*eff*	*secs*	*cps*	*eff*	*secs*	*cps*	*eff*	*secs*	*cps*	*eff*
ASCII Text	89	1549	81%	76	1814	95%	73	1889	98%	69	1999	104%
UNIX Binary	15	1638	85%	13	1890	98%	13	1890	98%	9	2648	138%
PC Binary	127	1558	81%	109	1816	95%	107	1850	96%	100	1979	103%
Precompressed	153	1559	81%	133	1794	93%	130	1835	96%	129	1849	96%

Table 3 shows the results for a local-call dialup connection using Telebit T3000 modems, V.32*bis* modulation (14400 bps), V.42 error correction, V.42*bis* compression, RTS/CTS hardware flow control, and an interface speed of 57600 bps. The efficiencies in this table are based on the modem's 14400-bps connection speed, and therefore also reflect the modem's compression methods.

Table 3 *X- Y- and ZMODEM vs Kermit with High-Speed Modems*

	XMODEM			*YMODEM*			*ZMODEM*			*KERMIT*		
File Type	*secs*	*cps*	*eff*	*secs*	*cps*	*eff*	*secs*	*cps*	*eff*	*secs*	*cps*	*eff*
ASCII Text	221	624	43%	79	1746	121%	42	3283	228%	39	3535	246%
UNIX Binary	32	768	53%	13	1890	131%	15	1638	114%	3	8192	569%
PC Binary	346	572	40%	129	1534	106%	83	2385	166%	80	2474	172%
Precompressed	500	477	33%	208	1147	79%	149	1601	111%	148	1612	112%

So far we've looked only at connections with no delays. Table 4 (*also see cover, left group*) shows the results for a V.32 9600-bps cross-country dialup connection from the same PC to a PC/486-50 running UNIX System V R4, with the same C-Kermit, sx, sb, and sz software as on the Sun. The round-trip delay is a fraction of a second. No error correction or compression is done by the modems, but the connection is clean and no errors occurred.

Table 4 *X- Y- and ZMODEM vs Kermit with Delays at 9600 bps*

	XMODEM			*YMODEM*			*ZMODEM*			*KERMIT*		
File Type	*secs*	*cps*	*eff*	*secs*	*cps*	*eff*	*secs*	*cps*	*eff*	*secs*	*cps*	*eff*
ASCII Text	422	327	33%	253	545	57%	217	635	66%	151	913	95%
UNIX Binary	73	337	35%	41	599	62%	32	768	80%	8	3072	320%
PC Binary	536	369	38%	319	620	65%	271	730	76%	207	956	99%
Precompressed	710	336	37%	363	657	68%	314	759	79%	284	840	87%

But if we always had clean connections, why would we need error-correcting file-transfer protocols? Table 5 (*and middle group, cover*) shows the results for the same cross-country connection, same settings, but with short bursts of noise injected every two seconds, which cause errors and retransmissions in all four protocols.

Table 5 *X- Y- and ZMODEM vs Kermit with Delays and Noise at 9600 bps*

	XMODEM			*YMODEM*			*ZMODEM*			*KERMIT*		
File Type	*secs*	*cps*	*eff*	*secs*	*cps*	*eff*	*secs*	*cps*	*eff*	*secs*	*cps*	*eff*
ASCII Text	3346	41	4%	*fail*	0	0%	438	315	33%	206	669	70%
UNIX Binary	573	43	4%	58	424	44%	144	171	18%	9	2736	284%
PC Binary	5154	42	4%	*fail*	0	0%	566	350	36%	281	706	74%
Precompressed	5917	40	4%	*fail*	0	0%	694	344	36%	385	621	65%

What about 7-Bit Connections? No Contest!

The foregoing benchmarks were conducted in environments where XMODEM, YMODEM, and ZMODEM can work, namely 8-bit transparent connections that are not sensitive to any control characters. Now let's look a different, but very common, situation. Table 6 (*and right group, cover*) shows the results of downloading the same files from an IBM Mainframe running VM/CMS and Kermit-370 4.2.5 to the PS/2 over a 19200-bps serial connection through an IBM 7171 protocol converter, which uses even parity and Xon/Xoff flow control. Kermit's window size is 1 because the mainframe can operate only in half duplex, and the packet length is 1920, the largest allowed by the 7171. All control characters are prefixed.

Table 6 *File Transfer on a 7-Bit Connection*

	XMODEM			*YMODEM*			*ZMODEM*			*KERMIT*		
File Type	*secs*	*cps*	*eff*	*secs*	*cps*	*eff*	*secs*	*cps*	*eff*	*secs*	*cps*	*eff*
ASCII Text	-	0	0%	-	0	0%	-	0	0%	81	1702	88%
UNIX Binary	-	0	0%	-	0	0%	-	0	0%	9	2730	142%
PC Binary	-	0	0%	-	0	0%	-	0	0%	162	1221	63%
Precompressed	-	0	0%	-	0	0%	-	0	0%	243	981	51%

The table shows Kermit file transfer to be infinitely more efficient than X-Y-Z-MODEM transfer with IBM mainframes, because X-Y-Z-MODEM implementations *do not work* with IBM mainframe operating systems such as VM/CMS, MVS/TSO, or CICS, whereas Kermit works with all of them. Of course, 7-bit connections are not peculiar to IBM mainframes. They are also used by other types of mainframes and front ends as well as many types of networks and devices, including some X.25-based public data networks and certain terminal servers. You can use Kermit to transfer files on these connections, but not X-Y-Z-MODEM protocols.

Locking Shifts

Although Kermit, unlike X-Y-Z-MODEM, can transfer 8-bit data over 7-bit connections, there is often a performance penalty. This penalty is particularly unfair to people whose written languages are encoded primarily in 8-bit characters, as are Russian, Hebrew, and Japanese. Russian text encoded in any of the commonly used 8-bit Cyrillic character sets typically consists of about 80% 8-bit characters and Japanese Kanji text often consists of nearly 100% 8-bit characters.

Table 7 shows the results of attempting to upload typical Russian and Japanese 8-bit text files over a 19200-bps 7-bit serial connection to an IBM mainframe using X-Y-Z-MODEM (it can't be done), Kermit with only single shifts (SS), and Kermit with locking shifts (LS). The Kermit window size is 1 and the packet length is 1920. In these cases, locking shifts improve the speed of transfer 30–40%.

Table 7 *Effect of Locking Shifts*

		X-Y-Z-MODEM			*KERMIT (SS)*			*KERMIT (LS)*		
File Type	*Size*	*secs*	*cps*	*eff*	*secs*	*cps*	*eff*	*secs*	*cps*	*eff*
Russian Text	52046	-	0	0%	55	946	49%	39	1334	69%
Japanese Text	29706	-	0	0%	34	873	45%	20	1485	77%

Conclusion

Kermit protocol works in practically every communication and computing environment. You don't have to be a data communications expert to transfer a file with Kermit software. Its first priority is getting your data through safe and sound, and its second is efficiency. Kermit's conservative protocol defaults reflect these priorities: *First make it work, then make it fast.* But as the tests show, today's Kermit software, when given several simple commands to enable its efficiency features, outperforms X-, Y-, and ZMODEM protocol transfers every time. And *real* Kermit software also outperforms the Kermit protocol implementations found in commercial and shareware communications programs. Skeptical? Run your own tests!

Kermit and the British Relief Mission to Bosnia

Lieutenant Colonel John F. J. Allen, MBE
Royal Logistic Corps, UK Army, Andover, Hampshire, UK

We hear daily of the huge amounts of food and aid brought into the besieged areas of the former Yugoslavia, and we have a successful system, proven in an operational environment, that has now come of age. Further developments will see direct satellite communication to and from relief convoy escort vehicles, and integrated information and communications systems —and at their heart lies Kermit.

Browsing through our software library last autumn in search of inspiration, I chanced to stumble across an early version of MS-DOS Kermit that recalled a passage I had read on Kermit a couple of years earlier in a communications textbook. I had the germ of an idea and, a few trans-Atlantic phone calls later, I found myself in New York for my first visit to the United States in late October 1992, about to meet the Kermit team at Columbia University. *But to put the tale in fuller perspective* . . .

Once the political decision was taken for British participation in the United Nations relief mission to the former Yugoslavia, preparations for deployment began in earnest. For the Army Logistic (G4) area, this meant the responsibility for supply and equipment management support to the United Kingdom force deployed in Bosnia.

Accountability and total asset visibility of consumables, stores, and spares would be vital to the UN mission in the relief areas. The requirement became evident: to establish a system of adequate controls for asset tracking and in-transit visibility from point of despatch in the base area to the receiving distribution point in theatre —a logistic-support asset-tracking system. With such a plethora of commercial systems available, neither the hardware or software solutions were insurmountable. But selection of communications and file transfer protocols would require careful consideration.

No current computer system specifically addressed storage and distribution commodity tracking, although progress was being made in a number of related areas that would integrate in later years to provide in-transit visibility of assets. Provision was made within the ICL 3900 Series mainframe-based Stores System accessed through an online information system, and other operational and administrative information systems, to link demands to issues, and track containers in the logistic pipeline.

No further visibility was currently available beyond this. Future fourth-generation information systems were also under development, but those components that would link issues to freight consignments to support asset tracking were not to be in service in the immediate future.

Our aim was therefore to produce an overarching mantle system in support of humanitarian missions in Bosnia, based upon a central data repository with information access points across the supply and distribution chain. The system would address the requirement of the British contingent on United Nations relief work, presenting visibility of items from source to destination. It would achieve the resupply operation as economically as possible and with a more effective, efficient method of control.

The system presupposed a central distribution point in the Theatre of Operations and the despatch of items for the relief operation to the forward areas in the former Yugoslavia from single points of departure in the Base or Forward Depots in the United Kingdom and Germany.

The asset tracking system would require a database using data captured at issuing depots and at various points in the distribution chain. Data on commodity visibility would then be drawn from the system, through communications links at the source, transit points and destination, with the ability to generate reports and produce meaningful data on location, content, quantity and status.

The solution, within the limitations of the time permitted, was found by developing an operational prototype, followed by a two-phase development, from initial research and development undertaken within the aegis of a peripheral peacetime project. Project VITAL (Visibility In Transit Asset Logging), was therefore highjacked and harnessed to our needs in support of the UN operation.

Procedures were put in place to enable information gathering and to advance and modify relevant aspects in the prototype development to satisfy the urgent operational requirement. At this stage, *after comparison with proprietary commercial software, Kermit was identified as the proper solution to our file transfer, network protocol, and terminal emulation needs, linking the entire spectrum of the project operation, from mainframe, minicomputer, PCs, to handheld devices and barcode readers.*

Once the decision was made to proceed, funding was negotiated and development proceeded in two distinct phases: *(1)* the basic system that can be put in place quickly, and *(2)* aspects of the system that would require more investigation and analysis.

PC access points to the system, using IBM-compatible 386 SX PCs with printers and necessary software, were installed at multiple locations along the supply and distribution chain to the forward bases that had been established in Bosnia.

The profile of the system envisaged a central computerised repository of commodity and transit data, hosted on a UNIX-based ICL DRS 6000 machine, drawn from existing logistic information systems, this information being accessed and supplemented at nodal points along the logistic pipeline, and at operational or logistic Command and Control (C^2) Headquarters, using Kermit and data communication links to provide the required visibility.

The heart of the system is a relational database, situated at the Directorate of Logistic Information Systems, in the county of Oxfordshire, that draws information from the Stores System mainframe, and data from the VITAL input devices.

The system is linked on a network by line or Hayes-compatible modems through national and international ISDN telephone and through INMARSAT-C satellite communications, through the British Isles commercial communication hub in Cornwall in the South West of England. Each access point has a PC, enabling each station to interface to the data repository through the network.

Handheld Tandy-Grid (US model 2350) electronic palm-pad data input devices with inductive pen contact and character recognition on touch-sensitive screens, again using MS-DOS Kermit loaded on SUN RAM disks, are in use along the logistic pipeline, allowing electronic download of data through the PCs, or direct over the communications links, to the central database, and data retrieval, screen and report printing at each access point.

Verification is carried out on-line with Kermit file transfer and terminal emulation, to the stores system to complete the loop on the status of the commodities. The system is able to operate in either direction.

The system tracks the progress of commodities, as single items or as part of larger consignments, along the pipeline, by air, road, rail, or sea routes, and may also be applied to postal despatch of items. Items are identified through a designator code, to combine with transit information, to allow visibility of progression along the pipeline through to the troops deployed on relief convoy work.

The development software to support the immediate urgent operational requirement was in service during November 1992, with Phase 1 work complete in December 1992. Communications links and user software were successfully tested over line and satellite and the prototype system went live in December 1992, with Phase 2 completion due in 1993. Links for air freight are being developed and have been established with the Royal Air Force (RAF) and civil airline air cargo systems.

We hear daily of the huge amounts of food and aid brought into the besieged areas of the former Yugoslavia, and we have a successful system, proven in an operational environment, that has now come of age. Further developments will see direct satellite communication to and from relief convoy escort vehicles, and integrated information and communications systems — and at their heart lies Kermit.

The verdict: *We have been most impressed with Kermit in all its forms, especially with the support, versatility, the ease of use, and lack of problems.* We were not previously aware of its potential and capabilities, but now we are adapting it in several other systems and extending its use amongst diverse information areas.

A very big thank you to the Kermit team at Columbia University in New York City, who most generously cooperated on the project, giving consultancy during a whistle-stop visit to New York, including rapid development of prototype scripts for automated connection establishment, authentication, and data transfer, and their continued help desk support and further software upgrades.

The author, Lieutenant Colonel John F. J. Allen, MBE, is a career officer in the UK Army in the newly formed Royal Logistic Corps, responsible for Logistic Support Information Systems Policy and Strategy currently serving at the Ministry of Defence Logistic Headquarters in the South of England.

Kermit in Germany

Gisbert W. Selke
Ermekeilstraße 28
D-5300 Bonn 1, Germany

Over the past several years, three major events have happened to Germany:

- Re-unification of East and West,
- the advent of umlaut-preserving file transfer,
- and a German MS-DOS Kermit book.

While there is no direct causal connection between any two of these, they are not altogether unrelated either; so let's look at each of them in turn.

First, re-unification. After nearly 50 years, the Iron Curtain that separated the Western 75% of the population from the Eastern 25% has at last been taken down. Everyone will have read in the papers about this, so we won't go into the details here; just let me say that it feels great to be able to see my relatives drüben (over there) whenever I want.

Apart from my personal feelings, however, there is one aspect that has, in fact, to do with computing and, more specifically, with Kermit: the computer market, both private and business, is a hot spot in the region commonly referred to as the ''5 neue Länder'' (or 5NL, for short, although Americans might prefer the colloquial ''Neufünfland'').

And so is the telecommunications market. While telephones have been hard to come by previously, the German PTT is bustling to bring the telephone net up to standard, for which there is an enormous need.

Together, these factors have created high demand for computer communications. The 5NL universities have joined the Internet, and private mailboxes (BBS's) are mushrooming. For many purposes, telefax is the service used heavily between the two parts of Germany, but there are also many companies that have to exchange data between their Western head offices and their Eastern branches (note the asymmetry here!).

Since currently the Eastern phone net is still in deplorable shape in many places, a reliable, yet fast file transfer protocol is needed. Of course, it should be easily adjustable to take advantage of improving conditions; it should be able to handle those funny characters (like ä and ß) that Germans seem to like so much; and implementations should be easy to handle, since you wouldn't want to employ a computer scientist just for this purpose.

Sounds familiar? Yes, there we are: Kermit fits the bill nicely. So, somewhat unexpectedly, we have a whole new market for Kermit software. And this extends to other parts of Eastern Europe as well: since I'm giving a hand with Kermit promulgation on this side of the Atlantic, I have noticed a considerable demand for Kermit software from Poland, Czechoslovakia, and even as far as Kazakhstan, when it still was a part of the Soviet Union.

Kermit and German Text

For decades, computers have been made *by* English-speaking people *for* English-speaking people. But German, like many other languages, has special characters that do not fit into standard 7-bit ASCII code (which, after all, is the *American* Standard Code for Information Interchange).

So, various manufacturers have looked for ways to circumvent this. One way of doing this – with the ISO's blessings – was to scrap the braces, brackets and so forth and use their character positions for the umlaute. This was widely accepted; but what if you needed those braces?

Your beautiful C programme, when printed on a germanicized printer, might look like this:

```
if ((a==1) öö (a==9)) ä
     printf("Grüße aus KölnÖn");
ü
```

However, if you switched the printer back to plain ASCII, your programme would look like this:

```
if ((a==1) || (a==9)) {
    printf("Gr}~e aus K|ln\n");
}
```

Not what you intended, either . . . And no way around it. With the advent of the IBM-compatible PC, a different standard emerged, which at least preserved normal ASCII as well as many European special characters. The Macintosh, of course, was different. And of late, Windows has yet another conception of the special characters. You are not, of course, surprised to hear that MS-Windows NT is almost entirely unlike the others.

As time went by, we gained some proficiency in deciphering on the fly. Depending on the machine you'd work with, you'd know what key (or sequence of keys) to type to get the desired letter. What, however, if you had to exchange files between different platforms? Let me recall one early day in the WIdO (Wissenschaftliches Institut der Ortskrankenkassen), where we had been running a Modcomp MAX IV as a host computer for years, and the first PCs arrived and were wired up as terminals over the serial line. Boy, were we happy to have found MS-

DOS Kermit 2.28 to transfer files in the first place! But then we sent an ordinary (or so it seemed) text file from a PC to the host. For the greater part, all went well; but some characters were missing, and strange runs of repeated characters could be found in places.

The explanation turned out to be simple: characters with their eighth bit set were a special MAX code used for a simple run-length encoding. Annoying, yes; and although it was easy to write a programme to convert the umlaute to braces and brackets, Murphy's Law tells us that you would forget to use this filter on none but the largest files . . .

Today, all this is gone. Using MS-DOS Kermit, I can easily configure my ''terminal'' to display braces as umlaute: SET TERM CHAR GERMAN! Or, to show them as braces, SET TERM CHAR LATIN1, as would be used with a host employing an ISO 8859-1 character set (which, incidentally, is also used by MS-Windows). Our secretaries no longer have to remember to hit '[' when they want 'Ä' to be printed — some progress! Or, when I connect to an IBM VM/CMS mainframe (an EBCDIC machine), accidentally hitting one of the umlaute on my PC keyboard tended to wreak havoc on the connection. Nowadays, I have a few commands like:

```
SET KEY Ä Ae
```

in a special TAKE file, and whenever I hit the Ä key, ''Ae'' is sent instead (where Ae is the standard transliteration for Ä, from auld lang pre-computer syne).

File transfer, too, is no longer a problem. On our UNIX host, we use C-Kermit; sending a file to the MAX host, we can SET FILE CHAR LATIN1, SET TRANSFER CHAR ASCII, and SET LANGUAGE GERMAN, and all the umlaute are converted automatically to braces, etc., on the fly. As I write this, the old MAX IV host is being taken down and replaced by a MAX 32; and here's another advantage for us: no more serial links at 9600 or 19200 bits per second, for now we've got an Ethernet! And, surprise, Kermit supports LANs, too; no need to change to any other Telnet terminal emulation programme, no need to give up Kermit's speed, easy configurability and robustness.

There was even a story about a user who wanted to convert a German text file from PC standard to the newer Windows standard. So he started Windows, ran MS-DOS Kermit in two different windows, hooked up COM1 and COM2 with a null-modem cable, SET this, SET that, and off he went, communicating with himself, so to speak, but transliterating the file in the process. This story was related at a Rhineland Karnival session, so take it for what it's worth . . .

Kermit auf Deutsch

Well, there we are, with Kermit — a programme to cater to all the Germans' communication needs. There was only one problem remaining: the only Kermit documentation available was in English. Not too much of an obstacle for your local hacker . . . But it's sometimes good to remember that there are ordinary people, too, to whom a computer and a modem are just tools and who don't want to have to learn a foreign language just to use these tools. (Conversely, it is sometimes worth noting that there are people who can actually type and not just chase a rodent around on their desktop. But I digress.)

Fortunately, Chris Gianone had written an excellent book on MS-DOS Kermit; not just a manual, but a gentle introduction into terminal emulation and file transfer. While she was taking this book to its second edition, covering all the new features since MS-DOS Kermit 3.0, she was looking for foreign-language publishers as well. And we made it — a publisher known for high-quality magazines and books on all aspects of computers and electronics proved to be interested, indeed. Nearly in parallel to Chris's writing the second edition, I prepared the German translation, so the German book appeared on the market barely three weeks later than the American original. The representative of the American publisher couldn't quite conceal his astonishment when presented with the German translation at the October 1991 Frankfurt Book Fair, and our German editor took some pride in this — rightly so! By the way, all the queries and last-minute corrections between Chris and myself were done using Kermit software (making Kermit a recursive application?).

At 69 marks, *MS-DOS-Kermit — das universelle Kommunikationsprogramm* includes the official Kermit distribution disk with all the text files translated into German. To my knowledge, this is the first book on the German market to cover computer communications at this scope, and the sales figures show that it fills a need: it is selling well, so get your copy before they are sold out! — OK, don't panic... the second printing has just hit town. (Which shows that the book is more successful than had been anticipated by the publishers themselves.)

Hold the presses! Here we go again: Frank da Cruz and Chris Gianone have collaborated on a magnificent book on C-Kermit. C-Kermit runs on an amazing variety of machines whose least common denominator is just the existence of a C compiler. It's a natural for all those UNIX machines, of course; but it also runs on . . . no, wait, I'm not going to bore you with a list several pages long. Why not browse your friendly local book-seller's shelves? You say you're living in Austria (the one without kangaroos) and you don't exactly fancy manuals in English? No problem: the German translation is underway right now, and you'll be able to pick it up at the Frankfurt Book Fair in early October. And, lest I forget: if, by ''manual'', you mean ''dreary and unreadable,'' you're dead wrong. Were it otherwise, I wouldn't have translated it. I can't stand boring books.

Let me mention a final point that is often overlooked. Kermit software has the greatest user support I have ever seen. This shows in fast response to (even minor) complaints, in lots of care spent on the fine points, and, of course, in the concern given to the non-standard user. Among these, I count the non-English speakers, but also those with visual, auditory or physical challenges. This concern does not go without saying in today's computer market; and, speaking for the non-English people at least, I'd like to say thanks to Frank da Cruz (who started it all, and who spends a lot of time on C-Kermit), to Chris Gianone (who keeps it all running and whose book is terrific) and to Joe Doupnik, whose programming skills I won't even mention, but whose wit and understanding have proven invaluable over all those years since I first got in touch with MS-DOS Kermit version 2.28.

Kermit in China

Quanfang Zhang and Jijiao Zheng
Department of Computer Science and Engineering
Zhejiang University, Hangzhou, China

Today, Kermit has spread all over the world and has been implemented in many computer systems. Quietly, it arrived in China and was adapted to many Chinese-version operating systems. You can see more and more computer specialists and users using Kermit to transfer files or connect a local computer to a remote host in Chinese universities or institutes. Kermit is now a popular topic for discussion among people who are engaged in computer communications.

As teachers in the Department of Computer Science and Engineering of Zhejiang University, we are very interested in computer networks and communications. We began to study Kermit at the end of 1988, when our Computer Network Research Laboratory was entrusted with designing the Zhejiang University campus computer network (ZUnet). In the first stage of development, we planned a network system based on a PBX. We found it very difficult to design such a network because many computers, distributed in all departments and administrative offices and running many different operating systems, would be connected by ZUnet.

Fortunately, we found the article written by Frank da Cruz and Bill Catchings in *BYTE* magzine and realized that Kermit was just what we were looking for! After receiving tapes containing all Kermit programs and documents from Columbia University, we wrote I/O driver routines for the Data/Voice integrated communication adapter, a powerful network card used in ZUnet that handles data and voice simultaneously. Then we modified MS-DOS Kermit 2.32/A and C-Kermit 4E to make them run on ZUnet. Because CC-DOS (the Chinese version of DOS used in IBM-PC and its compatibles) is widely used in ZUnet, we also modified MS-DOS Kermit 2.32/A for Chinese DOS and named it CC-Kermit 2.32/A *[On Tape C]*.

Kermit and Chinese DOS

In China, the most popular microcomputers are IBM PCs and compatibles. CC-DOS (or CDOS) is a Chinese DOS for PCs, which has many different versions; basically it is MS-DOS with Chinese character I/O processing modules. Most application management systems run under CC-DOS. From the user's point of view, characters are displayed on the screen in character mode, but from the system's point of view, Chinese characters are actually displayed in graphic mode. In general, a Chinese character code is represented by two bytes. The IBM PC version of MS-DOS Kermit can't run correctly on CC-DOS because it accesses video RAM directly.

We converted MS-DOS Kermit 2.32/A to CC-Kermit 2.32/A, a Chinese Kermit, which can run on MS-DOS and most versions of CC-DOS. Explanations and prompt messages are displayed in Chinese when it runs on CC-DOS. This is very important for the popularization of Kermit in China, as many users do not learn much English. When it runs on MS-DOS, CC-Kermit is the same as MS-DOS Kermit 2.32/A.

Chinese character codes are defined by *Chinese Character Set for Communication and its Exchange Codes,* GB2312-80. The size or resolution of a Chinese character displayed on the screen can vary. There are 16×16, 24×24, 32×32 and 48×48 dot matrices, depending on the CC-DOS and graphic adapter versions. We modified the display and keyboard input modes of Kermit for various display sizes, such as 11, 17, or 25 lines per screen.

Kermit at Zhejiang University

At Zhejiang University, there are more than 1000 computers including microcomputers (IBM PC, Apple II/III, etc.), minicomputers (MicroVAX II/III, VAX 11/785, PDP-11/23) and two mainframes in the Computing Center (Honeywell DPS8 and IBM 4361). There are also a lot of HP and Sun graphics workstations. *Teachers and students used to find file transfer between two machines annoying. Now Kermit makes it simple.*

Kermit plays an important role in ZUnet. There are over 60 computers in ZUnet distributed in 12 buildings. ZUnet makes use of the existing telephone system; each computer is equipped a Data and Voice integrated adapter with data transfer at speeds up to 19200 bps. ZUnet is a low-cost but useful system. It provides file transfer, eletronic mail, database retrieval, terminal emulation, and remote job entry.

Users can exchange e-mail internally and with CAnet (China Academic Network), and we are connected by Kermit to a host in Beijing for international electronic mail. Computer specialists consider ZUnet an economical, useful, and efficient campus network, well-suited to universities and institutes of China.

Kermit in China

Kermit is widely used not only at Zhejiang University but also in many other places in China. Users scattered in different districts in Zhejiang Province use Kermit to connect to the host at the Scientific and Technological Information Institute of Zhejiang Province to retrieve databases. In ShengLi Oil Field, the second largest in China, a whole oil-field information management network system is being built. The system is composed of a lot of computers distributed in the Oil Field headquarters, oil extraction factories, oil production teams, and well drilling teams. Kermit protocol and software are used in the large systems.

Papers about Kermit have been published in many Chinese computer magazines. including an article by us, *Kermit Protocol and its Programs*, presenting the background, development, running environment, functions, and protocol of Kermit, published in Chinese *Data Communications*, No.2, 1991.

Kermit has already played an important part in China, especially in Zhejiang University. We are sure that it will be recognized by more and more computer users and become their good friend. *Kermit makes complicated things simpler and longer distances shorter.* We hope to make and keep contact with other Kermit developers and work together for the development and populization of Kermit.

We would like to express our sincere thanks to Christine M. Gianone and Frank da Cruz for their protracted support and guidance to us. Without their help, our ZUnet could not have been put into working order in such a short time.

Report from London: Kermit in Medical Research

Robert Clark
Joint Computing Unit, Institute of Neurology, University of London, Queen Square, London

The Institute of Neurology is a postgraduate medical research Institute of the University of London, closely linked with the National Hospital for Neurology and Neurosurgery and an internationally renowned centre for teaching and research in neurology and the neurosciences. The Institute of Child Health is responsible for research and teaching within the field of child health and pædiatric sub-specialities, and is closely associated with the Hospital for Sick Children at Great Ormond Street. Since the two institutes are next door, a Joint Computing Unit was formed to look after the information technology needs of academics, medical researchers, clinicians, scientists, administrative and library staff. It is in this heterogeneous environment that Kermit has proved to be a most versatile and valuable tool.

Initially the predominant use of Kermit was for mainframe access via Packet/Assembler/Disassemblers (PADs) linked to Britain's X25 Joint Academic Network −JANET. Kermit was also used extensively for micro-micro file transfers. In the days of CP/M, every machine seemed to differ with respect to disk format, and Kermit liberated data. Even when Kermit did not exist for particularly idiosyncratic machines, we still could use it for data transfers.

Perhaps the most interesting and topical use of Kermit is the way its elegant terminal emulators are being used in AIDS research.

For example, we had to replace several dozen hard-disk Z80 machines with a multiuser operating system (TurboDOS) for which there was no Kermit [*There is now! –Ed.*]. Fortunately TurboDOS had a hex dump and a CP/M batch capability. We replaced dumb terminals connected to TurboDOS machines with PS/2s running Kermit. LOG SESSION was used to capture directory listings from the TurboDOS machine which were then processed to produce a BAT file with a series of dump commands. This was TRANSMITted to the TurboDOS machines into a WordStar document and saved. The BAT file was then executed and LOG SESSION was used to capture the hex dumps. A small BASIC program converted the hex back to binary. In this way several years and megabytes of medical text and data were rescued from oblivion.

Another useful role of Kermit has been in the area of data capture. Many of the departments have medical data acquisition apparatus that produce ASCII data on a serial line. Establishing the appropriate communications parameters (baud rate, number of bits, parity, etc.) is easy with Kermit, as you can make changes until you see what looks sensible on the screen. Having established parameters, Kermit then becomes a production tool, using LOG SESSION to capture data for subsequent analysis. Typical of this is our Neuropathology Department who use a light pen with digitising apparatus to trace around the inner and outer circumferences of cross-sections of cells photographed on an electron microscope. The aim of this is to determine the thickness of the cell walls, as with the progression of neurological disease, cell walls get thinner. An analysis of the distribution of the thicknesses of cell walls from a representative sample gives a prognostic indication. Kermit captures the data with LOG SESSION for subsequent analysis by statistical packages.

On a more general level, Health and Safety Regulations require all departments to check laboratory apparatus for electrical safety on an annual basis. The checking apparatus is attached to a Toshiba portable; data is captured by Kermit for subsequent print-out back at base.

In addition to research and laboratory work we use another feature of Kermit — the script capabilities — for administrative purposes. We have a number of X.25 PADS and two X.25 switches that need detailed and longwinded configurations. The configuration on these devices is held in battery-backed RAM on a loader board. From time to time these boards fail and over one hundred lines of configuration parameters must be reentered. The configurations are now kept as Kermit scripts which reduce reconfiguration from a couple of hours to a couple of minutes. In a similar way, a Kermit script running on a PC plugged in to a VAX as a console allows us to automate ''standalone backups.'' Manually typing in standalone backup commands was a time-consuming and potentially dangerous operation as the VMS DCL procedural language is not available in standalone mode.

Kermit and AIDS Research

Perhaps the most interesting and topical use of Kermit is the way its elegant terminal emulators are being used in AIDS research. The Department of Epidemiology and Biostatistics at the Institute of Child Health coordinates the European Collaborative Study on AIDS research, a prospective study of children born to HIV-positive mothers in 10 European centres. Because maternal antibodies cross the placenta during pregnancy, the standard test for diagnosing HIV infection is of no value for infants. As HIV infection causes a lowering of CD4 cell count, pædiatricians attempt to monitor changes in CD4 levels to identify infected infants. But satisfactory ''normal'' age-related curves for CD4 count do not exist for children.

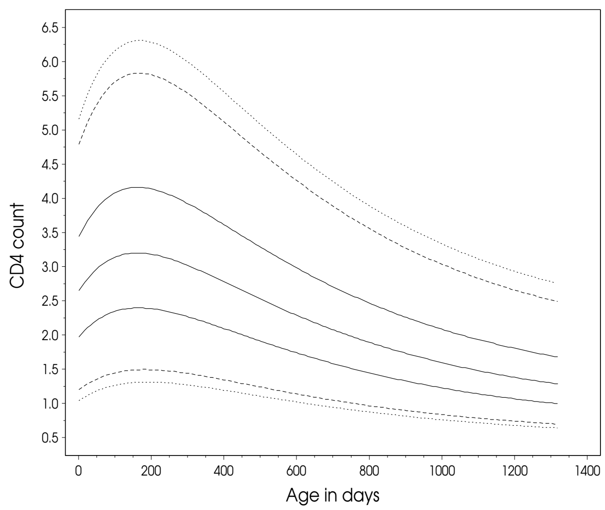

Figure 1 *Age vs CD4 Count*

At birth, CD4 count is relatively high; it rises further to peak at about 6 months before tailing off slowly to adult levels. The department is currently developing age-related standards for CD4 counts using blood samples taken from children from the study who were subsequently found to be uninfected. To create the curves the data is analyzed by being ''chopped'' into intervals and deskewed within each interval. This transformed data is then plotted in Tektronix emulation and curves are fitted.

How does Kermit help? Developing the curves becomes an interactive, iterative modelling process. A program has been written that allows suitable intervals to be chosen and entered into a VT320 screen, standard deviations are calculated and plotted by toggling the screen into Tektronix mode; if the curve is of interest it is saved for hard-copy laser printing, the screen is then toggled back to VT320 for the next set of intervals to be entered. An example of such a curve is shown in Figure 1. The X-axis shows age in days and the Y-axis shows the CD4 count. Each curve is a centile at the 3%, 5%, 25%, 50%, 75%, 95% and 97% intervals, reading from bottom to top. Thus 50% of non-infected children should have a CD4 count of about 3.2 at age 200 days.

These curves were recently presented at a workshop on Measurement and Use of CD4 and Other Lymphocyte and Immunophenotypic markers in Pediatric HIV Infection, organised by the National Institutes of Health in Washington DC (October 1992).

And Now, TCP/IP

What of the future? Until very recently, like most British academic sites, we were entirely X.25-based. Our Joint Computer Unit is part of a consortium of academic computing facilities, the Bloomsbury Computing Consortium. The farsighted directors of this consortium have designed and implemented an architecture based on networked UNIX machines, with a metropolitan area network using TCP/IP protocols. At grassroots level we have had to learn a whole new set of software tools.

Although being part of the Internet is, to say the least, exhilarating, we found products such as PC-NFS difficult to configure and the TELNET terminal emulators rather limited. The recent TCP additions to Kermit and the sheer readability of the parameters in an `MSKERMIT.INI` file has moved us from a situation of struggle to a situation of load and go, whereby we can bring new connections on stream in a matter of minutes. Finally, we have adapted the excellent dial-up script supplied with Kermit 3.11 to set PC e-mail lists that interface through the e-mail capability of C-Kermit on various UNIX hosts. It looks as though we will be using Kermit well into the next millennium.

Grateful acknowledgement to Angie Wade and David Dunn, Department of Epidemiology and Biostatistics, Institute of Child Health, University of London and to Peter Sacares, Statistician, Institute of Neurology, University of London.

Kermit in a Nonprofit Environment

Buz Overbeck, President
Grief Resource Foundation, Dallas, Texas, USA

Selecting software for a new nonprofit organization can be very challenging! Unlike most large corporations, whose budget allows the purchase and/or evaluation of large expensive packages, nonprofits, unless extremely well funded (a rarity), seldom have the money to buy many large packages, nor the name recognition to receive evaluation copies from vendors.

The Grief Resource Foundation falls into this category. Our funding comes from the sale of our publications and services, grants, and donations. The funding we receive is plowed back into the research and development of new services, with little left over for software. Like many nonprofits, we pay all of our bills on receipt which means we have no credit history. This alone disqualifies us from a corporate account at, say, Egghead Software, which would be very useful to us.

My fundamental approach to software selection is to find the most cost-effective solution to our applications needs. If there exists free software that will do the job, we use it. If not, we look at shareware. If only a commercial package will do, we'll write the company and ask that the package be donated. If all else fails, we'll buy it.

> *So far, I haven't found anything that Kermit can't do or be taught to do.*

One application of great importance to us is telecommunications. One of our functions is to provide resources and information to health care professionals and the public. So it's crucial that we are on top of what's going on in the areas of grief, loss, death and dying specifically and the mental health care field in general.

Software selection, in this case, was easy. We chose Kermit! With Kermit we can connect to the many different online resources available, find others in our field, and exchange information with them.

Although there are shareware and commercial packages which can (probably) do the same thing, for us, Kermit is the logical choice for the following reasons:

1. Kermit is inexpensive both in terms of cost and hardware resources.
2. Kermit is what Esperanto was supposed to be. Everyone speaks it. This is really important as we connect to a variety of systems including Library Bibliographies, File Servers, Bulletin Board Systems, and Commercial Databases. In other words, at any one time we may connect to a micro, mini, or mainframe computer where Kermit is the only common denominator.
3. Kermit is now ''Competitive''. When I first used Kermit, I had trouble adapting it to the micro world of BBS systems due to the lack of documentation in this area. For that reason, I almost passed on Kermit for the Foundation. But then came *Using MS-DOS Kermit* by Christine M. Gianone. That changed everything. Here, in one place, was the answer to almost every concern I had. No more excuses.

So, we adopted Kermit. This decision allowed us to put the money budgeted for an expensive communications package to better use. Since then, we've become pretty Kermit-literate. So far, I haven't found anything that Kermit can't do or be taught to do.

And for telecommunications at the Grief Resource Foundation, Kermit is the package of choice. I only wish that other application solutions were as obvious.

Miscellany

Modem Watch

Around mid-1992, the market was inundated with low-cost, high-speed V-Dot-Everything modems and 14,400-bit-per-second communication was suddenly as affordable as 2,400 bps was just a few years ago.

The TIES That Bind

One of the factors that contributes to the low cost of these newer modems is their elimination of the escape-sequence ‘‘guard time.’’ The guard time prevents accidental return to the modem’s command processor when the data stream happens to contain the escape sequence (most commonly +++). If the +++ is not preceded and followed by a full second of inactivity, it is transmitted safely rather than triggering an unwanted escape. The guard time concept is patented by Hayes Microcomputer Products, Inc. To cut costs, some modem manufacturers no longer support guard time. These companies say they have adopted a *new ‘‘technology,’’* which they call Time-Independent Escape Sequence, or TIES, in its place. To reduce the risk of accidental escape, some (*not all*) TIES modems require the escape sequence to be followed by a valid modem command, for example:

```
+++ATH0
```

To illustrate the effects of TIES, suppose the modem’s escape sequence was +++ and you wanted to upload this article through a TIES modem, using ASCII, XMODEM, YMODEM, ZMODEM, UUCP, or most other protocols. As soon as ‘‘+++ATH0’’ arrives at the modem, the connection hangs up.

The good news: It won’t happen during a Kermit file transfer. All the popular Kermit software versions, including the current releases of MS-DOS Kermit, C-Kermit, Mac Kermit, and IBM Mainframe Kermit are TIES-resistent. (Or should we say, *TIES-compliant?*)

Internal PC Modems

Many PC users are attracted by the even lower cost of internal versions of the new modems, which is achieved by eliminating the case, lights, and power supply. The internal modem must be installed in your PC in way that does not conflict with existing serial ports or other devices, often as COM3 or COM4 devices, using various interrupts, which tends to cause problems with many of our PC communication packages, including (until now) MS-DOS Kermit. Version 3.13 has been adapted to high-speed internal modems by allowing you to specify the hardware address and interrupt of any serial device, COM1 through COM4, and it treats the device with special care so as not to tickle the many bugs that have surfaced in these products.

Acknowledgements

Because of space constraints in this issue of *Kermit News*, many acknowledgements were omitted from their proper places. Kermit software development is a worldwide voluntary effort on the part of thousands of programmers, testers, and ordinary users who report problems or make suggestions. You already know the magnificent work of Joe Doupnik and John Chandler. Special thanks to Mike Normile of Data General Corporation for supporting and assisting with major portions of MS-DOS Kermit 3.13 development, and also to Novell (and particularly Brian Meek), to Microsoft, Inc., to Beame and Whiteside, Inc., and to Interconnections, Inc., for additional corporate assistance, and to Moshe Solow and Shalom Mitz at Hebrew University, Gudmundur Bjarni Josepsson at the University of Iceland, Hirofumi Fujii of the Japan National Laboratory for High Energy Physics, and to James Sturdevant and John Chandler for important information, code, and/or testing of new features.

C-Kermit 5A, written by Frank da Cruz of Columbia University, is the result of a massive three-year effort that also involved countless experts in UNIX, VMS, OS/2, AOS/VS, and other operating systems and their legion variants and releases. The list of contributors to its development takes up five pages in *Using C-Kermit!* To list only a few:

Chris Adie (Edinburgh U, Scotland), William Bader (Software Consulting Services, Nazareth, PA), Fernando Cabral (Padrão IX, Brasília, Brazil); Joe R. Doupnik (Utah State U); Stefaan Eeckels (Statistical Office of the European Community, CEC, Luxembourg); Kristoffer Eriksson (Peridot Konsult AB, Örebro, Sweden); Marcello Frutig (Catholic U, São Pãulo, Brazil); Hirofumi Fujii (Japan National Laboratory for High Energy Physics, Tokyo); William Glass; Andy Fyfe (Caltech); Eugenia Harris (Data General); Charles Hedrick (Rutgers U); Christian Hemsing (Rheinisch-Westfälisch Technische Hochschule, Aachen, Germany); Terry Kennedy (St Peter’s College, Jersey City, NJ); Lawrence Kirby (Wiltshire, UK); John Klensin (MIT); Tom Kloos (Sequent Computer Systems, Inc.); Bo Kullmar (ABC-Klubben, Stockholm, Sweden); David MacKenzie (Environmental Defense Fund, U of Maryland); Fulvio Marino (Olivetti, Ivrea, Italy); Peter Mauzey (AT&T); Bruce J. Moore; Paul Placeway; Kai Uwe Rommel (Technische Universität München, Germany); Jay S. Rouman (U of Michigan); Friðrik Skulason (U of Iceland, Reykjavik); Lee Tibbert (DEC); Warren Tucker (Tridom Corp, Mountain Park, GA); Konstantin Vinogradov (ICSTI, Moscow, Russia); Eduard Vopicka (Prague School of Economics, Czechoslovakia); Stephen Walton (California State U at Northridge); Jamie Watson (Adasoft, Switzerland); Rick Watson (U of Texas); Patrick Wolfe (Kuck & Associates, Inc.).

Ordering Information

Kermit software is distributed by Columbia University on magnetic tape, tape cartridges, and certain diskette formats. Tapes and cartridges include all source code and supporting files in machine-readable form for each Kermit implementation, and in some cases also binaries (encoded in hex or other printable format). Diskettes have no source code except when noted on the order form.

Kermit software programs are collected on six reel-to-reel 9-track tapes: A, B, C, D, E, and F. The programs are assigned to tapes A–F as shown in the second column of the Kermit version list as follows: Tape A has the MS-DOS, CP/M-80, and Apple II versions. Tape B has the IBM mainframe and DEC PDP-11 versions. Tape F has C-Kermit. Tape C has other miscellaneous microcomputer, PC, and workstation versions. Tape D has other miscellaneous minicomputer and mainframe versions. Tape E contains machine-readable copies of the Kermit protocol manual, various other manuals, articles, the Info-Kermit Digest, newsletters, a character-set-aware text-to-PostScript printing utility, and tape utilities. Tapes and cartridges are available in these formats:

ANSI: ANSI labeled ASCII, format D (variable length records), blocksize 8192. 9-track, half-inch, reel-to-reel, 1600 bpi. Readable by many computer systems, including VAX/VMS.

TAR: UNIX TAR format, blocksize 10240, 9-track, 1600 bpi.

OS: IBM OS standard labeled EBCDIC, format VB (variable length records), blocksize 8192, 9-track, 1600 bpi, for MVS, CMS, and other mainframe systems. IBM VM/CMS users should order the OS format and use one of the included tape-reading programs to read the tape on a CMS system; printed instructions are included with the OS tape.

TK50: TK50 tape cartridge for the DEC MicroVAX or VAXstation. VMS BACKUP format. Also readable by TZ30, TK70, and compatible drives.

QIC: UNIX TAR-format quarter-inch tape cartridge. Readable on Sun computers, IBM RS/6000, SCO systems, and other UNIX systems equipped with QIC cartridge drives.

8MM: EXABYTE 8-millimeter cassette, UNIX TAR format.

Diskettes formats are 5.25-inch 360K and 3.5-inch 720K.

NEWS AND UPDATES

Kermit News is mailed periodically free of charge to all our Kermit customers to bring news of Kermit software releases and related developments. Ordering any Kermit material from us automatically adds you to the subscriber list. We do not have the resources to send automatic software updates. Use the order form on page 27 to obtain new versions of the Kermit software, or call +1 212 854-3703 for inquiries.

TERMS AND CONDITIONS

The Kermit software −including source code −is furnished without warranty of any kind, and neither Columbia University, nor the individual authors or publishers, nor any institution that has contributed Kermit material, acknowledge any liability for any claims arising from the use of Kermit. Since source code is available, users may fix bugs and make improvements, and are encouraged to contribute their work back to Columbia for further distribution.

Kermit software may be ordered by private individuals, corporations, academic or government institutions, and other organizations for their own internal use, but the software may not be resold or otherwise redistributed to external clients, customers, or contractors without written permission of the Manager of Kermit Development and Distribution at Columbia University. Contact us for further information.

TO ORDER FROM COLUMBIA UNIVERSITY, fill out and return the enclosed order form. PREPAYMENT by credit card or check is encouraged; an additional ORDER PROCESSING FEE is required if we must issue an invoice. Orders are shipped by delivery service or US mail, normally within 2–4 weeks of receipt, but firm delivery schedules or methods cannot be guaranteed. Prices are in US dollars and include shipping costs. When two prices are shown (like $100 / $135), the first price applies to the USA, Canada, and Mexico and the second price is for shipments to other countries (exception: if you can supply with your Federal Express account number, then pay the first price). Rush service is available for an extra fee. Call +1 212 854-3703 for additional ordering information. Telephone and Fax orders are accepted if payment is by Master Card or Visa. Use the order form for Fax orders, and, for payment by credit card, be sure to include your signature.

Prefix	*Tape*	*Machine*	*Operating System*	*Program Language*	*Program Version*	*Released yy/mm/dd*
CK	F	*various*	4.1-3 BSD	C	5A189	93/06/30
CK	F	*various*	4.4 BSD/A	C	5A189	93/06/30
CK	F	*various*	POSIX	C	5A189	93/06/30
CK	F	*various*	QNX 4.1	C	5A189	93/06/30
CK	F	*various*	UNIX SIII	C	5A189	93/06/30
CK	F	*various*	UNIX SVR2	C	5A189	93/06/30
CK	F	*various*	UNIX SVR3	C	5A189	93/06/30
CK	F	*various*	UNIX SVR4	C	5A189	93/06/30
TRI	C	*various*	TRIPOS	BCPL	-	87/07/10
TD	C	*various*	TurboDOS	ASM	-	92/10/29
FLX	C	*various 6809*	Flex 9	6809 Asm	-	86/04/17
UF	C	*various 6809*	UniFLEX	C	1.4	89/02/02
OS9	C	*various 6809*	OS-9	C	1.5	85/09/20
CK	F	*various 68000*	OS-9	C	5A189	93/06/30
OS9	C	*various 68000*	OS-9	C	1.5	85/09/20
CP	A	*various 808x*	CPM80 2.2	LASM	4.11	91/04/23
CP	A	*various 808x*	CPM80 3.0	LASM	4.11	91/04/23
MS	A	*various 80x86*	MS-DOS	MASM	3.13	93/07/08
CP	A	Access Matrix	CPM80 2.2	LASM	4.11	91/04/23
BBC	C	Acorn BBC B	OS1.20	6502 Assem.	1.45	87/05/19
BBC	C	Acorn BBC B+,128	OS 2	6502 Assem.	1.45	87/05/19
CP	A	Acorn BBC	CPM80 2.2	LASM	4.11	91/04/23
BBC	C	Acorn Compact	OS 3	6502 Assem.	1.45	87/05/19
BBC	C	Acorn Compact	Panos	C	4C 57	87/07/14
BBC	C	Acorn Master 128	OS 3	6502 Assem.	1.45	87/05/19
AC	C	Acorn Workstation	PANOS	C	-	87/07/13
MS	A	ACT Apricot	MS-DOS	MASM	2.30	88/05/12
CP	A	Action Discovery	CPM80 2.2	LASM	4.11	91/04/23
AM	C	Alpha Micro 68K	AMOSL 1.2	AM68K Asm	1.0	85/02/11
CK	F	Altos ACS68000	UNIX III,V	C	5A189	93/06/30
CK	F	Amdahl Mainframe	UTSV, SVR2	C	5A189	93/06/30
CP	A	Amstrad	CPM80 2.2	LASM	4.11	91/04/23
APO	C	Apollo	Aegis	Pascal	2.9	89/05/07
CK	F	Apollo	Aegis 9.x	C	5A189	93/06/30
CK	F	Apollo/HP	SR10.0	C	5A189	93/06/30
CP	A	Apple II	CPM80 2.2	LASM	4.11	91/04/23
APP	A	Apple II	DOS,ProDOS	CROSS	3.87	90/12/05
UCA	C	Apple II	UCSD p-Sys	UCSD Pascal	1.0	86/04/08
CK	F	Apple Macintosh	AU/X	C	5A189	93/06/30
CK	F	Apple Macintosh	Mac OS	MPW C	09188	93/05/24
CN8	C	Argos Pro PC	CCPM86	ASM86	2.9	86/04/10
CK	F	AT&T 3B Series	UNIX SVR3	C	5A189	93/06/30
CK	F	AT&T 6300 PLUS	UNIX SVR3	C	5A189	93/06/30
CK	F	AT&T 6386 WGS	UNIX SVR3	C	5A189	93/06/30
CK	F	AT&T 7300 UNIXPC	UNIX Sys V	C	5A189	93/06/30
AR	C	Acorn Archimedes	Arthur	C	1.3	93/05/01
AR	C	Acorn Archimedes	RISC_OS	C	1.3	93/05/01
ATA	C	Atari Home Comp.	DOS	Action!	-	84/01/09
UCP	C	Atari MEGA ST2	UCSD p-Sys	Pascal	1.1	90/08/05
CK	F	Atari ST	GEMDOS	C	5A189	93/06/30
CK	F	Atari ST	MINIX ST	C	5A189	93/06/30
CP	A	BigBoard II	CPM80 2.2	LASM	4.11	91/04/23
B78	D	Burroughs A-Series	MCS/AS	Algol	1.019	86/09/11
CT	C	Burroughs B20	BTOS	C	2.00	93/01/20
B68	D	Burroughs B6800	CANDE	Algol	-	85/02/15
B78	D	Burroughs B7800	Burroughs	Algol	1.019	86/09/11
B79	D	Burroughs B7900	Burroughs	Algol	5.2	85/11/27
CD3	D	CDC Cyber	NOS	Fortran 5	3.4	88/05/10
CYB	D	CDC Cyber	NOS 2.2	Compass	1.0	86/04/17
NOS	D	CDC Cyber	NOS 2.4	Compass	1.30	87/05/19
CDC	D	CDC Cyber 170	NOS,NOS/BE	Fortran-77	2.2	84/09/07
CK	F	CDC Cyber 910	IRIX	C	5A189	93/06/30
CK	F	Charles River	UNOS	C	5A189	93/06/30
CIE	C	CIE 680/XX	REGULUS	C	-	87/01/26
CP	A	Cifer 1886	CPM80	LASM	4.11	91/04/23
CP	A	Comart Communica	CPM80 2.2	LASM	4.11	91/04/23
C64	C	Commodore 64	FORTH	FORTH	1.5	85/02/08
C64	C	Commodore 64/128	DOS	CROSS	2.27	92/09/30
CK	F	Commodore Amiga	3000UX	C	5A189	93/06/30
CK	F	Commodore Amiga	Intuition	C	5A189	93/06/30
CK	F	Commodore Amiga	UNIX SVR4	C	5A189	93/06/30
CP	A	Compupro IF 3/4	CPM80 2.2	LASM	4.11	91/04/23
CVK	D	Computervision	CGOS	Fortran S	1.21	87/03/04
PER	D	Concurrent 3200	OS/32 MT72	Fortran	1.0	87/03/04
CK	F	Concurrent 3200	Xelos SV	C	5A189	93/06/30
CK	F	Concurrent 6xxx	RTU 4,5,6	C	5A189	93/06/30
CK	F	Convergent	CTIX	C	5A189	93/06/30
CT	C	Convergent NGEN	CTOS	C	2.00	93/01/20
CK	F	Convex	ConvexOS	C	5A189	93/06/30
CP	A	CPT-85xx	CPM80 2.2	LASM	4.11	91/04/23
CK	F	Cray X/MP, Y/MP	UNICOS 6,7,8	C	5A189	93/06/30
CR	D	Cray-1,Cray-XMP	CTSS	Fortran-77	-	85/02/08
CK	F	Cray-2,Cray-3	CSOS	C	5A189	93/06/30
CP	A	Cromemco	CPM80 2.2	LASM	4.11	91/04/23
CN8	C	Daisy PCi	CCPM86	ASM86	2.9	86/04/10
CK	F	DEC Alpha AXP	OpenVMS	C	5A189	93/06/30
CK	F	DEC Alpha AXP	OSF/1	C	5A189	93/06/30
K12	D	DEC PDP-8	OS/8 Fam.	PAL-8	10g	90/09/13
CK	F	DEC PDP-11	2.1x BSD	C	5A189	93/06/30
K11	B	DEC PDP-11	IAS 3.1	Macro-11	3.60	89/06/13
MP	B	DEC PDP-11	MUMPS-11	MUMPS-1982	-	84/04/11
K11	B	DEC PDP-11	RSTS/E	Macro-11	3.60	89/06/13
K11	B	DEC PDP-11	RSX-11/M,+	Macro-11	3.60	89/06/13
K11	B	DEC PDP-11	RT-11	Macro-11	3.60	89/06/13
K11	B	DEC PDP-11	TSX+	Macro-11	3.60	89/06/13
K12	D	DEC PDP-12	OS/12	PAL-8	10g	90/09/13
K11	B	DEC Pro-3xx	P/OS	Macro-11	3.60	89/06/13
K11	B	DEC Pro-3xx	Pro/RT	Macro-11	3.60	89/06/13
C86	C	DEC Rainbow	CPM86	ASM86	2.9	84/12/03
MS	A	DEC Rainbow	MS-DOS	MASM	3.10	91/03/18
QNX	C	DEC Rainbow	QNX 1.x	C	1.0	85/09/23
CK	F	DEC VAX	4.xBSD	C	5A189	93/06/30
CK	F	DEC VAX	(Open)VMS	C	5A189	93/06/30
CK	F	DEC VAX	ULTRIX	C	5A189	93/06/30
CK	F	DEC VAX	UNIX SVR3	C	5A189	93/06/30
VMS	B	DEC VAX	VMS	Bliss,Macro	33128	91/01/04
MS	A	DEC VAXmate	MS-DOS	MASM, C	3.13	93/07/08
CK	F	DEC VAXstation	(Open)VMS	C	5A189	93/06/30
CK	F	DEC VAXstation	ULTRIX	C	5A189	93/06/30
CP	A	DEC VT-180 Robin	CPM80 2.2	LASM	4.11	91/04/23
K12	D	DEC VT-78	OS/8 Fam.	PAL-8	10g	90/09/13
K12	D	DEC VT-278	OS/78 V4	PAL-8	10g	90/09/13
K12	D	DECmate I	OS/78 V4	PAL-8	10g	90/09/13
K12	D	DECmate II,III	OS/278 V2	PAL-8	10g	90/09/13
CP	A	DECmate-II,III	CPM80 2.2	LASM	4.11	91/04/23
MS	A	DECmate-II,III	MS-DOS	MASM	2.29	86/05/28
CK	F	DECstation	OSF/1	C	5A189	93/06/30
CK	F	DECstation	ULTRIX	C	5A189	93/06/30
K10	D	DECsystem-10	TOPS-10	Bliss, Macro	3.134	89/09/18
K20	D	DECSYSTEM-20	TOPS-20	MACRO-20	4.2	88/01/25
CK	F	Dell PC	Dell UNIX	C	5A189	93/06/30
CP	A	Delphi 100	CPM80 2.2	LASM	4.11	91/04/23
RD2	D	DG 800	RDOS	BASIC	-	87/03/26
CK	F	DG AViiON	DG/UX	C	5A189	93/06/30
AOS	D	DG MV	AOS,AOS/VS	SP/Pascal	-	85/02/08
CK	F	DG MV	AOS/VS,-II	C	5A189	93/06/30
DGM	D	DG MV	AOS/VSMVUXC		-	85/11/27
RDO	D	DG Nova	RDOS	Fortran-5	-	84/09/14
AOS	D	DG S250	AOS	Fortran-5	-	84/09/14
CK	F	DIAB DS90	DNIX	C	5A189	93/06/30
CK	F	Dolphin Triton	UNIX SV/88	C	5A189	93/06/30
CK	F	Encore	UMAX 4.x	C	5A189	93/06/30
CK	F	Encore	UMAX V	C	5A189	93/06/30
CP	A	Epson PX8	CPM80 2.2	LASM	4.11	91/04/23
LUX	C	FACIT DTC/DTC2	ABC-DOS	ABC-BASIC-II	4.11	90/07/13
CN8	C	Fallon 2000	CCPM86	ASM86	2.9	86/04/10
CP	A	Ferguson BigB.I	CPM80 2.2	LASM	4.11	91/04/23
CK	F	Fortune 32:16	For:Pro2.1	C	5A189	93/06/30
CK	F	FPS 500	FPX 4.1	C	5A189	93/06/30
CN8	C	FTS PCi	CCPM86	ASM86	2.9	86/04/10
C86	C	Fujitsu Micro16s	CPM86	ASM86	2.9	85/09/23
C86	C	Future FX20/FX30	CPM86	ASM86	2.9	86/04/10
GEC	D	GEC 4000 Series	OS4000	MUM/SERC	3.9	89/05/07
OS9	C	Gimex III	OS-9	C	1.5	85/09/20
GM	D	Gould/SEL 32	MPX-32	Fortran 77+	2.3	86/12/10
MS	A	GRiD Compass II	MS-DOS	MASM	3.10	91/03/18
H1	D	Harris H100-1	VOS 4.1.1	Fortran-77	1.06	88/03/17
H8	D	Harris 800	VOS	Pascal,Asm	-	85/02/11
CK	F	Harris NightHawk	CX/UX 6.1	C	5A189	93/06/30
CP	A	Heath H8	CPM80 2.2	LASM	4.11	91/04/23
CP	A	Heath/Zenith-89	CPM80 2.2	LASM	4.11	91/04/23
CP	A	Heath/Zenith-100	CPM85	LASM	4.11	91/04/23
MS	A	Heath/Zenith-100	MS-DOS	MASM	3.10	91/03/18
MU	D	Honeywell	MULTICS	PL/I	2.0h	84/09/20
HD6	D	Honeywell DPS6	GCOS6	C	2.01	91/06/03
HG	D	Honeywell DPS66	GCOS3,8	C	3.0	84/10/05
HC6	D	Honeywell DPS8	CP-6	PL/6	1.00	88/01/28
HCP	D	Honeywell DPS8	CP-6	Pascal	-	85/04/04
HDP	D	Honeywell DPS8	GCOS/TSS	B	1.1	85/03/21
HG	D	Honeywell DPS8	GCOS3,8	C	3.0	84/10/05
HC6	D	Honeywell DPS90	CP-6	PL/6	1.00	88/01/28
HCP	D	Honeywell DPS90	CP-6	Pascal	-	85/04/04
HL6	C	Honeywell L6/10	MS-DOS	MASM	1.20A	84/10/05
MS	A	HP Portable Plus	MS-DOS	MASM	3.10	91/03/18
MS	A	HP-110, 150	MS-DOS	MASM	3.10	91/03/18
CP	A	HP-125	CPM80 2.2	LASM	4.11	91/04/23
HPM	D	HP-1000	RTE6, RTEA	F77 & Asm	1.99D	90/01/09
HP2	C	HP-264x	-	8080ASM	1.2	87/10/09
HP3	D	HP-3000	MPE	SPL	-	91/10/21
HP3	D	HP-3000	MPE	C	-	93/05/15
CK	F	HP-9000	HP-UX	C	5A189	93/06/30
HPB	C	HP-9000/200,/300	HP-BASIC	HP BASIC	1.02	89/06/21
CK	F	HP-9836CU	HP-UX	C	5A189	93/06/30
HP9	C	HP-9845	BASIC/SAM	HP BASIC	1.00	86/10/07
HP9	C	HP-98xx	UCSD p-Sys	HP Pascal	-	84/01/20
HP8	C	HP86	HP-BASIC	HP BASIC	1.01	87/04/29
HP8	C	HP87	HP-BASIC	HP BASIC	1.01	87/04/29
IK	B	IBM 370 Series	CICS/MVS	Assembler	4.2.4	92/09/30
IK	B	IBM 370 Series	CICS/VSE	Assembler	4.2.4	92/09/30
MT2	D	IBM 370 Series	MTS	PLUS	-	86/11/03
MTS	D	IBM 370 Series	MTS	Assembler	-	84/01/06
MTS	D	IBM 370 Series	MTS	Pascal	1.0	84/01/06

Prefix	*Tape*	*Machine*	*Operating System*	*Program Language*	*Program Version*	*Released yy/mm/dd*
IK	B	IBM 370 Series	MUSIC	Assembler	4.2.3	92/09/30
GUT	D	IBM 370 Series	MVS/GUTS	Assembler	-	85/04/05
IK	B	IBM 370 Series	MVS/ROSCOE	Assembler	4.2.4	92/09/30
IK	B	IBM 370 Series	MVS/TSO	Assembler	4.2.4	92/09/30
TS2	B	IBM 370 Series	MVS/TSO	Pascal/VS,..	2.3	87/10/01
TSN	B	IBM 370 Series	MVS/TSO	ALP/Assem	1.1A	87/09/17
IK	B	IBM 370 Series	MVSXA/TSOE	Assembler	4.2.4	92/09/30
IK	B	IBM 370 Series	VM/CMS	Assembler	4.2.5	92/12/23
IK	B	IBM 370 Series	VM/ESA/CMS	Assembler	4.2.5	92/12/23
IK	B	IBM 370 Series	VM/HPO/CMS	Assembler	4.2.5	92/12/23
IK	B	IBM 370 Series	VM/IS/CMS	Assembler	4.2.5	92/12/23
IK	B	IBM 370 Series	VM/SP/CMS	Assembler	4.2.5	92/12/23
IK	B	IBM 370 Series	VM/XA/CMS	Assembler	4.2.5	92/12/23
MS	A	IBM compatibles	MS-DOS	MASM, C	3.13	93/07/08
CS9	C	IBM CS9000	CSOS	Pascal	-	92/09/10
CK	F	IBM Mainframe	AIX 1.2	C	5A189	93/06/30
CK	F	IBM Mainframe	AIX/ESA2.1	C	5A189	93/06/30
UCI	C	IBM PC	UCSD p-Sys	UCSD Pascal	0.1	84/05/23
CC	C	IBM PC + clones	ChineseDOS	MASM	2.32A	91/09/09
MX	B	IBM PC family	MINIX	C	4D 61	88/05/17
TP4	C	IBM PC family	PC-DOS	Turbo Pascal	1.1a	88/04/15
QK	C	IBM PC,XT,AT	PC-DOS	Turbo Pascal	3.1	88/12/14
QNX	C	IBM PC,XT,AT	QNX 1.x	C	1.0	85/09/23
MS	A	IBM PC,XT,AT,jr	PC-DOS	MASM, C	3.13	93/07/08
MS	A	IBM PC family	MS Windows	MASM, C	3.13	93/07/08
WK	A	IBM PC family	MS Windows	Microsoft C	-	91/10/19
PIC	D	IBM PC/XT,AT	PICK	DATA/BASIC	0.3	89/08/21
CK	F	IBM PS/2	AIX 1.2	C	5A189	93/06/30
CK	F	IBM PS/2	OS/2 1.3	C	5A189	93/06/30
CK	F	IBM PS/2	OS/2 2.00	C	5A189	93/06/30
MS	A	IBM PS/2 Series	PC-DOS	MASM, C	3.13	93/07/08
CK	F	IBM RS/6000	AIX 3.x	C	5A189	93/06/30
CK	F	IBM RT PC	4.3BSDReno	C	5A189	93/06/30
CK	F	IBM RT PC	ACIS 4.x	C	5A189	93/06/30
CK	F	IBM RT PC	AIX 2.x	C	5A189	93/06/30
VME	D	ICL 2900	VME	S3	1.01	87/07/14
CK	F	ICL DRS3000,6000	DRS/NX	C	5A189	93/06/30
CN8	C	ICL PC 2,Quattro	CCPM86	ASM86	2.9	87/05/17
PQ	C	ICL/Perq	Perq OS	Pascal	2.0	84/12/04
CK	F	Integr.Solu.VS8	ISI 4.2BSD	C	5A189	93/06/30
MS	A	Intel 300 Series	iRMX-286	MASM/ASM86	2.30	88/05/02
CK	F	Intel 302	Bell Tech	C	5A189	93/06/30
RMX	C	Intel 86,286	RMX 1.0	PL/M	1.0	85/10/25
I86	C	Intel 86/380	iRMX-86	PL/M	2.3	85/09/23
IRM	C	Intel 86/380	iRMX-86	PL/M	2.41	87/03/04
MD	C	Intel MDS	ISIS	PL/M	-	87/04/06
CK	F	Intergraph Clipper	CLIX	C	5A189	93/06/30
CP	A	Ithaca Intersys	CPM80 2.2	LASM	4.11	91/04/23
UCJ	C	J Loebl Magiscan2	UCSD p-Sys	UCSD Pascal	-	86/06/23
CP	A	Kaypro II	CPM80 2.2	LASM	4.11	91/04/23
CP	A	Kaypro 4	CPM80 2.2	LASM	4.11	91/04/23
M2	C	Lilith Worksta.	Medos	Modula-2	1.0	87/05/17
LM	C	LMI Lispmachine	LMI-Lambda	ZETALISP	1.0	85/09/12
CP	A	Lobo Max-80	CPM80 2.2	LASM	4.11	91/04/23
LUX	C	Luxor ABC-80	ABC-DOS	Z80 Asm	1.0	90/07/13
LUX	C	Luxor ABC-80x	ABC-DOS	ABC-BASIC-II	4.11	90/07/13
CK	F	Luxor ABC-9000	DNIX	C	5A189	93/06/30
MBF	D	MAI Basic Four	BOSS/VS	BASIC BB86	1.0	88/04/11
CK	F	Masscomp	RTU 4.0+	C	5A189	93/06/30
CP	A	Merlin M2215	CPM80 2.2	LASM	4.11	91/04/23
PIC	D	MicroDataREALITY	PICK	DATA/BASIC	0.3	89/08/21
CP	A	Micromint SB180	CPM80 2.2	LASM	4.11	91/04/23
CK	F	MIPS System	RISC/os	C	5A189	93/06/30
CK	F	Modcomp 9730	Real/IX	C	5A189	93/06/30
MOD	D	Modcomp Classic	MAX IV	Fortran/ASM	A.0	87/01/26
CK	F	Modcomp Realstar	Real/IX	C	5A189	93/06/30
CP	A	MorrowDecisionI	CPM80 2.2	LASM	4.11	91/04/23
CP	A	MorrowMicroDecI	CPM80 2.2	LASM	4.11	91/04/23
FL	C	Motorola 6809	Flex	Assembler	-	86/02/14
FL2	C	Motorola 6809	FLEX-09	C	3.0	87/03/04
CK	F	Motorola Delta	SV/68 R3x	C	5A189	93/06/30
CK	F	Motorola Delta	SV/88 R3,4	C	5A189	93/06/30
NCR	D	NCR 9800-4	VE4.0	C	4E 72	90/07/13
CP	A	NCR Decisionmate	CPM80 2.2	LASM	4.11	91/04/23
CK	F	NCR System 3000	UNIX SVR4	C	5A189	93/06/30
CK	F	NCR Tower 1632	UNIX SVR2	C	5A189	93/06/30
CK	F	NCR Tower 32	UNIX SVR3	C	5A189	93/06/30
C86	C	NEC APC	CPM86	ASM86	2.9	84/12/03
MS	A	NEC APC	MS-DOS	MASM	2.29	86/05/28
MS	A	NEC APC III	MS-DOS	MASM	2.30	88/03/21
MS	A	NEC PC9801	MS-DOS	MASM	3.10	91/04/18
CK	F	NeXT	NeXT OS	C	5A189	93/06/30
CK	F	Nixdorf Targon31	TOS	C	5A189	93/06/30
CP	A	Nokia MikroMikko	CPM80 2.2	LASM	4.11	91/04/23
ND	D	Norsk Data 10,..	SintranIII	ND-Pascal	3.1b	85/06/24
CK	F	Norsk Data 88/17	UNIX SV/88	C	5A189	93/06/30
CP	A	Northstar	CPM80 2.2	LASM	4.11	91/04/23
CP	A	Ohio Scientific	CPM80 2.2	LASM	4.11	91/04/23
CK	F	OkiStation 7300	UNIX SVR4	C	5A189	93/06/30
CK	F	Olivetti CP 486	UNIX SVR4	C	5A189	93/06/30
CK	F	Olivetti LSX30xx	X/OS	C	5A189	93/06/30
CK	F	Olivetti PCs	SCO UNIX	C	5A189	93/06/30
MS	A	Olivetti PCs	MS-DOS	MASM, C	3.13	93/07/08
CN8	C	Orion PCi	CCPM86	ASM86	2.9	86/04/10
CP	A	Osborne 1	CPM80 2.2	LASM	4.11	91/04/23
UCM	C	PascalMicroengin	UCSD p-Sys	Pascal	III.0	84/12/03
MS	A	PC/8088 & higher	MS-DOS	MASM, C	3.13	93/07/08
CK	F	PC/286 & higher	Microport	C	5A189	93/06/30
CK	F	PC/286 & higher	SCO Xenix	C	5A189	93/06/30
CK	F	PC/386 & higher	386BSD	C	5A189	93/06/30
CK	F	PC/386 & higher	BSDI/386	C	5A189	93/06/30
CK	F	PC/386 & higher	Coherent	C	5A189	93/06/30
CK	F	PC/386 & higher	ESIX SVR3	C	5A189	93/06/30
CK	F	PC/386 & higher	Interactive	C	5A189	93/06/30
CK	F	PC/386 & higher	Linux/386	C	5A189	93/06/30
CK	F	PC/386 & higher	Microport	C	5A189	93/06/30
CK	F	PC/386 & higher	MINIX/386	C	5A189	93/06/30
CK	F	PC/386 & higher	MtXinuMach	C	5A189	93/06/30
CK	F	PC/486 & higher	NeXTSTEP	C	5A189	93/06/30
CK	F	PC/386 & higher	OS/2 1.x	C	5A189	93/06/30
CK	F	PC/386 & higher	OS/2 2.0	C	5A189	93/06/30
CK	F	PC/386 & higher	SCO ODT	C	5A189	93/06/30
CK	F	PC/386 & higher	SCO UNIX	C	5A189	93/06/30
CK	F	PC/386 & higher	Solaris	C	5A189	93/06/30
CK	F	PC/386 & higher	UnixWare	C	5A189	93/06/30
UCP	C	Pecan	UCSD p-Sys	Pascal	1.1	90/08/05
PER	D	PerkinElmer 3200	OS/32 MT72	Fortran	1.0	87/03/04
CK	F	PerkinElmer 3200	Xelos SV	C	5A189	93/06/30
PE7	D	PerkinElmer 7000	IDRIS	C	1.1 0	86/12/08
CP	A	PMC Micromate101	CPM80 2.2	LASM	4.11	91/04/23
PRI	D	Prime	PRIMOS R2x	PL/P	8.15	93/04/19
CK	F	Pyramid 9810x(T)	OSx	C	5A189	93/06/30
CP	A	Rair Black Box	CPM80 2.2	LASM	4.11	91/04/23
CP	A	RM380ZF, ZM	CPM80 2.2	LASM	4.11	91/04/23
RM	C	RML 480Z	ROS 2.x	C	1.22	86/11/03
RM	C	RML Nimbus	MS-DOS?	C	1.22	86/11/03
CP	A	Sanyo 1100 MBC	CPM80 2.2	LASM	4.11	91/04/23
MS	A	Sanyo 550 MBC	MS-DOS	MASM	2.30	88/05/16
CP	A	ScreenTyper	CPM80 2.2	LASM	4.11	91/04/23
CK	F	Sequent	DYNIX	C	5A189	93/06/30
CK	F	Sequent	DYNIX/ptx	C	5A189	93/06/30
CK	F	Silicon Graphics	IRIX	C	5A189	93/06/30
QL2	C	Sinclair QL	QDOS	BCPL	-	87/05/15
QLK	C	Sinclair QL	QDOS	C	1.10	87/05/15
CK	F	Solbourne	OS/MP	C	5A189	93/06/30
CK	F	Sony NEWS	NEWS-OS	C	5A189	93/06/30
UN	D	Sperry 1100	Exec	Assembler	2.5	86/09/03
UN	D	Sperry 1100	Exec	NOSC Pascal	2.0	84/10/08
CK	F	Sperry 5000	UNIX SVR3	C	5A189	93/06/30
CK	F	Sperry 5000	UTS V	C	5A189	93/06/30
SP9	D	Sperry 90/60	VS9	Assembler	-	86/04/09
CK	F	Stardent 1520	UNIX SVR3	C	5A189	93/06/30
CK	F	Sun, all models	Solaris 1,2	C	5A189	93/06/30
CK	F	Sun, all models	SunOS 3,4,5	C	5A189	93/06/30
CP	A	Superbrain	CPM80 2.2	LASM	4.11	91/04/23
LM	C	Symbolics 36xx	Lisp	ZETALISP	1.0	85/09/12
TAN	D	Tandem Nonstop	Guardian	TAL	1.0	86/04/08
TA1	C	Tandy 100	Tandy 100	BASIC	1984	90/10/07
TA2	C	Tandy 2000	MS-DOS	MASM	1.20	84/02/16
CK	F	Tandy Model 16	Xenix 3.0	C	5A189	93/06/30
C86	C	Tektronix 4170	CPM86	ASM86	2.9	84/12/03
CK	F	Tektronix 4xxx	UTek	C	5A189	93/06/30
CK	F	Tektronix 6130	UTek	C	5A189	93/06/30
CK	F	Tektronix XD88	UTek	C	5A189	93/06/30
CP	A	Telcon Zorba	CPM80 2.2	LASM	4.11	91/04/23
CP	A	Teletek	CPM80 2.2	LASM	4.11	91/04/23
UCT	C	Terak	UCSD p-Sys	UCSD Pascal	-	84/04/11
TI9	D	TI 990	DX10	Pascal	1.0	87/07/10
EXP	C	TI Explorer	LISP	Common Lisp	1.0	87/03/04
MS	A	TI Professional	MS-DOS	MASM	2.29	86/05/28
CP	A	Torch Unicorn 5	CPM80 2.2	LASM	4.11	91/04/23
CK	F	Tri Star Flash	ESIX	C	5A189	93/06/30
CO	C	TRS-80 CoCo	DOS	EDTASM	1.1	85/03/21
OS9	C	TRS-80 CoCo	OS-9	C	1.5	85/09/20
TRS	C	TRS-80 I and III	TRSDOS	M80	3.5	84/08/08
CP	A	TRS-80 Model II	CPM80 2.25	LASM	4.11	91/04/23
TR2	C	TRS-80 Model II	TRSDOS	Assembler	1.2	87/03/26
CP	A	TRS-80 Model 4	CPM80 2.2	LASM	4.11	91/04/23
M4	C	TRS-80 Model 4	TRSDOS	ASM	5.2	86/10/29
UM	C	UMicro U-MAN1000	CP/M-68K	C and Asm	-	86/04/10
CK	F	UNISYS 5000	UNIX SVR3	C	5A189	93/06/30
CK	F	UNISYS 5000	UTS V	C	5A189	93/06/30
UAS	D	UNISYS A-Series	MCS/AS	Algol	1.041	90/07/13
CP	A	USmicroSalesS100	CPM80 2.2	LASM	4.11	91/04/23
PIC	D	VAX/Ultimate	PICK	DATA/BASIC	0.3	89/08/21
CP	A	Vector Graphics	CPM80 2.2	LASM	4.11	91/04/23
C86	C	Victor/Sirius 1	CPM86	ASM86	2.9	86/07/07
MS	A	Victor/Sirius 1	MS-DOS	MASM	3.10	91/03/18
CP	A	Video Genie	CPM80 2.2	LASM	4.11	91/04/23
MS	A	Wang PC, APC	MS-DOS	MASM	2.31	88/08/13
CP	A	Xerox 820	CPM80 2.2	LASM	4.11	91/04/23

Prices, terms, and items are subject to change. If this form is dated more than 6 months prior, please contact us for new information. Please order carefully since we CANNOT refund or exchange items. Prices are in US dollars ($), first price for North America / second price for shipping outside North America (unless you pay shipping).

9-TRACK MAGNETIC TAPE. Price: $100 / $135 *per tape:*

	ANSI	TAR	OS		ANSI	TAR	OS		ANSI	TAR	OS
Tape A:	❐	❐	❐	Tape C:	❐	❐	❐	Tape E:	❐	❐	❐
Tape B:	❐	❐	❐	Tape D:	❐	❐	❐	*See below for C-Kermit . . .*			

9-TRACK TAPE SUBTOTAL . . . . $____________

TAPE CARTRIDGES. Price: $150 / $185.

	8MM	QIC	TK50
Contents of Tapes A, B, and E:	❐	❐	❐
Contents of Tapes C, D, and E:	❐	❐	❐

TAPE CARTRIDGE SUBTOTAL . . . . $____________

MS-DOS KERMIT

IBM PC/PS2 MS-DOS Kermit software with book *Using MS-DOS Kermit*, $34.95 / $45:

❐ 5.25-inch ❐ 3.5-inch: . . . . $____________

IBM PC/PS2 MS-DOS Kermit utilities and technical documentation: $35 / $40:

❐ 5.25-inch ❐ 3.5-inch: . . . . $____________

MS-DOS IBM PC/PS2 Kermit source code, $60 / $68:

❐ 5.25-inch ❐ 3.5-inch: . . . . $____________

Crynwr (formerly Clarkson) packet drivers. For the IBM PC family on DOS diskettes:

Binaries and Docs, $35 / $40: ❐ 5.25-inch ❐ 3.5-inch . . . . $____________

Source Code, $60 / $68: ❐ 5.25-inch ❐ 3.5-inch . . . . $____________

C-KERMIT 5A, includes book *Using C-Kermit*:

❐ 9-Track Tape, $135 / $170, Format: ❐ ANSI, ❐ TAR . . . . $____________
❐ TK50 cartridge, DEC VMS / OpenVMS BACKUP format, $185 / $220 . . . . $____________
❐ Quarter-Inch Cartridge (QIC), UNIX TAR format, $185 / $220 . . . . $____________
❐ 8mm EXABYTE cartridge, UNIX TAR format, $185 / $220 . . . . $____________
❐ Source code on DOS-format diskettes, $100 / $115: ❐ 5.25-inch, ❐ 3.5-inch: . . . . $____________

Binaries on DOS-format diskettes, *Using C-Kermit* book included, $45 / $50 each:

3.5"	5.25"		
❐	❐	C-Kermit for OS/2 1.xx, 16-bit	$____________
❐	❐	C-Kermit for OS/2 2.00, 32-bit	$____________
❐	❐	C-Kermit for OS-9/68000	$____________
❐	❐	C-Kermit for Commodore Amiga	$____________
❐	❐	C-Kermit for Atari ST	$____________

MACINTOSH KERMIT

❐ *Mac Kermit* 0.9(40) 1988 release, 3.5-inch, $25 / $30 . . . . $____________
❐ *Mac Kermit* 0.99(???) latest prerelease, 3.5-inch, $25 / $30 . . . . $____________

LITERATURE

❐ Book: *Kermit, A File Transfer Protocol:* $29.95 / $35 . . . . $____________
❐ Book: *Using C-Kermit* (without software): $34.95 / $45 . . . . $____________
❐ Book: *Using MS-DOS Kermit:* $34.95 / $45 . . . . $____________
❐ Technical paper, Recent Kermit Protocol Extensions: $15 / $25 . . . . $____________

Kermit Digest volumes, $15 / $25 per year:

❐ This year ❐ Last 2 years ❐ Last 3 years ❐ Last 4 years . . . . $____________

Manuals for Kermit programs: $10 / $13 each:

❐ IBM 370 ❐ Apple II ❐ PDP-11 ❐ CP/M-80 . . . . $____________

SIDE 1 SUBTOTAL (please complete side 2 also) . . . . $____________

Shipping by UPS or post is included in the price. Please do not add sales tax.

SUBTOTAL from Side 1: . . . . $____________

❒ Voluntary tax-deductible donation *(help support the Kermit effort) :* . . . . $____________

USA, CANADA, AND MEXICO ONLY:
❒ For PRIORITY HANDLING and NEXT-DAY SHIPMENT, add $30: . . . . $____________

A. TOTAL MATERIALS, DONATION, AND SHIPPING: . . . . $____________

Please complete ONE of the numbered sections, 1, 2, or 3, and then fill in your shipping information.

1. PAYMENT BY CREDIT CARD

❒ MasterCard ❒ Visa AMOUNT OF YOUR PAYMENT (Line A above): $____________

Card Number ______________________________ Expiration Date ______________________________

Signature ______________________________ Today's Date ______________________________

2. PREPAYMENT BY CHECK

Please enclose a check for the total amount, *payable in US dollars*. PLEASE DO NOT MAKE ELECTRONIC BANK TRANSFERS OR SEND INTERNATIONAL POSTAL COUPONS. Make your check payable to:

Columbia University Kermit Distribution

If your check is *not* drawn on a US bank, please add a $35 check-cashing fee: . . . . $____________

TOTAL AMOUNT OF YOUR CHECK: . . . . $____________

3. PURCHASE ORDERS

B. Amount from Line A above: $____________
C. Add $25 order processing fee: $____________
D. If your check will *not* be drawn on a US bank, add $35 check-cashing fee: $____________
TOTAL, Lines B, C, and D. Please enclose your purchase order for this amount: . . . . $____________

SHIPPING INFORMATION (Do not use Post Office Box for UPS or Federal Express):

Name:______________________________ Organization: ______________________________

Address: ______________________________

City:____________________State or Province:______________Zip or Postal Code: ______________________

Country:______________________________Phone: ______________________________

If you want us to use your Federal Express account, please provide your account number and your signature:

Federal Express Account Number ______________________________

Signature ______________________________

MAIL YOUR COMPLETED ORDER FORM TO:

Kermit Distribution, Department OP
Columbia University Academic Information Systems
612 West 115th Street
New York, NY 10025-7721 USA

Phone: +1 212 854-3703, FAX: +1 212 663-8202, E-Mail: kermit@columbia.edu. Sorry, we can't respond by FAX.

Thank you!

Directory

Postal Address:

Kermit Development and Distribution
Columbia University Academic Information Systems
612 West 115th Street
New York, NY 10025-7721 USA

Staff:

Christine M. Gianone, Manager
Max Evarts
Andy Newcomb

Telephone:

Ordering Information: +1 212 854-3703
Technical Support: +1 212 854-5126
Fax: +1 212 663-8202

Networks:

BITNET/EARN/CREN: `KERMIT@CUVMA`
Internet: `kermit@columbia.edu`