ARPA Order No. 2223
ISI/TM-79-15
January 1979

the JOY of TENEX and TOPS-20

... in two parts

PART ONE

* A PROSE GLOSSARY ... is a glossary of technical terms arranged in a more or less logical fashion with an index [appended] to help the nonserial user find a particular term. If the definition does not shed sufficient light, the context may.

* The ARPANET ... is an operational, computerized, packet switching DoD digital network which provides a capability for terminals or geographically separated computers, called hosts, to communicate with each other.

* TIP USER'S GUIDE ... describes the use of a terminal connected to a Terminal Interface Processor (TIP) in the ARPA Network.

* TENEX ... is a time-sharing operating system built around the DEC PDP-10 computer.

* TOPS-20 ... is a time-sharing operating system based on the TENEX operating system with some features from the TOPS-10 operating system and a few entirely new features that make it possible to timeshare the DECSYSTEM-20 computer.

* SNDMSG ... is a program to send messages to users throughout the ARPANET.

* MSG MESSAGE HANDLING PROGRAM ... for reading, sending, and manipulating network mail and files which have a "message file format".

* INTRODUCTION TO HERMES ... a computer program for sending and receiving messages over a computer network, and creating and managing files of network messages.

* MAILSTAT PROGRAM ... lists and permits manipulation of undeliverable network messages.

Prepared by
Chloe Sommers Holg

UNIVERSITY OF SOUTHERN CALIFORNIA
INFORMATION SCIENCES INSTITUTE
4676 Admiralty Way · Marina del Rey · California 90291

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER: ISI/TM-79-15	2. GOVT ACCESSION NO. / 3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle): The Joy of TENEX and TOPS-20 Part One	5. TYPE OF REPORT & PERIOD COVERED: Technical Manual
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s): Chloe Holg	8. CONTRACT OR GRANT NUMBER(s): DAHC 15-7200308
9. PERFORMING ORGANIZATION NAME AND ADDRESS: USC/Information Sciences Institute, 4676 Admiralty Way, Marina del Rey, CA 90291	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS: Defense Advanced Research Projects Agency, 1400 Wilson Blvd., Arlington, VA 22209	12. REPORT DATE: March 1979 / 13. NUMBER OF PAGES: 123
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office): -------	15. SECURITY CLASS. (of this report): Unclassified / 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
15. DAHC15-72-C-0308, ARPA Order-2223	
16. DISTRIBUTION STATEMENT (of this Report): This document approved for public release and sale; distribution unlimited. 12 125p.	
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)	
18. SUPPLEMENTARY NOTES	DDC RECEIVED JUN 6 1979 A
19. KEY WORDS (Continue on reverse side if necessary and identify by block number): ARPANET, Hermes, MAILSTAT, MSG, SNDMSG, TENEX, TOPS-20	
20. ABSTRACT (Continue on reverse side if necessary and identify by block number): This is a basic manual discussing the operation of the ARPANET, the TENEX and TOPS-20 operating systems, and the following mail handling programs: SNDMSG, MSG, Hermes, and MAILSTAT.	

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 68 IS OBSOLETE
S/N 0102-014-6601

Unclassified
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

ARPA Order No. 2223
ISI/TM-79-15
January 1979

the JOY of TENEX and TOPS-20 in two parts

PART ONE

This document is one of a series of user manuals being made available by ARPA IPTO through ISI.

Prepared by Chloe Sommers Holz

USC/Information Sciences Institute

This research is supported by the Advanced Research Projects Agency under Contract No. DAHC 15 72 C 0308, ARPA Order No. 2223.

CONTENTS

1. A PROSE GLOSSARY

Glossaries of technical terms are very often of little use to the novice since their alphabetic order does not reflect any other logical structure. Attempted here is a glossary arranged in a more or less logical fashion with an index [appended] to help the nonserial user find a particular term. If the definition does not shed sufficient light, the context may.

A *computer* is a machine for performing complex processes on information without manual intervention. *Analog computers* perform this function by directly measuring continuous physical quantities such as electrical voltages. The best-known analog computer is a slide rule. *Digital computers* represent numerical quantities by discrete electrical states which can be manipulated logically and hence arithmetically. Digital computers are sometimes refered to as *electronic data processing machines, EDP,* or *processors.* In order to distinguish the actual physical equipment from the programs which extend its usefulness, the former is called *hardware.*

The *central processing unit (CPU) or main frame* is the portion of the computer which performs the calculations and decisions; the *memory* or *storage* is the part in which the data and programs are stored. The *core memory* is the main memory of most modern machines; it is normally the only memory directly accessible to the CPU. Its name derives from its composition; small ferrite rings called *cores.* The computer may have additional memory devices; information is transferred between these and the core memory. The most usual such memories are *magnetic drums* (spinning cylinders with a magnetizable recording surface) and *magnetic discs* (flat spinning discs with magnetizable surfaces).
The capability of memory devices is measured in capacity and speed of access. The *storage capacity* of a memory is measured in *words* (also called *cells* or *registers*) which are usually of fixed length, consisting of 12 to 48 *bits.* This number is called

the machine's *word length*. A bit (binary digit) is the minimum unit of information storage and has only two possible values. Capacity can also be measured in *bytes*, units of 6 or 8 bits, each

Access speed of a memory is the time it takes for the processor to obtain a word from memory. Core memory is called *random access* when any word can be obtained at any time without regard to its serial order. Drum, tape, and disc memories are *serial access*, because the words pass one at a time as they move past the station where they may be accessed. Speed is usually spoken in terms of *milliseconds* (msec) (thousandths of a second), *microseconds* (usec) (millionths of a second), or *nanoseconds* (nsec) (billionths of a second). One nanosecond is the time required for light to travel almost one foot.

The central processor and the memory constitute the computer per se; to get data and programs into the machine and the results out are the role of the *input/output equipment* or *I/O*.

Input devices convert information to a form which it can be stored in the computer's memory. The commonest form of input is the *punched card* or *Hollerith card* (after its inventor). Input devices which accept cards are called *card readers* and the function they perform is commonly called *reading*, as is that of all input devices. Cards have 80 columns with 12 possible punch positions; normally, each *column* is used to represent one character. A set of cards is called a *deck*. Another form of input is *punched paper tape* -- continuous tape approximately 1 inch wide, with holes punched across its width to represent characters or numeric quantities. *Magnetic ink character readers* have come to be used for input, particularly in banking; they can interpret characters printed with a special ink. More recently, *optical scanners* have appeared, which can read clearly printed or typed material of given type fonts.

Output devices usually include a *card punch* (which converts the characters stored in memory to punched holes in a card), a *tape punch* (which performs the same function for punched paper tape), and a *line printer* (which prints numerals, letters, and other characters of conventional design on continuous rolls of paper). When it passes information to these devices, the computer is *writing*. Recent additions to the output family include the *display device* which exhibits readable characters or graphic information on the face of a *cathode ray tube* or *CRT*. These images must be read at once, of course, since they are not permanent.

Information which can be taken away in permanent form (such as the output of a line printer) is called *hard copy*. A *plotter* is an output device which, under computer control, can draw continuous lines or curves on paper, thus producing graphs, maps, etc., in hard copy. *Magnetic tape* is widely used both as a form of memory and I/O. It can be stored conveniently away from the machine and can be read or written by the computer if it is put on a *tape drive* attached to the computer. It is the fastest type of I/O and the slowest type of memory except when used for serial reading.

I/O devices connected directly to the computer memory and under control of the CPU are spoken of as being *on-line*. They are placed *off-line* when they are used to perform independent functions. For example, it is common to exchange information between punched cards and magnetic tape off-line. Some devices are always off-line. They are *peripheral equipment* and are generally called collectively *electromechanical accounting machines* or *EAM*. These are frequently used independently of the computer and in fact antedate computers by many years. The most common are the *keypunch*, used to punch cards, the *reproducer*, which makes copies of decks of cards, and the *sorter*, which places cards in different bins as a function of which holes are punched. In some recent systems, another on-line I/O device has been added, the *console* or *terminal*. These are intended for the user to interact with the machine, and usually consist of a typewriterlike keyboard, and either a typewriterlike printing mechanism or another display device for output.

Information is stored in the computer's memory in the form of the presence or absence of a magnetic charge. A collection of such "yes or no" physical states is usually thought of as a *binary number* (a number whose only possible digits are 0 and 1). Depending on context, such numbers can have many meanings; in this sense, the numbers are *coded*. They can be interpreted as numeric quantities, *characters* (letters, digits, punctuation marks) or *instructions* or *commands* which will direct the computer to perform its basic functions (add, compare, read, etc.).

A set of instructions to perform a specified function or solve a complete problem is called a *program*. The computer performs such instructions sequentially. However, as the computer can modify the data in its memory, it can also modify its program. This capability to modify its own directions is a case of the engineering principle called *feedback*, the modification of future performance on the basis of past performance. It is because

of this distinctive feature that modern digital computers are sometimes called *stored program computers*. Parts of programs are sometimes called *routines* or *subroutines*. Subroutines which perform generally useful functions are sometimes combined into a subroutine *library*, usually on magnetic tape. Copies of relevant subroutines will be added to a program automatically and hence need not be developed by hand. Single *instructions* in a program are sometimes called *steps*. When a sequence of program steps is operated repeatedly, the process is called a *loop*. Certain instructions compare two quantities and select either of two program paths on the basis of the result: these are called *branching instructions*.

The data on which a program acts are usually structured into *tables*. Individual values which control the operation of programs or subroutines are *parameters*. An organized collection of information in the computer or on tape is called a *file*, like the organized set of papers in a file cabinet. A *data base* or *data bank* is a large and complex set of tables which describes some aspect of the world outside of the computer (a library catalog, a student record file, a budget).

A *programmer* is a person who converts a problem into a set of directions to a computer to solve it. The function is sometimes broken down into several parts, particularly if the problem is very complex. The task of stating the problem in a clear and unambiguous form is performed by an *analyst* or *system analyst*. The technique of specifying methods of solution for mathematical problems is *mathematical analysis* or *numerical analysis*. A specific procedure for solving a problem is an *algorithm*. The process of writing the detailed step-by-step instructions for the computer to follow is *coding* done by a *coder*.

After a program is written, it is tested by letting it perform its function in the computer on test data to which the proper solution is known. This process is *code checking* or *debugging*. The coder will also produce some descriptions of this program and how it operates so that others may understand how it works, in case at a future date it is necessary to modify it. This *documentation* may include a *flowchart*: a graphic description or diagram of the various paths and branches followed by the program.

The repertory of instructions available to the programmer for a specific computer is that computer's *machine language*. Other *higher-order languages* have been developed to help the programmer by simplifying the tedious aspects of writing machine language;

these are called *procedure oriented languages* or *problem oriented languages* or *POL*. Commonly used POLs are *Fortran, Algol*, and *Cobol*; the first two were devised mainly for scientific computation and the latter for business data processing. A new type is represented by *list processing languages*; because of greater flexibilities in dealing with data, these languages are particularly useful in nonnumeric computations such as are frequently involved in research. Their particular virtues are most apparent in *heuristic processes*: methods where the precise method of solution is not spelled out but is discovered as the program progresses and as it evaluates its progress toward an acceptable solution. (Because this use of the "language" is somewhat misleading, human languages such as English are distinguished as *natural languages*.)

Programs which convert higher-order languages into machine language are called *compilers*; programs which perform similar functions but at a much simpler level are *assemblers*. The term *translator* is used sometimes for a compiler, but it is used less frequently because of the possible confusion with programs which perform translation between natural languages. *Interpreters* do not compile the entire program but translate and perform one statement of the program at a time; effectively, they perform both functions-compiling and running a program.

Software is the term used to refer to the totality of programs and procedures available on a computer; sometimes it us used more specifically to mean those programs of general usefulness (such as compilers) which are available to all users. These are sometimes called *utility programs*. All machines today have *operating systems* to aid the user (and the operator) in sequencing jobs, accounting, and calling up other utility programs. Operating systems or programs are also called *control programs, supervisors* or *executives*.

Applications are the problems to which a computer is applied; the names for most common applications are self-explanatory, but some are not. A *simulation* is the representation of a real or hypothetical system by a computer process; its function is to indicate system performance under various conditions by program performance. *Information retrieval* is the name applied to processes which recover or locate information in a collection of documents. An *information management system* helps a user maintain a data base, modify it, and get reports from it. It is usually defined as a *general purpose device*; this means that it can accommodate a large range of applications. A *management information system*

supplied to the management of an organization the data that it requires to make decisions and to exercise control. A *report generator* is a program which allows the user to specify in some simple way the content and format of reports which the computer is to produce.

To *run* a program is to cause it to be performed on the computer. Running a program to solve a problem or produce real results (as opposed to debugging) is called a *production run*. Installations in which the user runs his own job are called *open shops*. Installations which have a *computer operator* who runs the programs for the user are *closed shops*. Computers are usually operated in *batch processing mode*, the operator assembles a batch of programs waiting to be run and puts them serially into the computer; output from all the programs is returned in one batch. *Turnaround time* is the time between the user's delivering his job to the center and his receipt of his output. *Time sharing* is a method of operation by means of which several jobs are interleaved, giving the appearance of simultaneous operation. In many timeshares systems, users have individual terminals which are on-line. Such terminals may be located far from the computer; this is *remote access*. This allows users to interact with the computer on a time scale appropriate for human beings--on the order of a few seconds between responses. This capability is called operating in *real time*. Using the computer for frequent interaction with the user in this way is called an *interactive* or *conversational mode* of computing.

Like all electronic devices, computers sometimes break down. The prevention and correction of such situations is *maintenance*. *Preventive maintenance* finds failing components before they actually break down. *Reliability* is the measure of the frequency of failure of the computer. During *downtime* the machine is being maintained or repaired; during *uptime* it is available for normal productive use.

TERM and (LINE)

access speed (32)
Algol (156)
algorithm (140)
analog computer(s) (2)
analyst (137)
application(s) (187)
assembler(s) (171)

batch processing mode (208)
binary number (99)
bit (27)
branching (124)
byte(s) (30)

capacity, storage (25)
card, Hollerith (47)
card punch (60)
card, punched (47)
card reader(s) (48)
cathode ray tube (68)
cell(s) (26)
central processing unit (12)
character(s) (103)
closed shop(s) (208)
Cobol (157)
code checking (145)

coded (102)
coder (142)
coding (142)
column(s) (51)
command(s) (104)
compiler(s) (170)
computer (1)
computer, analog (2)
computer, digital (5)
computer operator (207)
computer, stored program (114)
console (93)
control program(s) (185)
conversational mode (222)
core memory (15)
core(s) (18)
CPU (12)
CRT (68)

TERM and (LINE)

data bank (129)
data base (129)
debugging (145)
deck (52)
digital computer(s) (5)
disc, magnetic (22)
display device (66)
documentation (148)
downtime (228)
drive, tape (77)
drum, magnetic (21)

EAM (86)
EDP (9)
electromechanical accounting
machine(s) (86)
electronic data processing
machines(s) (9)
executive(s) (186)

feedback (111)
file (128)
flowchart (149)
Fortran (156)

general purpose device (196)

hard copy (71)
hardware (11)
heuristic process(es) (163)
higher-order language(s) (153)
Hollerith card (47)

information management
system (194)
information retrieval (192)
input device(s) (45)
input/output equipment (44)
instruction(s) (104)
interactive mode (222)
interpreter(s) (174)
I/O (44)

keypunch (89)

2. The ARPANET

The ARPANET is an operational, computerized, packet switching DoD digital network which provides a capability for terminals or geographically separated computers, called hosts, to communicate with each other.

The elements of the network are the Interface Message Processors (IMP), Terminal Interface Processors (TIP), access lines, host computers and terminals.

The host computers typically differ from one another in type, speed, word length, and operating system. Each terminal or host computer is connected into the network through a local small computer called an IMP (a store and forward packet switch which will accommodate up to four host computers) or TIP (a store and forward packet switch which will accommodate up to 3 host computers and 64 terminals).

The complete network is formed by interconnecting the IMP's through wideband communication lines (50KB) supplied by common carriers. Each IMP and TIP is programmed to receive and forward messages to the neighboring IMP's or TIP's in the network.

During a typical host-to-host operation, a host passes a message to its IMP; the message is passed from IMP to IMP through the network until it finally arrives at the destination IMP, which passes it along to the destination host. A terminal (teletypewriter or CRT) accesses the network through a Terminal Interface Processor (TIP) and sends a message to a host or another terminal. The terminal message passes through the network in the same manner as a host-to-host message.

3. TIP USER'S GUIDE

The following was excerpted from USER'S GUIDE TO THE TERMINAL IMP, Bolt, Beranek and Newman, Inc. Report No. 2183, NIC No. 10916, July 1977 Revision.

2

3

3.1 INTRODUCTION

This report describes the use of a terminal connected to a Terminal IMP (TIP) in the ARPA Network. The report assumes that the user knows how to operate a server Host system somewhere on the network once he becomes connected to that system, and the report defines the procedures and options the user has available to establish that connection.

The ARPA Network, IMPs and TIPs, hardware maintenance, TIP operation, and formats and protocols are not described here. The bibliography lists the relevant documents.

It should be noted that the TIP is designed to support interactive, asynchronous terminals. It cannot normally support synchronous devices, or devices whose input (to the TIP) characteristics are significantly different from a human typist. Thus the connection of computers, paper tape readers, polled circuits, buffered terminals, and so on to TIP ports is likely to be difficult or impossible.

3.2 THE TIP COMMAND FORMAT

The user at a terminal will at various times be talking directly to his TIP instead of to the remote Host. A typical message of this sort might look like:

@OPEN 15

Such a command always starts with symbol @ and ends with either a linefeed or a rubout, depending on whether the user is satisfied with the command or wishes to abort it. On 2741 terminals the return key transmits carriage-return/ linefeed to the TIP and ASCII terminals are normally operated in a mode where typing a carriage-return is interpreted as carriage-return/linefeed; both can be used to terminate TIP commands in addition to a linefeed alone. The only exception to this rule is the specific command

which inserts an @ in the data stream to the Host. Commands may occur anywhere, and need not start on a new line. Upper and lower case may be freely intermixed in the command.

Between the @ and the linefeed there will typically be one or more words to identify the command, perhaps followed by a single parameter. The TIP is not very sophisticated, and thinks the only important thing about a word is its first letter. This permits the user to abbreviate a bit; the more usual rendering of the first example might be:

```
@O 15
```

Once the user has started typing the parameter of a command the old value of the command will have been destroyed, and cannot be recovered by aborting the command.

Almost without exception the effect of a TIP command is to set a parameter or mode for the terminal. Even apparently direct commands like

```
@OPEN 15
```

(which initiates an elaborate exchange of messages resulting in a connection to the remote Host system) actually set a mode flag to request the appropriate action when the TIP is free to undertake it. To understand the TIP behavior is really to understand the complete set of parameters and the commands to change them. Normally, any parameter can be changed at any time by the user at his terminal. Exceptions occur when the user tries to change connection parameters on an open connection. An @OPEN 13 executed while talking to Host 15 would generate the error message "Can't" (the connection to Host 15 must be closed before a connection can be opened to Host 13).

Commands often consist of several command words; for example,

```
@DEVICE CODE ASCII
```

Such commands may be abbreviated; for example

```
@D C A
```

The spaces are required: @DCA is not a legal command. Upper and lower case letters may be freely intermixed.

An unusual variation in command format is to place a number between the @ and the first word of the command. In this case, the command is not meant for the terminal typing but for the terminal attached to the port of that number on the same TIP as the user.

3.3 TYPICAL USE OF THE TIP

In the normal course of things, a user will go through four more or less distinct stages in typing into the net. First, he will be concerned with hardware-power, dialing in, etc. Then he will establish a dialogue with the TIP to get a comfortable set of parameters for this usage. Next, he will instruct the TIP to open a connection to a remote Host; and finally, he will mostly ignore the TIP as he talks to the remote Host. The following sections will describe these stages in more detail.

3.4 ESTABLISHING PARAMETERS

In stage two, the user is concerned with initializing parameters. The naive user should skip stage two and accept the TIP's default parameters until an obvious problem arises. The following questions are answered in stage two.

1. *When shall the TIP send off messages to the remote Host?* Here there are several options. (The TIP is initialized to send on every character, which is simple but inefficient.)

```
@TRANSMIT NOW
@TRANSMIT ON MESSAGE-END
@TRANSMIT ON LINEFEED
@TRANSMIT EVERY
```

TRANSMIT NOW causes the message currently being accumulated to be sent as soon as possible. **TRANSMIT ON MESSAGE-END** causes a message to be sent as soon as possible after an ASCII DC3 (control-S) is encountered. **TRANSMIT ON LINEFEED** causes a message to be sent as soon as possible after a linefeed is encountered. Additionally, both **TRANSMIT ON MESSAGE-END** and **TRANSMIT ON LINEFEED** cause characters to be accumulated in the message buffer until it is almost full. **TRANSMIT EVERY** causes a message to be sent as near as possible to every th character. The command **TRANSMIT EVERY 0** will reset the TIP to its initial state, transmitting every character. If the parameter to **TRANSMIT EVERY** n is a large number (e.g., 250) the TIP will save up as many characters as it can before sending a message, but does not offer any guarantee that the total number specified can be buffered.

2. *Who shall echo and when?* Echoing is a complex problem, without any neat solution. We have chosen to give the user the means to tell the TIP how he or she wants it done, since it is hard to guess correctly in advance. Basically, echoing can occur at the terminal hardware, in the TIP, or in the remote Host. The corresponding TIP commands are:

```
        @ECHO HALFDUPLEX      (Echo at the terminal)

        @ECHO LOCAL
or      @ECHO ALL             (Echo at the TIP)

        @ECHO REMOTE
or      @ECHO NONE            (Echo at the remote Host)
```

Each of these commands sets the internal state of the TIP to the correct mode, and in addition sends the appropriate TELNET command to the Host if the connection is open when the command is given. The commands ECHO ALL and ECHO NONE are now exactly equivalent to ECHO LOCAL and ECHO REMOTE; they have been left temporarily while users become accustomed to the new usage.

In the ECHO REMOTE mode, although characters for the remote Host are not echoed, the TIP will echo commands. Network protocol specifies that echoing shall start out in the @ECHO LOCAL or @ECHO HALF modes. The TIP will try to guess from the terminal type which of the two is appropriate. The goal of the echoing strategy is to avoid the unreadable alternatives of the blank page and the doubling of every character. The naive user is advised to accept the TIP's default parameters until trouble of this sort arises. A 2741 cannot change echo mode and is always halfduplex.

Because both the TIP and the remote Host need to agree on who is to echo the characters, the TELNET protocol provides a mechanism whereby the remote terminal user may instruct the serving Host to set a particular mode and vice versa, the remote Host may instruct the TIP. As noted above, this happens automatically when the command is given by the user while the connection is open. But the Host too may issue such commands to the TIP, and the TIP is not able to refuse the request (even for 2741 terminals). If the connection becomes established in an undesirable mode, the user may change this by one of the echo commands above.

3.5 CONNECTION TO REMOTE SITES

In stage three, the user is concerned with establishing a connection to a remote site.

```
@OPEN 15
```

This amounts to "set the Host number parameter" and "add the user to the queue of users waiting for the TIP's connection mechanism".

Currently, Host addresses are single decimal numbers, as shown. Additionally the TIP accepts a two number Host address as follows:

<Host # on the IMP>/<IMP #>

Leading zeros are not required. Therefore, for example, @Host 44, @Host 0/44 or @Host /44 are all identical.

Connecting to a Host requires establishing a bi-directional link, so that the terminal can send characters to the Host and vice versa. The request to connect to a Host is thus really a request to establish both transmit and receive sections.

When the user reaches the head of the queue waiting for the TIP's "connection" mechanism, the TIP will type "Trying...".

Following the message "Trying", the user will receive some of the following messages:

```
Open              success*
Net Trouble       remote site cannot be reached
Refused           remote site up but refusing
Host Scheduled    Down Until Sat. at 1850 GMT
                  Host will be back up at time and date
                  indicated
Host Unscheduled  Down Until Sat. at 1850 GMT
                  Host will back up at time and date
                  indicated
Host not responding
                  Remote site not up, unknown when up
                  service will resume
ICP Interfered With
                  The Host has not performed the ICP
                  correctly and the TIP has refused
                  to open a connection.
```

[*"Open" indicates both halves of the TELNET connection have been opened simultaneously. Sometimes "Open R" followed by "Open T" (or vice versa) will be printed; this too indicates both halves of the connection have been opened, but not simultaneously. If only "Open R" or "Open T" is printed, then the server Host has failed to open one half of the TELNET connection.]

The connection mechanism will run continuously until a state described above occurs. This can be annoying when the remote site is obviously not going to respond. The command

@CLOSE

will abort the current connection attempt. The user is then free to reattempt to open the connection or to attempt to open a connection to some other Host.

The TIP's connection mechanism has caused users some problems. Perhaps a discussion of what the connection mechanism is doing and how it works will alleviate some of the grief.

First of all, users attempting to connect to different Hosts will never interfere with each other, although users simultaneously attempting to connect to the same Host will be serviced serially.

For the user, opening proceeds in three phases. In the first, the user is queued up waiting to "get" the TIP's connection mechanism. In the second, the user has gotten the TIP's connection mechanism and is beginning the connection sequence. In the third, the user has completed the connection sequence and is waiting for the Host to open up the actual data connections. Many of the problems stem from the fact that only one user may be proceeding through phase 2 at a given time to a given Host. Hence the the TIP types out "Trying" when you get off the queue and the connection mechanism begins trying to open your connections. Thus the "Trying" message signifies the transition from phase 1 to phase 2.

3.6 USE OF REMOTE SITES

In stage four, the user is normally talking to the Host without concern for the TIP. All the TIP commands are still available.

One command that will eventually be of interest here is

```
@CLOSE
```

This command starts the shut-down procedure. The TIP will echo "Closed" when the process is finished. "Closed" indicates that the server Host agreed to close both halves of the TELNET connection simultaneously. If the halves of the connection are closed one after another, "Closed R" followed by "Closed T" (or vice versa) will be printed. If only one of "Closed T" or "Closed R" is printed, wait a minute and the TIP will force the other half of the connection to be closed.

The TIP does not know how to log you out of the remote Host. You must do this yourself before closing the connection.

To return to the nominal mode of carriage-return/linefeed, the command

```
@INSERT LINEFEED
```

should be executed.

If at any given time the user types characters faster than a server Host will take them from the TIP, the TIP discards characters it can not buffer and echos them with an ASCII BEL (on a 2741, the type element is wiggled).

The user may sometimes use a server Host with which it is desirable not to have @ be a TIP reserved character. The user can change the character which introduces TIP commands using the command

```
@INTERCEPT #
```

By typing @INTERCEPT followed by a decimal number representing an ASCII character, the user changes the TIP command character for his device to the ASCII character represented by the number. The INTERCEPT ESC command resets the TIP command character to at-sign (@). Thus,

```
@INTERCEPT 42
*INTERCEPT ESC
```

changes the TIP command character to asterisk (*) and back to at-sign (@) assuming the device was in the nominal mode (@) before the first command was executed.

If the user attempts to change the intercept character but fails to type a valid decimal number (or a character string beginning with E) the TIP will type the diagnostic "Num" and will set the intercept character to at-sign.

3.7 CONNECTION LOSS AND RESTORATION

Starting with TENEX Hosts running Software Version 1.32, if TENEX halts, the TIP will notify users connected to it of this fact by typing "Connection Suspended". [Other Hosts may also implement the mechanisms which will allow the suspension and restoration of connections.] At this point the users are free to do one of two things. First, they can wait till TENEX restores service, in which case the TIP will type out "Connection Restored" (or if after the the service interruption the connection could not be restored, the TIP will type out "Host broke the connection"). Alternatively, the user is free to open a connection to any other Host, in which case the TIP will invisibly close the TENEX connection. It is also important to point out that if a user just leaves his terminal unattended across a TENEX service outage without releasing the connection (any network-related command such as @H, @O, @N, @C will do the job) his job, directory, etc., are left at the mercy of anyone who acquires that terminal.

3.8 TIP NEWS AND USER FEEDBACK

There is frequently information which the group developing and debugging the TIP system wishes to convey to TIP users. For instance, when a bug is detected, we may wish to warn users not to use a certain feature until the bug is fixed. When a minor improvement is made, we may wish to notify users. Further, there is frequently news about the state of the network or the state of a particular Host which should be conveyed to TIP users. Finally, TIP users may wish to communicate with the TIP development staff or the Network Control Center staff about problems or suggested improvements for the TIP or the network. Consequently, we have constructed a mechanism which we hope will provide for communication in all the above directions. This mechanism is the Network Virtual TIP Executive (a version of the Resource Sharing Executive being developed by the BBN TENEX group). To activate this mechanism, the TIP user may give the TIP command @N. This command causes the TIP to perform the necessary protocol to make a connection to the Network Virtual TIP Executive which resides on several of the network TENEX systems. Once the Network Virtual TIP Executive has been activated, we think its operation is self-explanatory. Presently available features within the Network Virtual TIP Executive are a Network News feature, a Host Status feature, and a "Gripe" feature. The latter provides users with a mechanism for sending messages to the TIP development or NCC staffs. We recommend that TIP users get the network news at the beginning of every TIP session.

The TIP will normally prompt the user to consider reading the news by typing the message:

```
Latest net news DATE
Use "@N<cr>" followed by "netnews<cr>"
```

at some point(s) during the user's session. The point chosen is at the time of terminal recognition for "hunting" terminals, or at each time a connection is closed for "non-hunting" terminals.

When a user issues an @N command, the TIP requests support from all cooperating servers. Thus, the user should be able to reach a news facility, somewhere, almost all of the time. However, in the event that no cooperating server is available the TIP will time out the @N command in about thirty seconds. An @C command will abort an @N immediately.

Of course, TIP users with an immediate need for communication with the NCC or TIP development staffs should telephone (collect) the Network Control Center (617-661-0100). Users with general questions questions about network usage (How do I find out if Host X is ever going to be up again? What's happening with a Host/Host protocol for graphics?) may also call the NCC.

3.9 UNUSUAL USES OF THE TIP

The "usual" use of a TIP is to connect one of the terminals which the TIP supports to a remote Host. We have tried to make this operation as easy and natural as possible for the user. "Unusual" uses of the TIP are such things as connecting a non-standard terminal, talking terminal-to-terminal, or using unusual protocols. Such uses are possible, but within the constraints of the TIP's size it has not always been feasible to make them easy.

3.10 THE DIVERT OUTPUT COMMAND

It is possible (with some care) to divert the output intended for one terminal to another terminal. Presumably the second terminal has a desired feature, like hard copy or high speed. The command

```
@16 DIVERT OUTPUT
```

will cause all remotely generated output to be diverted from the terminal on which the command was typed to Terminal 16. This state will continue until any other command is executed at the diverting terminal. (Executing another command does not do a Give Back.) Local-echoing will not be diverted, and input may proceed at the diverting terminal.

This mechanism is not natural to the structure of the program. In particular, the buffer allocation structure becomes confused if the diversion aborts while output is in progress. Chained diversion will also confuse the TIP. Please don't try these things: all that will happen is that one or both terminals involved will stop responding. In fact, in general, we discourage use of the DIVERT OUTPUT command and suggest printing to TIP devices other than the user's own terminal by programs such as the TIPCOPY program which runs on many network TENEXs.

3.11 EDITING

At the moment the only editing command available is

```
@FLUSH
```

which deletes all the characters in the TIP's input buffer. In current practice this command is used to clear out any odd characters stuck in a TIP prior to giving an OPEN command. (The TIP accumulates characters typed in during periods of non-connection and sends them as the first data over a new connection.)

To abort a command, type a rubout or merely make something about the command illegal; for example,

```
@O X
```

The "X" aborts the OPEN command.

3.12 THE RESET DEVICE COMMAND

The reset device command (@RESET) will restore a port to its initial state. Implicit in this command is the concept of a "permanent device." Currently, "permanent" is the same as "non-hunting" but this will not always be so. Permanent devices are those requiring a particular set of parameters that is not expected to change or perhaps for which the TIP cannot hunt. Examples are the IMLAC (no high speed hunt) and a line printer (no input possible). Everything is reset as follows:

a. All network connections and attempts to open connections are cleared;
b. If this device is captured by any other device, that state is cleared and if this device is capturing any other devices, that state is cleared;
c. Input buffers are initialized;
d. "Connection" parameters are reset to a default state* (i.e., echoing all, terminate every character, no insert linefeed, no device wild, intercept all);
e. The code, rate, and size are set to the hunting state.
f. The port's data set is hung-up.
g. Sets the terminal to make connections under OLD TELNET protocol.

For permanent devices the RESET command does only a, b, c, f, and g above and prints "TIP NAME" followed by the TIP version number and the octal port number.

3.13 COMMAND SUMMARY

[* *DENOTES A DECIMAL NUMBER UNLESS OTHERWISE STATED.*]

BINARY INPUT END

Leave 8-bit binary input mode

BINARY INPUT START

Enter 8-bit binary input mode

BINARY OUTPUT END

Leave 8-bit binary output mode

BINARY OUTPUT START

Enter 8-bit binary output mode

CLEAR DEVICE WILD

Set device to be unwild

CLEAR INSERT LINEFEED

Stop inserting linefeed after carriage-return

CLOSE

Close all outstanding connections, or abort current Host login

DEVICE CODE 37

Establish parity computation for Model 37 Teletype

DEVICE CODE ASCII

Establish code conversion for an ASCII terminal

DEVICE CODE EXTRA-PADDING

Establish code conversion for a terminal with slow CR

DEVICE CODE OTHER-PADDING

Establish code conversion for a line printer

DEVICE RATE #

 # is a 10-bit code specifying hardware rate and character size settings

 # DIVERT OUTPUT

Capture device # and divert this terminal's output to it. # is an octal number.

ECHO ALL

Local TIP-generated echo -- TIP echoes everything

ECHO HALFDUPLEX

Terminal-generated echo -- TIP echoes nothing

ECHO LOCAL

Send the Telnet "ECHO LOCAL" character and perform internal E A

ECHO NONE

Remote Host-generated echo for data -- TIP echoes commands

ECHO REMOTE

Send the Telnet "ECHO REMOTE" character and perform internal E N

FLUSH

Delete all characters in input buffer

GIVE BACK

Release control of captured device #. # is an octal number.

HOST #

Simultaneous "@S T H" and "@R F H"

INITIAL CONNECTION PROTOCOL

Start the initial connection protocol

INSERT LINEFEED

Insert linefeed after carriage-returns

INTERCEPT #

Use # as TIP command character

INTERCEPT ESC

Leave 7-bit binary mode

INTERCEPT NONE

Enter 7-bit binary mode

LOGIN #

An obsolete form of OPEN

M # #

Mag tape command # with argument #

NETWORK-VIRTUAL-TIP-EXECUTIVE

Connects the user to the Network-Virtual-TIP-Executive.

OPEN #

Open a bi-directional connection to the Host decimal address specified.

PROTOCOL BOTH

Simultaneous "@P T T" and "@P T R"

PROTOCOL TO RECEIVE

Manually initiate connection protocol.

PROTOCOL TO TRANSMIT

Manually initiate connection protocol.

RECEIVE FROM HOST #

Establish Host # parameter for manual initialization.

RECEIVE FROM SOCKET #

Establish socket # parameter for manual initialization of connection -- socket # is given in octal.

RECEIVE FROM WILD

Equivalent to "@R F S <any>".

RESET

Reset current TIP port parameters.

RESET NCP

Resets NCP.

SEND BREAK

Send the Telnet "BREAK" character.

SEND COMMAND

Send the command escape character.

SEND SYNC

Send the Telnet "SYNC" character and an "INTERRUPT SENDER" message.

SEND TO HOST #

Establish Host-# parameter for manual initialization of connection.

SEND TO SOCKET #

Establish socket # parameter for manual initialization of connection -- socket # is given in octal.

SEND TO WILD

Equivalent to "@S T S <any>".

SET DEVICE WILD

Equivalent to the commands "@R F H <any>", "@S T H <any>", "@S T S <any>", and "@R F S <any>".

TRANSMIT EVERY #

Send off input buffer at least every #th character where 0<#<256.

TRANSMIT NOW

Send off input buffer now.

TRANSMIT ON LINEFEED

Send input buffer every time a linefeed is encountered.

TRANSMIT ON MESSAGE-END

Send input buffer every time an end-of-message is encountered.

3.14 BIBLIOGRAPHY

Most of the following should be available through your Network Information Center Station Agent or the

ARPA Network Information Center
Augmentation Research Center
Stanford Research Institute
333 Ravenswood Avenue
Menlo Park, California 94025

Specifications for the Interconnection of a Host and an IMP, BBN Report No. 1822 (IMP-HOST section of NIC 7104).

IMP Operating Manual, BBN Report No. 1877.

The Interface Message Processor for the ARPA Computer Network, Heart et al., Proceedings AFIPS 1970 Spring Joint Computer Conference (NIC 4655).

The Network Working Group "Request for Comments" Series: A Set of Working Papers on Host Protocol.

The Network Resource Notebook, NIC 6740.

The BBN Terminal Interface Message Processor (Hardware Manual), BBN Report No. 2184.

Specifications for the Interconnection of Terminals and the Terminal IMP, BBN Report 2277.

ARPA Network Current Network Protocols, NIC 7104.

The Terminal IMP for the ARPA Computer Network, Ornstein et al., Proceedings AFIPS 1972 Spring Joint Computer Conference.

Terminal Access to the ARPA Network: Experience and Improvements, Mimno et al., COMPCON 73, Proceedings Seventh Annual IEEE Computer Society International Conference, San Francisco, February 27 - March 1, 1973.

TIP Users Group Notes, a series of informal notes designed to increase communication among the developers of the TIP, TIP users, and Hosts frequently used from TIPs. To be put on the distribution list for these notes, apply to the NIC Station Agent.

3.15 TERMINALS USED WITH THE TIP

The following terminals are reputed to have worked with the TIP.

A. B. DICK VIDEOJET 9600 LINE PRINTER (2400 bps)
ANDERSON-JACOBSON (models 630 and 841)
ARDS
KSR-35 Teletype
CALCOMP 565
CDI 1030/ "MULTICS" Terminal
DATA 100 (model 73)
DATAPOINT (models 2200, 3000, and 3300)
DELTA TELTERM 2
DIGITAL EQUIPMENT CORP. (models VT05 and GT40A)
EDT 1200
HAZELTINE 2000
LINEOLEX (model A)
MEMOREX 1240
SUGARMAN (model S-4300)
TECTRAN CASSETTE
TEKTRONICS (models 4010 and 4013)
TELETERM (model 1030)
TELETYPE (MODEL 38)
TERMINET 300
TI Silent 700
TYCOM
TRENDATA (model 1000)
UNIVAC DCT 500
VIDEO SYSTEMS (models 1200 and 5000)

We would be pleased to hear of any other terminals that have operated on a TIP.

At the International Conference on Computer Communications held in Washington, D.C., in October 1972, we had the opportunity to personally test a number of the above terminals with the TIP. As a result of this experience we now hold opinions as to the methods and difficulty of connecting a number of these terminals to the TIP. We suggest you call the Network Control Center to be put in touch with someone about this subject.

3.16 NEW TELNET PROTOCOL IMPLEMENTATION

The NEW TELNET protocol has been implemented in TIP systems numbered greater than 350. Wherever possible, the TIP should appear just the same to the user, but some differences do occur. Some background information is perhaps useful. Both protocols are based on the concept of the NETWORK VIRTUAL TERMINAL; each also defines TELNET commands that may be sent to instruct the other side, for instance, concerning echo mode. In the OLD TELNET, such commands are unequivocal demands, and cannot be refused even if the state is undesirable. A Host can of course undo the state with a counter command of its own, but it first must understand what has happened; in addition, there is no guarantee that the same sequence won't occur again. The NEW TELNET protocol defines a minimal default set of parameters, then allows each side to negotiate for non-standard options. No Host is required to accept an undesirable state; in fact, a Host can simply refuse to do "whatever the other side requested" and need not even understand what it has just refused. This protocol should allow sophisticated Hosts to define complex options without burdening simpler Hosts.

A direct implication to the TIP user is that a "command" is now a "request" and may be refused. It is possible, for instance, to ask for remote echoing (local echo is the default) and not get it. In practice, systems provide the same, or increased, capabilities under the new protocol.

On the positive side, the TIP now acts as an advocate for the remote terminal user. In the case of a halfduplex terminal like the 2741, the TIP can refuse to accept the remote echo option and the user no longer needs to cope with double echoing when the connection opens.

The NEW TELNET protocol specifically affects the following aspects of the TIP.

3.16.1 Getting In and Out of NEW TELNET Mode

The commands @OLD TELNET and @NEW TELNET set the TIP to expect OLD or NEW TELNET respectively. The default as of this writing is OLD TELNET, and @RESET

automatically returns the device to OLD TELNET for all devices. This action also occurs when a non-permanent -- that is, hunting -- terminal hunts to its correct rate.

The TIP saves this mode state for each device. When a command is given, for say echo or binary, the TIP determines which protocol the device is using and responds appropriately.

3.16.2 Echoing

The TIP echo commands have the same meaning in NEW TELNET as in OLD TELNET, as revised above. In addition, the TIP now remembers the mode that the user desires, even though it may not be possible to be in that state at the time. If the command is given when a connection is open, the effect is the same as OLD TELNET, with the TIP sending the request to the remote Host. If, however, the command is given while the connection is closed, the TIP will save the user's request and send it to the remote Host automatically, each time a connection opens.

Non-hunting terminals will continue to be pre-initialized as arranged by the designated site representative and the Network Control Center. Most hunting terminals will default to remote echo mode; however, physical halfduplex terminals (2741s) will not accept remote echoing ever, whether hunting or not.

3.16.3 Binary Input and Output Options

Binary input and/or binary output are available under the NEW TELNET protocol. The commands are:

```
@BINARY INPUT START      begins binary input mode
@BINARY INPUT END        ends binary input mode
@BINARY OUTPUT START     begins binary output mode
@BINARY OUTPUT END       ends binary output mode
```

As with the echo option, the TIP sets an internal flag for the mode the user wishes, and attempts to get the remote Host to agree. This feature is not yet automatic: the TIP will not send a request unless the connection is open at the time of the command, but the TIP will passively accept a request for binary mode from a remote Host until the user ends binary mode. This may be useful for paper tape punches, for example.

3.16.4 Log or Open a Connection

The @OPEN and @LOG commands will automatically determine the appropriate socket for the ICP and the TIP will interpret the correct TELNET protocol.

> NOTE: when connections are made directly (and not through the LOG or OPEN commands), the user should take care to be in the proper TELNET mode. An example might be a lineprinter or paper tape punch set wild. Any program connecting to such devices must use the same TELNET protocol that the device is set to.

3.16.5 Commands Available Under NEW TELNET

These commands send the following "New Telnet" function codes: Abort Output (AO), Are You There (AYT), and Interrupt Process (IP). To use these commands type:

```
@SEND ABORT OUTPUT

@SEND ARE YOU THERE

@SEND INTERRUPT PROCESS
```

The following commands cause the TIP to send the "New Telnet" function codes for Erase Character and Erase Line to the remote Host; if no connection is open when the commands are given, they have no effect.

```
@SEND ERASE CHARACTER

@SEND ERASE LINE
```

The command:

```
@ECHO REMOTE CONTROLLED
```

requests the TIP to negotiate with the remote Host for Remote Controlled Transmission and Echoing mode (RCTE) for that port. In this mode, the TIP does most of the echoing, but under instructions from the Host. As examples, the Host could instruct the TIP to suppress echoing of a password or have the TIP hold off echoing temporarily while the Host performs command completion (available on some Host systems). Like the other New Telnet echo modes, the TIP will remember that RCTE was requested and automatically

negotiate for the option when a new connection opens. To leave RCTE mode, simply give the TIP one of the other echo mode commands. Physical half-duplex terminals by their nature cannot use RCTE mode. NOTE: few Hosts implement RCTE mode at present and neither they nor the TIP are guaranteed to work perfectly as yet. Reporting any problems, questions, or comments to the NCC (via GRIPE) would be especially helpful.

4. TENEX

TENEX is a time-sharing operating system, built around the DEC PDP-10 computer. A time-sharing operating system is one that allows many people to use a computer at the same time; to create and run programs or share other users' programs or use standard programs (subsystems) provided with the system. While being able to share programs and data the user is also able to keep information confidential.

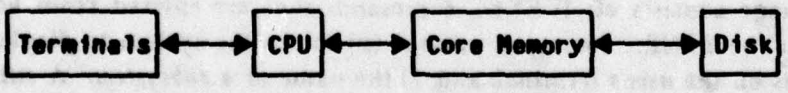
TENEX supports a number of subsystems, each of which does a particular job. Some subsystems are highly self-contained, and require little knowledge of the rest of TENEX; others do specific jobs such as editing or compilation, are typically used together with other subsystems, and require of the user more complete knowledge of TENEX.

Information about programs and subsystems supported by TENEX can be found in directory <DOCUMENTATION>. <DOCUMENTATION> files are catalogued according to type (e.g., .manual, .info, .changes) and files with a .changes type are generally in message file format.

4.1 OPERATING SYSTEM

The TENEX EXECutive (or simply, the EXEC or system) is the *operating system program* that supervises everything the user runs on the computer. The computer can run one program for each user at a time, and the first program run when the machine is started up is the EXEC, which also manages placement of files on the disk, where programs are in core, reading and writing characters on terminals and the line printer.

In its simplest form the *system* looks like this:



and the basic definitions are:

OPERATING SYSTEM

Usually refers to the whole picture above: the terminals, central processing unit, core memory, and disk.

CPU (CENTRAL PROCESSING UNIT)

The CPU follows the instructions (programs) given to it by the user -- it *runs the program.*

CORE MEMORY

Core Memory is where programs are held while the CPU *executes* them. *Data* from the terminals (characters) and from the disk (files) is stored *in core* while it is being worked on. Core memory is made of *bits*; 36 bits make a *word*; 512 words make a *page*.

DISK

Programs and data are stored as *files on disk*. *Program files* may be brought into core to be run. *Data files* may be created and changed (by text editor programs, for example); they may be *copied* to the terminal, to the *line printer*, or duplicated on disk.

The disk is where your work is kept when you are not working on it. Each user may have many files on the disk, and the names of the files are kept in the user's directory.

4.2 EXECutive LANGUAGE

The TENEX EXECutive Language allows the user to access and control the TENEX facilities. The language consists of: 1) EXEC commands that are entered from keyboard terminals, typing ? at the TENEX command prompt (@) causes the system to display a list of TENEX commands on the user's terminal; and, 2) the name of a *subsystem*. A subsystem is a program which was developed to enable users to perform tasks on a computer and any user may start them by just typing their names.

Communication with the EXEC takes the form of a dialogue: the user types a command, the system does what it is told, including filling in parts of the command it knows about, and then comes back for more input from the user. When the EXEC is ready to accept a

command it types an *at sign* (@) -- the user then types a command. Most commands are terminated with a *carriage return* (CR).

Login messages (messages automatically typed on the terminal when a user logs in) from the <SYSTEM> directory notify the user whenever changes are made to existing subsystems or programs. Further, when the EXEC is intialized (a user logs in), it saves the date and time of write of the file <DOCUMENTATION>SYSTEM.CHANGES, which will be compared against the date and time of write for programs run out of <SUBSYS>, <NEWSYS>, or <OLDSYS>. If the program is newer, the message "[Modified program]" is typed. The appropriate .CHANGES file in <DOCUMENTATION> should explain the differences.

4.2.1 Special Characters

In addition to EXEC commands and subsystems, TENEX supports a set of special characters -- namely, the ESCAPE (or ALTMODE) terminal key and control characters.

The ESCAPE (or ALTMODE) Character

The ESCAPE (or ALTMODE) terminal key is labelled ESC (or ALT) on most terminals and is located at the upper left side of the keyboard. It does two things: 1) recognizes and extends a partially typed command or filename, and 2) provides the user with command guide words (where applicable), and filename version information. If the user's partial type-in of a command or filename is not adequate for recognition,the EXEC will output a bell (the terminal will ding) indicating more user input is necessary. Thus, the user may type part of a file name or command and ask the EXEC to supply the rest by typing ESC (or ALT), which is shown in the examples and scenarios as a dollar sign ($).

Control Characters

Control characters signal the system to do something immediately. Control characters are shown by preceding them with an up arrow, e.g., ↑C, ↑A. They are typed by holding down the control key (on the left side of the keyboard) and then typing the character. Thus, the control key is like a shift key except that there is no "control lock" key.

The most important control characters are the following:

↑A (CONTROL A) - DELETE CHARACTER

> If a typing error is made, typing ↑A will cause TENEX to forget the last character typed. Usually a backslash (\) will be typed followed by the bad

character. Example: if the user types A followed by B followed by ↑A (control A) followed by C, the system will echo 'AB/bC' and it will be just as if 'AC' had been typed. On some display terminals when ↑A is typed the bad character simply disappears from the display.

↑C (CONTROL C) - CALL TENEX EXEC

Typing ↑C interrupts whatever is happening and returns control to TENEX for a new command input. ↑C is used to establish initial contact or to start over retyping a command, or to interrupt a running program. Thus, if output is madly scrolling off the screen, it may be interrupted with ↑C.

↑F (CONTROL F) - FILE NAME FIELD RECOGNITION

Typing ↑F causes recognition of a partially typed field of a file name and is useful when one or more of the fields has a long or complicated name. Both ESC and ↑F may be used in combination to perform recognition on the parts of a file name. When typed, ↑F does not print on the terminal. If ↑F cannot recognize the field the terminal "beeps", signalling the user to input more of the file name.

EXAMPLE:

```
@list <doc↑FUMENTATION>xed.m$ANUAL;*(CR)
```

Since ↑F, ESC and (CR) are *not* printed on the terminal, what the user actually sees is:

```
@list <docUMENTATION>xed.mANUAL;1
```

↑R (CONTROL R) - RETYPE CURRENT COMMAND LINE

Typing ↑R causes the the current command line to be retyped. After several control-A's, the line may become unreadable. ↑R will retype the line with all editing carried out up to the point where ↑R was typed.

↑W (CONTROL W) - DELETE CURRENT WORD

Typing ↑W is exactly like typing ↑A except instead of deleting the last character typed by the user, it deletes the last word the user typed. Typing ↑W causes TENEX to print a left arrow (←), indicating the word has been deleted. The left arrow character is shown on a terminal as a single underscore.

↑X (CONTROL X) - DELETE LINE

Typing ↑X at any point in a command will delete the entire command. This is useful when the user has made an error that can no longer be corrected with ↑A or ↑W. When a ↑X is typed TENEX types ↑X to indicate that the line has been deleted. The DEL terminal key can also be used to delete the entire line typed. Typing the DEL key causes TENEX to print XXX to indicate the line has been deleted and returns the user to EXEC.

↑T (CONTROL T) - Interrupt For Status

Frequently a user would like to know whether his program is making progress. Typing ↑T causes TENEX to provide status information to the user without stopping or interrupting his program. An example of a ↑T interrupt typeout:

```
RUNNING at 4263 LOAD AV. = 14.85, USED 0:01:46.2 in 3:39:30
```

4.3 FILE SYSTEM

The TENEX file system provides 1) access to and control over named files which contain data or text or a program, and 2) a means for storing and retrieving information. In order to bring an EXEC command to bear on a file, one must identify the file in question by giving its *file designator (or file name)*. Basically, the parts of a file name which are important immediately are:

```
<DIRECTORY>NAME.EXTENSION;VERSION
```

The *directory* is the general place under which all of a user's files are listed. A directory name is usually the last name of the user or the name of the user's group or project. If the directory name is not included as part of the file name, the system will assume the directory name of the logged in user.

EXAMPLE:

```
NAME.EXTENSION;VERSION
```

To access files in other directories the user must name the directory as part of the file name. In file name specifications, directory names begin and end with opening and closing angle brackets:

EXAMPLE:

```
<DIRECTORYNAME>NAME.EXTENSION;VERSION
```

TENEX will perform recognition on a directory name if a partially typed name is terminated by ESC (or ALT) or ↑F.

The *name* is the next part of the file name. Names may be up to any 39 letter and digits. The name is frequently all that need be typed to refer to a file. As with directory names, TENEX will recognize file names when they are terminated with ESC (or ALT) or ↑F. The user has total control over file names in his directory and can construct file names that require little input for recognition.

The *extension* is a way to have several related files with the same file *name*. Frequently it is desirable to create several files with the same file name, containing perhaps related information. TENEX file extensions makes this possible by allowing the user to create the extension. Extensions consist of a period followed by any combination of up to 39 letters and digits. Although TENEX regards all files as having extensions, the user can specify a null extension (one having no characters) by typing just the initial period.

The *version* numbers carry the idea of file extensions one step further and put it to use for a different purpose. Version numbers allow the user to create multiple disk files with the same combination of file name and extension. The intent of version numbers is to keep track of successive, modified versions of a single file. The system will usually assume the highest version number if it is left off (with certain desirable exceptions noted in the Basic Commands section).

Examples of File Designators:

```
REPORT1.;1
REPORT1.;2
REPORT.RNO;1
<BLACK>BANANA.;1

LPT:      (special designator meaning the line printer)
TTY:      (special designator meaning the terminal)
XGP:      (special designator meaning the Xerox Graphics Printer)
```

The structure of TENEX file name reflects the convenience and uniformity of the TENEX file system, most of which would be lost if the user had to enter all of the file name each time he/she accessed a file.

Certain TENEX commands can address several files at one time. For example, one can APPEND a series of files onto an existing file, or DELETE several files in one operation.

4.3.1 Group File Access

TENEX provides a mechanism whereby users can form groups for the purpose of file sharing. This is useful, for example, where several users are collaborating on a common task or project and wish to share the files they are creating. Each group is assigned a number. For each user, TENEX records a series of these numbers indicating what groups he belongs to. Likewise, for each file directory, TENEX records numbers indicating which groups have access to the files it contains. Thus a given user can belong to a number of groups. He can make his directory available to one or more groups. The assignment of group numbers to users and to file directories is done by the TENEX system operators.

4.4 EXECutive COMMANDS

TENEX EXECutive commands are made up of three components: *keywords*, *noise words*, and *arguments*. Each command begins with an initial keyword that identifies its main function; it may contain further keywords that select options. Noise words (always enclosed in parentheses) make TENEX commands more understandable to the user, but have no effect on command execution. Arguments, such as user or file names, tell what a command is to act on.

Example of the three components of a TENEX command:

```
@APPEND   (FILE LIST)   CHAP.2;1      (TO)        CHAP.1;1
(keyword) (noise word)  (argument)  (noise word)  (argument)
```

4.4.1 Command Fields

Each keyword or argument that the user must specify begins a *field*. The field runs up to the next keyword or argument or to the end of the command. Thus, a TENEX field contains one argument or one keyword, followed possibly by one or more noise words. The *value* of a command field is the particular choice of keyword or arguments that begin it. Thus, in example above:

```
@APPEND (FILE LIST) CHAP.2;1 (TO) CHAP.1;1
```

The values of the three fields are: APPEND, CHAP.2;1, and CHAP.1;1.

4.4.2 Command Input and Recognition

TENEX allows three styles of input. They can be intermixed freely within a command with a separate style applied, if desired, to each field. Further, commands may be typed in either UPPER or lower case, or a combination of UPPER/lower case.

Full Input -- The user can type any field in its entirety, spelling out fully the keyword or argument and the noise words (if any) with their enclosing parentheses. This style is laborious but safe for a new user. (In the following examples user input is shown in *italics*.)

@*append (file list) part2.;2 (to) part1.;1(CR)* [Old Version] *CR*

Abbreviated Input -- The user can omit the noise words from a TENEX command field; he can shorten the keyword by typing in enough characters (terminated by space) to make it unique from other commands acceptable in that context. If the user enters insufficient characters for unique identification, TENEX will type ? and abort the command. This style allows experienced users to type commands very concisely.

@*ap part2 part1(CR)* [Old Version] *CR*

Recognition Input -- The user can enter initial command characters as above, but terminate with ESC (or ALT) instead of space. TENEX will type back the remainder of the command (or keyword) *together with any immediately* following noise words -- i.e., everything through the end of the current field. Recognition input is helpful to less than fully experienced users because TENEX verified that the intended command has been given and supplies clues to what the user is expected to type next. (User input is shown in *lowercase italics*, ESC is shown as $.)

@*ap*$PEND (FILE LIST) *part2*$.;2 (TO) *part1*$.;1 [Old Version] *CR*

When using ESC (or ALT), insufficient character input causes TENEX to merely ring the terminal bell and wait for more input.

With certain frequently used commands, fewer than the minimum characters are accepted by convention in either *Abbreviated* or *Recognition* input. An example is *LOG*, which is taken to mean *LOGIN* even though *LOGOUT* also begins with *LOG*.

4.4.3 Command Terminators

Space, ESC (or ALT), and carriage return, shown as (CR), can terminate commands. Although (CR) is the standard TENEX command terminator, there are the following exceptions.

(CR) terminates last field. With most commands, TENEX will begin execution immediately if the user terminates the last field with (CR) instead of space or ESC.

Confirming (CR) needed. Some commands, notably those that change or destroy information, always require a final (CR), even if (CR) ended the last field. With these commands, TENEX reminds the user that final confirmation is needed by typing [Confirm].

No terminator needed. With certain 'harmless' commands, such as queries about the system, no terminator is required; TENEX will take action as soon as the last field is entered.

Subcommands to follow. Certain TENEX commands can be followed by optional subcommands. Typing comma (,) before a terminating CR tells TENEX that subcommands are to follow. TENEX types @@ and waits for subcommand input. Typing ? at @@ causes TENEX to list the available subcommands. When all subcommands have been entered, two CR must be typed to terminate input of the command.

4.4.4 Basic Commands

The commands given below form a basic set that will allow the new user to get started. A list of *ALL* of the commands supported by TENEX are available by typing ? at the TENEX command level (@). Typing ? and using the ESC (or ALT) terminal key will help in learning these and other commands.

We will show commands in both *recognition* and *abbreviated* form. User input is shown in lowercase; system output in UPPERCASE, or enclosed in square brackets ([]). ESC (or ALT) is shown as $. CARRIAGE RETURN (or RETURN) is shown as (CR). Remember, typing ESC (or ALT) causes the system to recognize the rest of the command and to prompt the user for the next thing to be typed.

```
@logSIN (USER) username$ (PASSWORD) password$ (ACCOUNT) account(CR)
or
@log username password account(CR)
```

A user name (or group name), a password, and a account name or number are necessary to use the LOGIN command. When the user types his/her password it does not appear on the terminal. LOGIN is the first command that the user should type after the host computer greeting line and the 'at-sign' (@) has been output by the system.

Should the user be unsure of his/her account name or number, typing ESC (or ALT) after typing in password space will cause TENEX to print the account. Typing (CR) then terminates the LOGIN command. TENEX records the LOGIN and prints login information together with system messages and notification of new mail, if it exists, the date and time of the user's last login and information about other jobs that may be logged in under the user's name.

EXAMPLE:

```
ISI-TENEX 1.34.7 ISI-SYSTEM-A EXEC 1.54.3
@log smith  123456(CR)
JOB 9 on TTY9 27-JUL-78 09:09
PREVIOUS LOGIN: 26-JUL-78 08:01
TENEX WILL GO DOWN THU 7-27-78 2200 TIL FRI 7-28-78
0500 FOR PREVENTIVE MAINTENANCE
[YOU HAVE NEW MAIL]
@
```

LOGIN.CMD File

Since many different types of terminals are supported by TENEX, the terminal type should be declared (and other mode settings, depending on the terminal type) as soon as LOGIN is completed. One way to set terminal parameters is to manually type them in each time you log in. Another way is to create a LOGIN.CMD file (using any text editor) and have TENEX take over the task of setting terminal parameters automatically by reading and executing the LOGIN.CMD file each time you log in.

The commands in the LOGIN.CMD file must appear exactly as you would type them directly to the EXEC. If any command errors appear in the file, the system will process all the commands up to the one in error, print a message telling the user it found an error and abort the file. As with any other text file, the LOGIN.CMD file may be edited to correct errors, or to add or delete commands as the user desires.

When a job is detached the terminal parameters are disabled. Use the REDIRECT

command to reenable the LOGIN.CMD commands immediately after attaching the job (i.e., @redirect login.cmd(CR)).

@logoSUT(CR)
or
@logo(CR)

The LOGOUT command is used to sign off from the system. The system prints LOGOUT information, including the number of jobs logged in under the user's name should they exist. At LOGOUT time, mailchecking is done. If mail exists, EXEC tells you and asks for LOGOUT confirmation.

EXAMPLE:

```
@logo(CR)
 TENEX WILL GO DOWN DAY DATE HOUR TIL DAY DATE HOUR FOR
 PREVENTIVE MAINTENANCE
 Killed Job  , USERNAME, ACCOUNT, TTY   AT DATE, HOUR,
 Used COMPUTER TIME IN CONNECT TIME
```

After logout the terminal will not respond until you make a connection to your host computer.

@sy$STAT
or
@sy(CR)

A nice command that shows how long the system is up, who is on, and what they are doing. See what your friends are up to. This is one of the commands that does not require a terminator if recognition (ESC) is used.

@wh$ERE (IS USER) directory name(CR)
or
@wh directory name(CR)

The WHERE command is used to obtain status of a single user and tells you his TTY number, JOB number, and if he is at TENEX EXEC or running a subsystem. It also will tell you his port number if he is logged in from a TIP port and the directory name if he is connected to another directory.

```
@di$RECTORY(CR)
or
@di(CR)
@di filename(CR)
@di filename,(CR)
@di *.extension(CR)
@di name.$(CR)
```

Typing DI$(CR) or DI(CR) shows on the terminal a list of all the files in the *user's directory*. Typing DI FILENAME(CR) shows the full name of the single file; DI FILENAME,(CR) shows the single filename and puts the user into subcommand mode (shown as @@). Typing ? lists the directory subcommands that are available, each of which is terminated by (CR). Two (CR)'s terminate subcommand mode.

Typing DI *.EXTENSION(CR) lists all of the files in the user's directory with the named extension; typing DI NAME.$(CR) lists all of the files in the user's directory with the name and all of the extensions associated with it.

```
@qfd(CR)
or
@qfd filename(CR)
@qfd <directoryname>filename(CR)
```

This is the Quick File Directory command that allows a user to get full file information on all of the files or a single file in his or another user's directory (to which he has access), i.e., filename, versions, protection, size in pages, date of creation, author, last read date, last reader. This is a quick version of the DIRECTORY command using the VERBOSE subcommand.

```
@del$ETE filename(CR)
or
@del filename(CR)
@del filename,filename,filename(CR)
```

Tells the system to delete the named file from disk and picks the LOWEST version number if the version number is left off. The DELETE command allows a user to delete several files in one operation and requires a confirming CR. When the user logs off (or EXPUNGEs) the deleted files go away forever (although there may be back up copies on tape). Once a week all files are *dumped on tape (this is known as a* FULL DUMP); daily, newly created files are also *dumped* on tape (this is known as an INCREMENTAL DUMP) and if a user DELETES and EXPUNGES a file in error it most likely can be restored from

either the FULL or INCREMENTAL dumps by sending a message to the ACTION directory.

If a file *disappears* from the user's directory, chances are that the file had not been read or written for approximately a month and has been *ARCHIVED*. The user should use the *INTERROGATE* command (see TENEX FILE ARCHIVAL/RETRIEVAL SYSTEM section).

```
@undSELETE (FILES) filename(CR)
or
@und filename,filename,filename(CR)
```

Before the user logs out or EXPUNGEs, one or several deleted files may be brought back with the UNDELETE command. Most useful (in addition to mistakes) if the user wishes to delete all versions of a file and then bring back just the latest one. Example:

```
@delETE memo.;*(CR)   ("*" stands for "use everything")
@undELETE memo.;(CR)  (system picks latest version)
```

```
@tySPE (FILE) filename(CR)
or
@ty filename(CR)
@ty filename,(CR)
@ty <directoryname>filename(CR)
```

Types a file on the terminal. The TYPE command allows a user to input several file names in one operation and has subcommands available. Typing ↑C aborts output on the terminal and returns the user to TENEX EXEC.

This command will not work for files with an extension of .SAV or .REL. The beginning of the file is checked for garbage control characters. If many are found, EXEC lets you know, and asks for confirmation.

```
@listS (FILE) filename(CR)
or
@list filename,(CR)
@list filename,filename,filename(CR)
@list <directoryname>filename(CR)
```

Lists a file, or several files in one operation, from the user's directory or from another directory to which he has access (or a combination of files from own or other directories) on the lineprinter. Typing LIST FILENAME,(CR) allows access to the subcommands supported by the LIST command. Typing ? in subcommand mode (@@) shows the subcommands; CR terminates each

subcommand input and 2 (CR)'s terminate subcommand mode. If the version number is omitted from the filename the system assumes the HIGHEST version.

This command will not work for files with an extension of .SAV or .REL. The beginning of the file is checked for garbage control characters. If many are found, EXEC lets you know, and asks for confirmation.

```
@subsystem name(CR)
or
@msg(CR)
@xed(CR)
@hermes(CR)
```

The user 'calls' (or starts) the various subsystems like XED, MSG or HERMES by typing the subsystem name followed by carriage return. Each subsystem has its own command level and commands. When a user is running a subsystem he is out of TENEX EXEC and must use the subsystem commands. Typing ↑C at any point during use of a subsystem causes an interrupt and the user is returned to TENEX EXEC. If a ↑C was typed accidentally, the user may use the *CONTINUE* command to put him back to where he was in the subsystem before the ↑C was typed.

```
@LinK (TO) username(CR)
```

This command creates a link between the user's terminal and the terminal of the person NAMEed such that anything output to either terminal also shows up on the other. Example: assume that Pat is working in a text editor and Dick links to her (by @LINK PAT). Dick then types on his terminal and this typing is shown on both his and her's. So far this will have had no effect on Pat's TECO work, and if Dick then breaks the link Pat can go on with no interruption to the text editing. If Pat wants to talk back, however, she may hit ↑C to get to the EXEC, type comments to Dick by starting each line with a semicolon so that the EXEC will ignore it, and then @CONTinue the text editor program after the link is broken by either her or Dick.

```
@brSEAK (LINKS)(CR)
or
@br(CR)
```

Used by any party to a link to break the link and return that person's terminal to normal.

```
@;comment...
```

While in the EXEC, any line starting with a semicolon is ignored. This is useful when talking to someone via a link.

```
@maSIL cSHECK (FOR USER) username(CR)
or
@ma c username(CR)
```

The MAIL CHECK command allows a logged in user to check (directory protection willing) to see if another user has new mail. This command is very useful to a user who, though not logged in, wishes to have his/her directory checked for new mail.

```
@ACCESS (OF FILE LIST) <files> (TO) <access path> (IS) <access>(CR)
```

The ACCESS command allows a user to control access to the files in his directory. <access path> is one or more of the following words separated by commas: SELF, GROUP, OTHERS, ALL. The default invoked by (ESC) mode is OTHERS.

<access> is one or more of the following words separated by commas: READ, WRITE, APPEND, EXECUTE, PAGE-TABLE, APPEND, NORMAL, ALL, NONE.

The default is NORMAL, which is the same as READ,EXECUTE,PAGE-TABLE.

Example:

```
ACCE FOO.MAC,*.F4;* G,O R,W,AP
```

```
@tcSOPY (FILES) filename(CR)
or
@tc filename(CR)
@tc filename,filename,filename(CR)
```

A combination of the TYPE and COPY commands. It COPY's a file list to the TTY: It takes no subcommands. This command will not work for files with an extension of .SAV or .REL. The beginning of the file is checked for garbage control characters. If many are found, EXEC lets you know, and asks for confirmation.

4.5 FILE ARCHIVAL/RETRIEVAL SYSTEM

The TENEX EXEC supports a file archival facility. The purpose of the archival facility is twofold:

1) It provides each user with a large, long-term storage capability.

2) It provides the system with a mechanism for freeing online storage.

User files are allowed to "age" in the user's directory for approximately 30 days. Files so "aged", neither read nor written for a month, will be copied several times to the magtape archive, verified, and deleted from the user's directory. The location of the file is recorded (see INTERROGATE below) so that the user will be able to notify the operator of a retrieval request.

4.5.1 The ARCHIVE Command

The ARCHIVE command provides the user with a method for setting archive requests, resetting archive requests, and determining if a file is in the process of being archived.

```
@ARCHIVE <second keyword> (FILE LIST) <file list>

<second keyword> is one of the following:

DELETE
EXPUNGE
FILE
RESET
STATUS
UNDELETE

DELETE, UNDELETE & EXPUNGE give users control over the names in
their archived files directory file )ARCHIVE-DIRECTORY[.;1.

FILE will request archival of the file(s) in the <file list>

RESET will clear archival information in the <file list>

STATUS shows the archival status for the <file list>
```

By typing a comma after <file list>, one can give subcommands to the ARCHIVE command. Current subcommands are:

```
@@DEFERRED
```

This tells the system to archive <file list> at the next time the archive system is run (the default).

```
@@DELETE
```

This tells the system to delete <file list> from disk AFTER they have been archived and verified (the default).

```
@@DON'T ARCHIVE
```

This tells the system to never archive the specified file.

```
@@DON'T DELETE
```

This tells the system to not delete the file from the directory after it has been successfully copied to the archive.

As with other EXEC subcommand modes, two carriage returns confirm the execution of the ARCHIVE command whenever subcommands have been specified.

Example of ARCHIVE command use:

```
@arcHIVE ? ONE OF THE FOLLOWING:
 DELETE
 EXPUNGE
 FILE
 RESET
 UNDELETE
```

```
 archIVE fILE (FILE LIST) graph.apr-78;1,
@@? ONE OF THE FOLLOWING:
 DEFERRED
 DELETE
 DON'T
 IMMEDIATE

@@don't dELETE
@@
@
@archIVE sTATUS (OF FILES) graph.apr-78;*
 GRAPH.APR-78;1  ARCHIVE WITHOUT DELETION PENDING
@
```

4.5.2 The INTERROGATE Command

The INTERROGATE command is used on a fully archived file. Typing

```
@INTERROGATE  <filename>(CR)
```

causes the archive to be interrogated. If the specified file is not found, THAT FILE IS NOT ARCHIVED is typed out; if the file is found, ARCHIVED ON TAPES MMM AND NNN is typed out; if found, DO YOU WANT IT RETRIEVED? is also typed out; if the answer is yes, a message will be printed on a special teletype in the computer room; the operator will periodically check for such requests; when the file has been retrieved, the retrieval will be confirmed via SNDMSG.

As with the ARCHIVE command, typing a comma after the file name (or list) gives the user access to the INTERROGATE subcommands. Two carriage returns confirm the execution of the INTERROGATE command whenever subcommands have been specified. INTERROGATE subcommands are shown in the following example.

```
@inteRROGATE graph.mar-78.;1
 <SIGNAL>GRAPH.MAR-78.;1  archived on tapes 3246 and 3241
 do you want it retrieved? (Y or N)  yes
@
```

or

```
@inteRROGATE graph.mar-78.;1,
@@?
  ALL.FILES
  BEFORE.DATE
  DATES
  DELETED.FILES.ONLY
  DOUBLE.SPACE
  EVERYTHING
  INDIVIDUAL
  LPT
  NO.HEADING
  NUMBERS
  OUTPUT.TO.FILE
  PROTECTION
  REVERSE.ORDER
  SINCE.DATE
  STATUS
  TAPE
  TIMES.AND.DATES
  UNDELETED.FILES.ONLY
  VERBOSE
  CARRIAGE RETURN TO GO

@@vERBOSE
@@

<SIGNAL>GRAPH.MAR-78;1;P775200
  3246   1-APR-78      3241   1-APR-78
do you want it retrieved? (Y or N)  yes
@
```

4.6 Command Summary

In the following summary of commonly used commands, parentheses () indicate prompting (or *noise words*), square brackets [] indicate optional input by the user, a single star * indicates subcommands are available, and double star ** indicates system defaults. Type ? at @ to see all of the commands available; type the command and use ESC (or ALT) and ? to see what the command does and how it may be helpful in using TENEX.

File Operations:

```
acceSS (TO FILES) filename (BY) access path (IS) access
apPEND (FILE LIST) filename (TO) filename
arcHIVE second keyword (FILE LIST) filename *
copY (FILE LIST) filename (TO) filename
delETE filename
diRECTORY <directoryname>filename *
expUNGE (DELETED FILES)
intERROGATE filename or ESC *
list (FILE) filename *
proTECTION (OF FILE) filename (IS) octal#
qfd filename
renAME (EXISTING FILE) filename (TO BE) filename
tcOPY (FILES) filename
tYPE (FILE) filename *
undELETE (FILES) filename
perpETUAL (FILES) filename
not perpETUAL (FILES) filename
```

Editing:

```
(↑A)  CTRL-A  (character delete)
(↑C)  CTRL-C  (abort and return to Exec)
(↑F)  CTRL-F  (field recognition)
(↑R)  CTRL-R  (retype edited command)
(↑W)  CTRL-W  (delete current word)
(↑X)  CTRL-X  (delete current line)
```

Formatting:

```
formFEED
indICATE (FORMFEED)
lowERCASE **
no formFEED **
no lowerCASE
no raISE
no tabs **
raISE **
stoPS N, N, ..., N
wIDTH (OF LINE IS)
```

Linking, Advising:

```
brEAK (LINKS)
link (TO) tty#  or  username
recEIVE (LINKS)
refUSE (LINKS)
whERE (IS USER) username
```

Status:

```
daYTIME
discUSE
dskslAT
jobSTAT
netlOAD
runslAT
sySTAT
trmSTAT
useSTAT
verSION
who
```

Miscellaneous:

```
attACH  (USER) username (PASSWORD) password (JOB #) #
chANGE paSSWORD (OF DIRECTORY) name (FROM PASSWORD) old password (PASSWORD)
                                    new password (PASSWORD) new password
connECT (TO DIRECTORY) directoryname (PASSWORD) PASSWORD
conTINUE
detACH
exec (or push)
logIN (USER) username (PASSWORD) password (ACCOUNT) account
logoUT
maIL chECK (FOR USER) username
numBER (OF DIRECTORY) directoryname
quit (or pop)
run (FILE) filename
teRMINAL (TYPE IS) kind
```

5. TOPS-20

The TOPS-20 timesharing operating system is based on the TENEX operating system with some features from the TOPS-10 operating system and a few entirely new features. A time-sharing operating system is one that allows many people to use a computer at the same time: to create and run programs or share other users' programs or use standard programs (subsystems) provided with the system. While being able to share programs and data, the user is also able to keep information confidential.

Information about the programs and subsystems supported by TOPS-20 can be found in directory <DOCUMENTATION>. <DOCUMENTATION> files are catalogued according to type (e.g., .manual, .info, .changes) and files with a .changes type are generally in message file format.

There is a HELP program which gets help information for some programs. Type HELP and the name of the program or subject, e. g., type

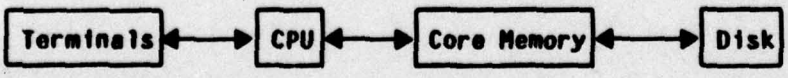
```
@HELP MANUALS(RET)
```

for a list of and information about TOPS-20 manuals available from The Digital Equipment Corporation (DEC).

5.1 OPERATING SYSTEM

The TOPS-20 *operating system* (also called a *monitor* or a *supervisor*) supervises everything the user runs on the computer. The computer can run one program for each user at a time; the first program run when the machine is started up is the operating system program, or simply the *system*. The system manages placement of files on the disk, where programs are in core, reading and writing characters on terminals and the line printer.

In its simplest form the *system* looks like this:



and the basic definitions are:

OPERATING SYSTEM

Usually refers to the whole picture above: the terminals, central processing unit, core memory, and disk.

CPU (CENTRAL PROCESSING UNIT)

The CPU follows the instructions (programs) given to it by the user -- it *runs the program.*

CORE MEMORY

Core Memory is where programs are held while the CPU *executes* them. *Data* from the terminals (characters) and from the disk (files) is stored *in core* while it is being worked on. Core memory is made of *bits*; 36 bits make a *word*; 512 words make a *page.*

DISK

Programs and data are stored as *files on disk. Program files* may be brought into core to be run. *Data files* may be created and changed (by text editor programs, for example); they may be *copied* to the terminal, to the *line printer*, or duplicated on disk.

The disk is where your work is kept when you are not working on it. Each user may have many files on the disk, and the names of the files are kept in a *directory* named after the user or the user's group.

Each *directory* has two disk allocations, both of which are enforced. One is a *working storage*, which is the maximum amount of disk you can use while being logged-in. The second is *permanent storage*, which is the maximum disk space the user may have at logout time. As the working storage limit is exceeded, deleted files will be expunged automatically to allow for more room, and the user is notified of each file which is expunged.

5.2 COMMAND LANGUAGE

The TOPS-20 Command Language allows the user to access and control the TOPS-20 facilities. The language consists of: 1) system commands that are entered from keyboard terminals, typing ? at the system prompt (@) prints a list of TOPS-20 commands, and; 2) the

name of a *subsystem*. A subsystem is a program developed to enable users to perform tasks on a computer. The subsystem programs are kept in a special directory of files on the disk, any user may start them by just typing their names.

Login messages (messages automatically typed on the terminal when a user logs in) from the <SYSTEM> directory notify the user whenever changes are made to existing subsystems or programs. Further, when the system is intialized (a user logs in), it saves the date and time of write of the file <DOCUMENTATION>SYSTEM.CHANGES, which will be compared against the date and time of write for programs run out of <SUBSYS>, <NEWSYS>, or <OLDSYS>. If the program is newer, the message "[Modified program]" is typed. The appropriate .CHANGES file in <DOCUMENTATION> should explain the differences.

Communication with the system takes the form of a dialogue: the user types a command, the system does what it is told, including filling in parts of the command it knows about, then comes back for more input from the user.

When the system is ready to accept a command it types an *at sign* (@). Commands are *terminated with a carriage return (RET).*

5.2.1 Special Characters

In addition to system commands and subsystems, TOPS-20 supports a set of special characters -- namely, the ESCAPE (ESC) (or ALTMODE) and DELETE (DEL) (or RUBOUT) terminal keys, and control characters.

The ESCAPE Key

> The ESCAPE (or ALTMODE) terminal key, located at the upper left side of the keyboard, is labelled ESC (or ALT) on most terminals. It does two things: 1) alerts TOPS-20 that a user wishes to communicate with the system, and 2) instructs the system to try to recognize what has been typed and to type out the rest. Thus, the user may type part of a command or file name and ask the system to supply the rest by typing ESC. If the partial type-in is not unique enough for recognition, then the system will output a bell (the terminal will ding) telling the user to type in a few more characters and another ESC. ESC is shown in examples and scenarios as $.

The DEL Key

> If a typing error is made while inputting a system command or a filename,

typing the DEL (DELETE or RUBOUT) key will cause the system to "forget" the last character typed. If the DEL key is typed, the system will print the bad character followed by a backslash (\), e.g., @lotT\gin. On some display terminals the system removes the bad character from the display.

CONTROL CHARACTERS

Control characters signal the system to do something immediately and are shown on the terminal preceded by an up arrow (↑), e.g., ↑C. They are typed by holding down the control key (on the left side of the keyboard) and then typing the character. Thus, the control key is like a shift key except that there is no "control lock" key.

The most important control characters are the following:

CTRL/C (↑C) - CALL TOPS-20 COMMAND LEVEL

Typing CTRL/C interrupts whatever is happening and returns control to the system command level. CTRL/C is used to abort a command, or to interrupt a running program.

CTRL/F (↑F) - FILE NAME FIELD RECOGNITION

Typing CTRL/F causes recognition of a partially typed field of a file name and is useful when one or more of the fields has a long or complicated name. Both ESC and CTRL/F may be used in combination to perform recognition on the parts of a file name. In the following example, CTRL/F is used for recognition of the directory name and ESC is used for recognition on the file name. What the user must type is shown is lower case, system output is shown in UPPER case; ESC is shown as $, typing the return is shown as (RET).

EXAMPLE:

```
@list <doc↑FUMENTATION>xed.m$ANUAL.1(RET)
```

Since CTRL/F, ESC and (RET) are *not* printed on the terminal, what the user actually sees is:

```
@list <docUMENTATION>xed.mANUAL.1
```

CTRL/H (↑H) - REPRINT PART OF COMMAND LINE

If a command line containing an error has been terminated (the RET key typed), the system informs the user of the error by printing a ?, sometimes followed by an error message. Typing CTRL/H (which will echo on the

terminal) causes the system to type the command line up to the error. The user may then complete the command correctly.

CTRL/W (↑W) - DELETE SINGLE FIELD

Typing CTRL/W is exactly like typing DEL, except that instead of deleting the last character typed by the user, it deletes the last word (or field) the user typed. Typing CTRL/W causes the system to print a left arrow (←), indicating that the word (or field) has been deleted. The user is then allowed to continue typing the command. On some terminals the left arrow character is printed as a single underscore; on some display terminals the system removes the word (or field) from the display instead of printing the underline or backarrow.

CTRL/U (↑U) - ERASE AND CANCEL CURRENT COMMAND

Typing CTRL/U at any point in a command will cancel the entire command and prevent the system from acting on it. The system prints XXX before printing the prompt character (@). On some display terminals the system simply removes the line instead of printing XXX.

CTRL/T (↑T) - INTERRUPT FOR STATUS

To find out if a program is running, type a CTRL/T; the system prints a status message telling the user the state of the program, the amount of system time used in the amount of time the user has been logged in (or connected to the system). Typing CTRL/T does not affect the program; it merely prints information about it.

EXAMPLE

```
RUNNING at 4263 USED 0:01:46.2 in 3:39:30
(status)              (CPU time)   (connect time)
```

CTRL/O (↑O) - SUPPRESS OUTPUT

Typing CTRL/O suppresses output to a terminal. The system prints ↑O... and stops all output to the terminal. The program (or command) still executes, but no output appears on the terminal. When the program (or command) finishes, the system prints the system prompt (@). Typing a second CTRL/O causes the output on the terminal to continue printing at the point where the program is currently executing.

To control information from moving off the screen of display terminals, set the PAGE argument of the TERMINAL command.

CTRL/S (↑S) and CTRL/Q (↑Q) - STARTING and STOPPING TERMINAL OUTPUT

Setting the TERMINAL PAGE mode allows a user to stop and start output to the terminal without interrupting the program (or command) producing the output. With TERMINAL PAGE mode set, typing CTRL/S causes the system to stop the output to the terminal, and typing CTRL/Q causes the system to resume typing the output to the terminal. If the output being typed is longer than the current page length for the user's terminal, the system automatically stops the printout and rings the terminal bell. Typing CTRL/Q continues printing the output.

Use the INFORMATION (ABOUT) TERMINAL-MODES command to see the current values of your terminal parameters (or settings). If useful, set the page length by including it in the TERMINAL PAGE command, e.g., TERMINAL PAGE 20 (sets the terminal page length at 20 lines).

5.3 FILE SYSTEM

The TOPS-20 file system provides 1) access to and control over named files which contain data or text or a program, and 2) a means for storing and retrieving information. In order to bring an system command to bear on a file, the user must identify the file in question by giving its *file specification (or file name)*. The parts (or fields) of a file name which are important immediately are:

<DIRECTORY>NAME.TYPE.GENERATION

The *directory* is the general place under which all of a user's files are listed. The directory name in a file specification must be typed enclosed in angle brackets (<>); the other parts of the file name must be separated by a period (.). A directory name is usually the last name of the user or the name of the user's group or project. If the directory name is not included as part of the file name, the system will assume the directory name of the logged in user.

To access files in other directories the user *must* name the directory as part of the file name. In file name specifications, directory names begin and end with opening and closing angle brackets. The system will perform recognition on a directory name if a partially typed name is terminated by ESC or CTRL/F.

The *name* specifies a particular file in the directory. Names may be any combination of up to 39 letter and digits. The name is frequently all that need be typed to refer to a file. As with directory names, the system will recognize file names when they are terminated with ESC or CTRL/F. The user has total control over file names in his directory and can construct file names that require little input for recognition.

The *type* specifies the contents of the file and is a way of having several related files with the same *name*. The *type* part of a file name consists of a period, followed by any combination of up to 39 letters and digits. Although the system regards all files as having *type* as part of the file name, a user can specify a null type (one having no characters) by typing just the initial period.

The *generation* number specifies how many times a file has been changed. Generation numbers allow the user to create multiple disk files with the same combination of file name and type. The intent of generation numbers is to keep track of successive, modified versions of a single file. The system will usually assume the highest generation number if it is left off (with certain desirable exceptions noted in the Basic Commands section).

Examples of File Specifications (File names):

```
MAIL.TXT.1
REPORT1..1,2
REPORT1.XGO.1
<DOCUMENTATION>XED.MANUAL.7
<DOCUMENTATION>XED.CHANGES.1
<DOCUMENTATION>COMMAND-SUMMARY.INFO.1
<HOLG>LOGIN.CMD.7

LPT:      (special designator meaning the line printer)
TTY:      (special designator meaning the terminal)
XGP:      (special designator meaning the Xerox Graphics Printer)
```

The structure of a TOPS-20 file name reflects the convenience and uniformity of the file system, most of which would be lost if the user had to enter all of the file name each time he accessed a file.

Certain system commands can address several files at one time. For example, one can APPEND a series of files onto an existing file, or DELETE several files in one operation.

5.3.1 Group File Access

TOPS-20 provides a mechanism whereby users can form groups for the purpose of file sharing. This is useful, for example, where several users are collaborating on a common task or project and wish to share the files they are creating. Each group is assigned a number. The system records a series of numbers for each user in a group, indicating what groups he belongs to; for each file directory the system records numbers indicating which groups have access to the files it contains. Thus a given user can belong to a number of

groups. He/she can make his directory available to one or more groups. The assignment of group numbers to users and to file directories is done by the TOPS-20 system operators.

5.4 SYSTEM COMMANDS

TOPS-20 system commands are made up of a combination of the following parts:

A *command name* identifies the command and describes its function.

guide words describe the kind of argument that a user should type.

An *argument* is the information acted upon by the command.

switches and *subcommands* modify or specialize the function of the command in which they appear.

A *command terminator* is a carriage return/linefeed sequence (typing the *Return*, *CR*, or *CAR RET* key) or a linefeed (typing the *LF*, or *LINEFEED* key).

One-line commands are terminated by typing the (RET) key; multi-line commands (those using subcommands) are terminated by typing the (RET) key twice.

Example of one-line command terminator:

```
@dirSECTORY (OF FILES)(RET)

   <HOLG>
 MAIL.TXT.1
 LOGIN.CMD.7
@
```

When subcommands are used, terminate the command line by typing a comma and RET. The system prints @@, indicating that it is ready to accept subcommands (typing ? prints a list of the subcommands). Each subcommand is terminated by typing (RET); the last subcommand is terminated by typing the (RET) key twice.

Example of multi-line command terminator:

```
@dir$ECTORY (OF FILES),(RET)
@@del$ETED (FILES ONLY)(RET)
@@(RET)

  <HOLG>
 LOGIN.CMD.6

 TOTAL OF 1 FILE
@
```

Each part of a command or subcommand (subcommands contain names, guide words, and arguments of their own) is known as a *field* and is separated from each adjacent field by a space.

Example of the APPEND command fields:

```
    @APPEND      (FILE LIST)   CHAP.2.1      (TO)        CHAP.1.1
(command name) (guide word) (argument) (guide word) (argument)
```

You already know that typing ? after the command prompt (@) prints a list of the TOPS-20 system commands. When unsure of a command or subcommand argument, type ? to have the system print information about what it expects you to type.

5.4.1 Typing Commands

Commands can be typed using three styles of input:

1) *Recognition Input* – type a portion of the command and press the ESC key.

2) *Abbreviated Input* – type the unique portion of the command, arguments, and subcommands and separate the fields with a space.

3) *Full Input* – type complete command names, arguments and subcommands, using a space to separate the fields.

The three styles of command input can be intermixed freely within a command, with a separate style of input applied, if desired, to each field. Further, commands may be typed in either UPPER or lower case, or a combination of UPPER/lower case.

In the following examples of recognition and abbreviated input, what the user types is shown in lowercase, what the system prints is shown in UPPER case or enclosed in square brackets ([]). ESC is shown as $; the command terminator (typing the return key) is shown as (RET).

Example of recognition input:

```
@ap$END (FILE LIST) part2$..2 (TO) part1$..1 [Old Generation](RET)
```

Example of abbreviated command input:

```
@ap part2  part1(RET) [Old Generation](RET)
```

When typing commands, if recognition (typing ESC or CTRL/F) input is used in a place where its use is ambiguous, the system rings the terminal bell. Type more information, or type ? to find out what the system expects you to type. With certain frequently used commands, fewer than the minimum characters are accepted by convention in either abbreviated or recognition input. An example is LOG, which is taken to mean LOGIN even though LOGOUT also begins with LOG.

5.4.2 Basic Commands

The commands given below form a basic set that will allow the new user to get started. A list of *all* of the commands supported by TOPS-20 is available by typing ? at the TOPS-20 command level (@). Use ESC and ? freely to learn these and other commands.

We will show commands in both recognition and abbreviated form. User input is shown in lower case; system recognition output in UPPERCASE or enclosed in square brackets ([]); ESC is shown as $ and the command terminator typed by the user is shown as (RET). Remember, typing ESC causes the system to recognize the rest of the command and to prompt the user for the next thing to be typed.

```
@log$IN (USER) username$ (PASSWORD) password$ (ACCOUNT) number(RET)
or
@log username password account(RET)
```

> A valid *user name*, a *password*, and *account name or number* are necessary to use the LOGIN command.
>
> *User name* is your name or your group name (if you are sharing a directory) and may consist of up to 39 alphanumeric characters, including hyphen.

Password is the password associated with your user name; it may consist of up to 39 letters and digits, including hyphen. *Passwords should be kept secret* and known only to those who are using the directory. Passwords may be changed using the SET PASSWORD command. Passwords typed during the LOGIN command are not printed on the terminal.

Account is your account name or number. Typing ESC as your account causes the system to automatically print it.

After you make a connection to your host computer the system will print a greeting line and the system prompt (@). You may then use the LOGIN command.

The LOGIN command validates you as a user, creates your job, and begins charging your account. After creating your job, the system executes the commands stored in your LOGIN.CMD file, if it exists in your directory.

EXAMPLE:

```
ISI-SYSTEM-E, TOPS-20 Monitor 101B(1221)-1
System shutdown scheduled for 27-Jul-78 22:00:00,
Up again at 28-Jul-78 05:00:00
@log smith  123456(RET)
JOB 9 on TTY9 27-JUL-78 09:09
End of LOGIN.CMD.7
[YOU HAVE NEW MAIL]
@
```

LOGIN.CMD File

A user may create a LOGIN.CMD file that is read by the system every time he logs in. Once a successful LOGIN command is completed, the system executes any TOPS-20 commands stored in a LOGIN.CMD file kept in the user's logged-in directory. After executing these commands, the system prints any output from the commands followed by the message END OF LOGIN.CMD and the system prompt (@). A LOGIN.CMD file is not required; the user should create the file only if it's convenient.

LOGIN.CMD files are created using a text editor (XED, TECO, etc.). You may enter any system commands in the LOGIN.CMD file; the system executes them each time you log in. The most common commands in a LOGIN.CMD file are TERMINAL commands (to declare the type of terminal you have and set appropriate terminal modes); e.g., TERMINAL HP, TERMINAL PAGE, TERMINAL LOWERCASE, etc.

If there is an error in one of the commands, the system processes all the commands up to the one in error, prints an error message telling you it found an error and aborted the file.

Make sure you can give the commands exactly as you have entered them into the file. If not, correct any errors and try the file again.

@logoSUT(RET)
or
@logo(RET)

The LOGOUT command expunges deleted files in your directory, deletes all temporary files (;T) created during the session, checks disk storage allocation, ends your job and prints a logout message. You may also log out another job as long as it is logged in under your user name.

EXAMPLE:

```
@LOGO (RET)
 Killed Job 99, User HOLG, Account UDR, TTY 63, at 27-Jul-78 09:38:24
 Used 0:0:5 in 0:0:19
```

or, if logging out another job, type the LOGOUT command and job number

```
@logo 72 (RET)
@
```

You cannot log out another user's job; you cannot give a job number when you are logging out your own job.

If you exceed your permanent disk storage allocation, the system prints a message at time of LOGOUT telling you the number of pages by which you are over permanent storage allocation. Before you log out, you can find out if you are over your allocation by using the INFORMATION (ABOUT) DISK-USAGE, or I D(RET). Since permanent storage disk allocation is strictly enforced, a good standard practice is to check your status before you log out. If you are running low on space you can delete unnecessary files at that time.

After logout the terminal will not respond until you make a connection to your host computer.

@sy$STAT(RET)
or
@sy(RET)

Prints the header and the job number, line number, current program name and user name for each job.

@sy username(RET)

Prints information about only the specified user.

```
@sy job number(RET)
@sy .(RET)
```

Sy job number(RET) prints information about only the specified job. Sy .(RET) prints information about your current job.

```
@sy all(RET)
```

Prints the header and job number, line number, current program name, state of the job, the accumulated run time, the run time limit, the user name, and connected directory for each job.

The SYSTAT command prints information about the current state of the system. The SYSTAT command does not change any program in memory and leaves your terminal at TOPS-20 command level.

```
@dir$RECTORY (OF FILES)(RET)
or
@dir(RET)
```

Prints the file name, type, and generation number for each file in the user's directory.

```
@dir filename(RET)
```

Prints the file name, type, and generation numbers for the file specified in the command.

```
@dir filename,(RET)
```

Typing comma (,) allows use of the subcommands associated with the DIRECTORY command. The system prints @@ (indicating subcommand mode); typing ? RET prints a list of the subcommands. Each subcommand must be terminated by RET; the last subcommand must be terminated by two RET's.

```
@dir *.type(RET)
```

Prints all of the file names with the type specified in the command.

```
@dir name.$(RET)
```

Prints all of the files with the name specified in the command.

The DIRECTORY command prints information about a file or group of files. The system lists the files in alphabetical order, unless the REVERSE or CHRONOLOGICAL subcommands have been given. If two files have the same file name, only the type and generation number for each succeeding file after the first are printed. The period before the type is indented three spaces. If a file is temporary, a ;T is printed after the generation number.

The DIRECTORY command does not change any program in memory and leaves the terminal at TOPS-20 command level.

```
@fdirSECTORY(RET)
or
@fdir(RET)
@fdir filename(s)(RET)
```

FDIRECTORY (or Full DIRECTORY) automatically prints everything about the files (or file names specified in the command), without any headings, and compresses the output. It is the same as giving the DIRECTORY command with the CRAM (output), EVERYTHING, and NO HEADING subcommands.

```
@tdirSECTORY(RET)
or
@tdir(RET)
@tdir filename(s)(RET)
```

TDIRECTORY (or Time-ordered DIRECTORY) orders the files (or files specified in the command) with the most recent write dates first and prints the write times and dates. It is the same as giving the DIRECTORY command with the CHRONOLOGICAL (BY) WRITE, REVERSE (SORTING) and TIMES (AND DATES OF) WRITE subcommands.

```
@vdirSECTORY(RET)
or
@vdir(RET)
@vdir filename(s)(RET)
```

VDIRECTORY (for Verbose DIRECTORY) automatically prints the protection, size, length, and write time and date for each file (or filename(s) specified in the command). No headings are printed. It is the same as giving the DIRECTORY command with the PROTECTION, SIZE, LENGTH (IN BYTES), TIMES (AND DATES OF) WRITE, and NO HEADING subcommands.

By giving the proper file specification(s), you can obtain information about a single file, a group of files, or files in another directory.

```
@del$ETE (FILES) filename(s)(RET)
or
@del filename(s)(RET)
```

The DELETE command identifies a disk file(s) for eventual deletion. If a generation number is not specified, the system assumes the oldest number. Once you delete a file, it is kept on disk storage until either you (or the operator) give an EXPUNGE command, or until you log off the system. If you want to see what files you have deleted, but not expunged, use the DIRECTORY command with the DELETED subcommand. Once you delete a file, you cannot access the file with a normal TOPS-20 command or program, except with an UNDELETE command, a DIRECTORY command (with the DELETED subcommand), or an EXPUNGE command.

As the system starts to delete a file, it prints the file name and leaves a space. When the file is deleted, the system prints the success message [OK].

The DELETE command does not change any program in memory and leaves the terminal at TOPS-20 command level.

If a file *disappears* from the user's directory, chances are that the file had not been read or written for approximately a month and has been *ARCHIVED*. The user should use the *INTERROGATE* command (see FILE ARCHIVAL/RETRIEVAL SYSTEM section).

```
@und$ELETE (FILES) filename(s)(RET)
or
@und filename(s)(RET)
@und name.*(RET)
```

The UNDELETE command restores previously DELETEd disk files to their normal status. If a generation number is *not* specified in the command, the oldest number is assumed. If any one of the files you are trying to undelete does not exist, the system ignores the entire command and prints an error message (e.g., ?File not found). The UNDELETE command prints no error message when it cannot restore a set of files that have been specified with the wildcard (asterisk as part of the file name) construction. Files that have been EXPUNGEd *cannot* be undeleted. If your attempt to UNDELETE a file fails, send a message to ACTION asking that the file be restored from system backup tapes.

If MAIL.TXT is deleted, its contents are *immediately expunged*. Using the UNDELETE command to restore MAIL.TXT will restore the filename but none of its contents.

The UNDELETE command does not change a program in memory and leaves the terminal at TOPS-20 command level.

@type$ (FILE) filename(s)(RET)

The TYPE command prints the contents of a single file name or a string of file names (separated by commas) on your terminal. If you want to stop the printing of a file, type a CTRL/O or two CTRL/Cs. If your file has line numbers, the system prints them on your terminal. If you type more than one file with a single TYPE command, the system prints the proper file name followed by a blank line, before it prints the file.

The TYPE command does not change any program in memory and leaves the terminal at TOPS-20 command level.

@print (FILES)/SWITCHES filename(s)/switches(RET)

The PRINT command places entries into the lineprinter output queue, which is a list of jobs waiting to be printed. To look at the contents of the entire queue, use the INFORMATION (ABOUT) OUTPUT-REQUESTS command. If you want only the entries for your job(s), include the /USER switch. To just print a file, no switches are necessary.

A *switch* modifies or specializes the function of the command in which it appears. If you want a file switch to apply to one file only, place the switch after the particular file name; if you want the file switch to apply to all files in the list, place the switch before the list of file names. Switches requiring a date and/or time argument: separate the hours from minutes by typing a colon (:), and the month, day, year by typing a hyphen (-), e.g., 9:30, Jul-4-77.

Some switches are:

/AFTER: date and/or time - submits the file for printing after the specified date or time.

/LOWERCASE - directs the printing job to a lineprinter that uses both upper and lower case characters.

/UPPERCASE - directs the printing job to a lineprinter that uses only uppercase characters.

/COPIES:n - specifies that n copies of the file be printed. n (the number of copies) must be less than 512. Normally, the PRINT command makes one copy of each file.

/DELETE - immediately removes the file from your directory and deletes it after it has been printed.

/SPACING:single - specifies no extra space between lines on a printout.

/SPACING:double - specifies a single line of spacing between lines on a printout.

/SPACING:triple - specifies a double line of spacing between lines on a printout.

Typing /? prints a list of all PRINT command switches.

The PRINT command does not affect memory; it leaves the terminal at TOPS-20 command level.

```
@INFORMATION (ABOUT) parameter
or
@I parameter
```

The INFORMATION command prints information about various system and job parameters. Type ? for a list of system and job parameters available with this command. Type (or use recognition on) the parameter name. If the parameter has a modifier, type (or use recognition on) the modifier, press the RET key and the system prints the information. You may default the modifier by simply omitting it.

EXAMPLES

```
@INFORMATION (ABOUT) ? One of the following:
 AVAILABLE
 CARD-READER-INPUT-SET
 COMMAND-LEVEL
 DISK-USAGE
 FILE-STATUS
 JOB-STATUS
 LOGICAL-NAMES
 MAIL
 MEMORY-USAGE
 MONITOR-STATISTICS
 NETWORK-STATUS
 PROGRAM-STATUS
```

```
 PSI-STATUS
 SPOOLED-OUTPUT-ACTION
 SUBSYSTEM-STATISTICS
 SYSTEM-STATUS
 TAPE-PARAMETERS
 TERMINAL-MODE
 VERSION


@1 j
 Job 47, User HOLG, Account UDR, TTY162
@

@set$ arguments(RET)
or
@set ? One of the following:
 ACCOUNT
 CARD-READER-INPUT-SET
 CONTROL-C-CAPABILITY
 ENTRY-VECTOR
 ERROR-RETRY
 FILE
 LATE-CLEAR-TYPEAHEAD
 MAIL-WATCH
 NO
 PASSWORD
 SPOOLED-OUTPUT
 TAPE
 TIME-LIMIT
 UUO-SIMULATION
```

The SET command sets the value of various job parameters. The INFORMATION command prints the values of many of the parameters you can set with the SET command. The TERMINAL command sets various terminal parameters which you can examine with the INFORMATION (ABOUT) TERMINAL-MODE command.

Type SET and leave a space; type (or use recognition on) the rest of the arguments to the SET command, then press the RET key.

The SET command does not change a program in memory and leaves your terminal at TOPS-20 command level.

```
@terMINAL (MODE IS) ? One of the following:
 33
 35
 37
 DM
 EXECUPORT
 FLAG
 FORMFEED
 FULLDUPLEX
 HALFDUPLEX
 HELP
 HP
 IMMEDIATE
 INDICATE
 LA30
 LA36
 LENGTH
 LINE-HALFDUPLEX
 LOWERCASE
 NO
 PAGE
 RAISE
 SPEED
 TABS
 TERMINET
 TI
 TYPE
 VT05
 VT50
 VT52
 WIDTH

@terMINAL (MODE IS) hp
@
```

The TERMINAL command changes various terminal parameters such as: input/output speed, handling of lowercase characters, length and width of the terminal page, and handling of the RUBOUT key.

The TERMINAL command helps make typing on the terminal more convenient. You can print the current values of your terminal parameters by giving the INFORMATION (ABOUT) TERMINAL-MODE command. If you use TERMINAL commands often, place the command in your LOGIN.CMD file.

The TERMINAL command does not change any program in memory and leaves the terminal at TOPS-20 command level.

```
@subsystem name(RET)
or
@msg(RET)
@xed(RET)
@hermes(RET)
```

The user 'calls' (or starts) the various subsystems like XED, MSG or HERMES by typing the subsystem name followed by RET. Each subsystem has its own command level and commands. When a user is running a subsystem he is out of TOPS-20 command level and must use the subsystem commands. Typing CTRL/C at any point during use of a subsystem causes an interrupt and the user is returned to TOPS-20 command level. If a CTRL/C was typed accidentally, the user may use the *CONTINUE* command to put him back to where he was in the subsystem before the ↑C was typed.

```
@cont$INUE(RET)
```

The CONTINUE command restores execution of a program that was interrupted by CTRL/C. The program is not changed by the CTRL/C-CONTINUE process as long as you do not change the contents of memory (e.g., load another program) or the files it manipulates.

You may stop a program by typing one or more CTRL/Cs, give a PUSH command (to create a new command level), run another program, give a POP command (POP terminates the current command level and returns you to the next higher level), then give a CONTINUE command to continue the initial program.

The CONTINUE command continues the program in memory and leaves your terminal under its control.

Some non-standard programs may not allow you to CONTINUE them once they are interrupted.

```
@talk$ (TO) user name or a line number
```

The TALK command creates a link between the user's terminal and the terminal of the user or line number specified in the command. Whatever you type is printed on both your terminal and the linked terminal; whatever a user types on the linked terminal appears both on his terminal and your terminal. The linkage affects only the output of the terminals; the input of the terminals is left intact, e.g., if your program is waiting for a command and a user types a

command on the linked terminal, that command works only for his job, not yours – you see only the resulting typescript.

To avoit getting the ?UNRECOGNIZED COMMAND error message, precede your comments with an !.

The BREAK command breaks any links you have established with the TALK command.

You may not link to a detached job. The TALK command does not print a password onto any linked terminals. If, however, you type a CTRL/R while giving a command containing a password, the password is printed.

The TALK command does not change any program in memory and leaves your terminal at TOPS-20 command level.

```
@bre$AK (LINKS)(RET)
or
@bre(RET)
```

The BREAK command clears *all links* that have been made to or from your terminal; the TALK command creates *only one* link. BREAK does not require you to be logged into the system, gives no error messages if you do not have any links to or from your terminal, does not change a program in memory, and leaves your terminal at TOPS-20 command level.

5.5 FILE ARCHIVAL/RETRIEVAL SYSTEM

The TOPS-20 EXEC supports a file archival facility. The purpose of the archival facility is twofold:

1) It provides each user with a large, long-term storage capability.

2) It provides the system with a mechanism for freeing online storage.

User files are allowed to "age" in the user's directory for approximately 30 days. Files so "aged", neither read nor written for a month, will be copied several times to the magtape archive, verified, and deleted from the user's directory. The location of the file is recorded (see INTERROGATE below) so that the user will be able to notify the operator of a retrieval request.

The following commands are available in the EXEC to support user use of the archive.

5.5.1 THE ARCHIVE COMMAND

The ARCHIVE command provides the user with a method for setting archive requests, resetting archive requests, and determining if a file is in the process of being archived.

```
@ARCHIVE <second keyword> (FILE LIST) <file list>

<second keyword> is one of the following:

DELETE
EXPUNGE
FILE
RESET
STATUS
UNDELETE

DELETE, UNDELETE & EXPUNGE give users control over the names in
their archived files directory file ]ARCHIVE-DIRECTORY[.;1.

FILE will request archival of the file(s) in the <file list>

RESET will clear archival information in the <file list>

STATUS shows the archival status for the <file list>
```

By typing a comma after <file list>, one can give subcommands to the ARCHIVE command. Current subcommands are:

```
@@DEFERRED
```

This tells the system to archive <file list> at the next time the archive system is run (the default).

```
@@DELETE
```

This tells the system to delete <file list> from disk AFTER they have been archived and verified (the default).

```
@@DON'T ARCHIVE
```

This tells the system to never archive the specified file.

```
@@DON'T DELETE
```

This tells the system to not delete the file from the directory after it has been successfully copied to the archive.

As with other EXEC subcommand modes, two carriage returns confirm the execution of the ARCHIVE command whenever subcommands have been specified.

Example of ARCHIVE command use:

```
@arcHIVE ? ONE OF THE FOLLOWING:
 DELETE
 EXPUNGE
 FILE
 RESET
 UNDELETE

 arcHIVE fILE (FILE LIST) graph.apr-78;1,
@@? ONE OF THE FOLLOWING:
 DEFERRED
 DELETE
 DON'T
 IMMEDIATE

@@don't dELETE
@@
@
@arcHIVE sTATUS (OF FILES) graph.apr-78;*
 GRAPH.APR-78;1  ARCHIVE WITHOUT DELETION PENDING
@
```

5.5.2 THE INTERROGATE COMMAND

The INTERROGATE command is used on a fully archived file. Typing

```
@INTERROGATE  <filename>(RET)
```

causes the archive to be interrogated. If the specified file is not found, THAT FILE IS

NOT ARCHIVED is typed out; if the file is found, ARCHIVED ON TAPES MMM AND NNN is typed out; if found, DO YOU WANT IT RETRIEVED? is also typed out; if the answer is yes, a message will be printed on a special teletype in the computer room; the operator will periodically check for such requests; when the file has been retrieved, the retrieval will be confirmed via SNDMSG.

As with the ARCHIVE command, typing a comma after the file name (or list) gives the user access to the INTERROGATE subcommands. Two carriage returns confirm the execution of the INTERROGATE command whenever subcommands have been specified. INTERROGATE subcommands are shown in the following example.

```
@interROGATE graph.mar-78.;1
 <SIGNAL>GRAPH.MAR-78.;1  archived on tapes 3246 and 3241
 do you want it retrieved? (Y or N)  yes
@
```

or

```
@interROGATE graph.mar-78.;1,
@@?
  ALL.FILES
  BEFORE.DATE
  DATES
  DELETED.FILES.ONLY
  DOUBLE.SPACE
  EVERYTHING
  INDIVIDUAL
  LPT
  NO.HEADING
  NUMBERS
  OUTPUT.TO.FILE
  PROTECTION
  REVERSE.ORDER
  SINCE.DATE
  STATUS
  TAPE
  TIMES.AND.DATES
  UNDELETED.FILES.ONLY
  VERBOSE

  CARRIAGE RETURN TO GO
@@vERBOSE
@@
```

```
<SIGNAL>GRAPH.MAR-78;1;P775200
  3246   1-APR-78      3241   1-APR-78
do you want it retrieved? (Y or N)  yes
@
```

5.6 Basic TENEX/TOPS-20 Command Language Differences

FUNCTION	TENEX	TOPS20
Delete character	↑A	DEL key
Delete line	DEL key	↑U
Delete field	↑W	↑W
Reprint line	↑R	↑R
Freeze typeout	See Note 1	↑S
Resume typeout	See Note 2	↑Q
File names	name.ext;version	name.typ.generation
Fast directory	QFD	FDIRECTORY
Status of system	WHO or SYSTAT	SYSTAT
Status of disk	DSKSTAT	INFO DISK
Status of files	FILSTAT	INFO FILE
Status of job	JOBSTAT	INFO JOB
Call inferior EXEC	EXEC	PUSH
Dismiss inferior EXEC	QUIT	POP
List file	LIST	LIST
Type file	TCOPY	TYPE
Change job account	CHANGE ACCOUNT	SET ACCOUNT
Assign file account	ACCOUNT	SET FILE ACCOUNT
Change password	CHANGE PASSWORD	SET PASSWORD
File protection	PROTECTION	SET FILE PROTECTION
Upper/lower case	NO RAISE	TER NO RAISE
Terminal linking	LINK	TALK
Comment	;	!
User number	NUMBER (or NUM)	TRANSL
Where is user	WHERE (is) username	SYSTAT username

Note 1: Any control character user selects with FREEZE command.

Note 2: Any character, e.g., space.

6. SNDMSG

SNDMSG is used to send messages to users throughout the ARPANET. It prompts for the addresses to which to send the message, then for the subject and text of the message. It automatically fills in the date and and sender. Finally it sends the message. If the message can't be delivered immediately, it is queued and automatically sent later by the background MAILER process.

SNDMSG has four fields: To:, Cc:, Subject:, and Message:. The following sections describe each of these fields and the form of the message that is actually sent.

6.1 THE ADDRESSES

Addresses are divided into "To" and "cc". SNDMSG prompts separately for each. When SNDMSG prompts for addresses, enter them separated by commas. Carriage return terminates the list and goes on to the next section unless it is preceded by a comma, in which case address input continues on the next line. There must be at least one "To" address, but there need not be any "cc". If you don't enter any "To" addresses, SNDMSG will skip to the subject and message fields, then will come back to the "To" field.

The following kinds of items may appear in the address list separated by commas. (Angle brackets (<>) are not typed -- they are just used in this description to enclose a type of item.)

<user name>

> The mail is addressed to that user at the default host. Normally the default host is the local host (i.e., the one on which you are running SNDMSG) but you can change it as desired.

@<host name or octal host number>

> Changes the default host to be the given host. The default remains in effect for the rest of the address list until explicitly changed. To set the default back to local, type just "@" (i.e., omit the host name).

<user name>@<host name or octal host number>

The mail is addressed to the given user at the given host. Putting the host name together with the user name makes it apply only to that user. That is, it does not change the default host, but overrides it just for the user. A blank host name means the local host.

<Group Name>:

All subsequent addresses will be considered part of the specified group (which can be any name you care to define). This group remains in effect until explicitly changed by another "<group name>:" item. Typing just ":" with no name in front sets the group to none, i.e., subsequent addresses are ungrouped. Group names do not affect the sending of the message (since they are not addresses), only what appears in the heading of the transmitted message.

Note: It is not necessary to separate the group name from the next address by a comma; the colon implies a comma.

***<File name>**

The message will be sent to the given file. Files are always local and must already exist. *Defaults for fields you omit are:*

```
directory:   logged in directory (not connected).
name:        SAVED
extension:   MESSAGES
version:     highest existing
```

File name addresses are omitted from the message heading.

6.1.1 Editing and Viewing Control Characters

There is no way to edit anything but the current item being typed (i.e., editing won't go past the preceding comma). The following control characters edit the current item or kill the whole address list.

	TENEX	TOPS-20
Delete character	↑A	DEL key
Delete current word	↑W	↑W
Delete entire address list	↑X	↑X

Control characters for viewing addresses:

Retypes current word	↑R	↑R
Retypes address list	↑S	↑D

6.1.2 Inserting a File

A file containing addresses may be prepared ahead of time and inserted into the SNDMSG address list. Addresses are entered on the file exactly as they would be typed to SNDMSG except that carriage returns from a file will not terminate the address list – thus, addresses on a file may appear on separate lines without commas if desired.

To insert the file, type ↑B (Control B) during SNDMSG address input. You will be asked for a file name and for confirmation of the file name. The file is read into the address list exactly as if you had typed it in manually. When SNDMSG types EOF (End-OF-File), you may continue entering additional addresses. Input is terminated by typing a carriage return.

Although files are often used to hold definitions of groups, i.e., they often start with a "<group name>:", files and groups are two distinct features. The purpose of a file is to permanently hold some list of addresses. The purpose of a group name is to prevent the component address from cluttering up the message heading. Although groups may be defined in files, they may also be typed directly into SNDMSG. Files inserted into SNDMSG need not define groups.

6.2 THE SUBJECT FIELD

When SNDMSG prompts for "Subject", type in up to one line, terminated by carriage return. You may type just carriage return to omit having a subject. If a subject is entered it appears in the heading of the message. The same control characters described in the

MESSAGE Field section apply to the SUBJECT field with the following additions and exceptions.

TENEX and TOPS-20

↑Z has no special meaning.
↑B automatically means insert file, never invoke TECO.
↑X deletes the whole subject and starts over.

6.3 THE MESSAGE FIELD

When SNDMSG prompts for "Message" type in the text of the message. When you are finished composing the message type ↑Z (Control Z) to terminate input and go on to the next stage.

Various control characters invoke features that aid in composing the message. They are:

	TENEX	TOPS-20
Delete previous character	↑A	DEL key
Delete current line	↑Q	↑U
Delete previous word	↑W	↑W
Retype the current line	↑R	↑R
Retype whole message	↑S	↑D

6.3.1 Inserting Pre-composed Text and Editing

When dealing with large messages you may wish to prepare your message ahead of time on a file, or you may find the local editing characters above insufficient. If you type ↑B (Control-B), SNDMSG asks whether you want to insert a file or invoke TECO (a general text editor).

If you answer "T", TECO is started up with your message in its buffer. When you exit TECO (with :EI) the edited message is passed back to SNDMSG and you continue composing the message. You are positioned at the end of the message, as if you had just typed it into SNDMSG.

If you answer "F", SNDMSG asks for the name of the file to insert. Type the file name (the usual editing characters and recognition apply). You will then be asked to confirm by typing carriage return. The file is then read in and appended to any text you have already entered and "EOF" (End-Of-File) is typed. You may then terminate input by typing ↑Z, or continue to add text to the Message field until your message is complete, then type ↑Z to terminate.

6.4 SENDING THE MESSAGE

When all information has been entered (addresses, message, etc.) SNDMSG asks whether to send the mail now or queue it for later delivery.

If you answer with just a carriage return the message is sent immediately. The queueing or sending begins after this command is given, i.e., when you terminate with carriage return. If you have addressed the message to a file using * in the address list, SNDMSG will try to deliver it even if queueing is in effect.

The message is queued for, or delivered to, each of the addresses. For local addresses (ones in which no host was specified) duplicates are eliminated.

SNDMSG types each address as it starts to process it, then the outcome. There are three possible outcomes:

1. "ok" -- the message has been successfully delivered to the address.

2. "queued" -- the message was not delivered, but has been queued for later delivery. If SNDMSG queued the message even though you asked that it be sent immediately, the reason it had to be queued is typed. The message is placed in the file [--UNSENT-MAIL--].address in your directory. MAILER will deliver it as soon as possible.

3. "can't" -- the message could not be delivered and SNDMSG did not queue it in the [--UNSENT-MAIL--].address file. The reason is typed out and the message is put in the /UNDELIVERABLE-MAIL/.address file in your directory. If the problem was that the name was misspelled, you have the option to correct the spelling and requeue the message without having to create it over again by using the MAILSTAT program.

7. MSG MESSAGE HANDLING PROGRAM

MSG is a program for reading, writing, and subsectioning files which have a "message file format". It is very simple and straightforward to use. Commands are initiated by typing one character, which causes the program to type out the rest of the command name and wait for input from you.

Before the commands are described, there are a few general statements about how MSG works and some conventions used in describing the commands that you should know about. The prompt characters letting you know that MSG is waiting for a command character to be typed are "<-". When MSG is started up (by typing MSG<return> to the EXEC) it will first try to read the mailbox file in your directory (named MESSAGE.TXT;1 in TENEX; MAIL..TXT.1 in TOPS-20). your directory. If this file does not exist MSG will say so. If you were not connected to your login directory, MSG will try to find a mailbox file there. If that also fails, it will say so and wait for a command to be typed. If you have a mailbox file, it will scan it and type out the header information (i.e. the date, from, and subject fields) for each message since the file was last read, preceded by a message number sequentially assigned by MSG. These message numbers are used in association with the various commands.

However, if you started MSG by typing MSG<space> to the EXEC, it will ask you for a file to be read. Typing an escape as the first character will cause the name of your mailbox file (MESSAGE.TXT;1 or MAIL.TXT.1) to be typed out, and confirmation requested from the user to ensure that that was what was intended. Once a file name has been specified and positively acknowledged, then the same information as described in the previous paragraph will be output to your terminal.

When reading a message file in MSG, either when starting up MSG or with the Read command described below, the file must be in the so-called "message file format". If MSG recognizes that the file does NOT conform to this format, you will be told so. However, you will be given the opportunity to keep everything that has been read so far, but NOT overwrite the 'bad' file. These two exceptional circumstances and some suggestions for getting around them are described at the end of this manual.

The following conventions and symbols are used in the command descriptions below. There are only five types of input MSG expects:

(1) a MSG command (or sub-command) character
(2) a message sequence specification
(3) a TENEX file name
(4) a confirmation character
(5) a local user name or remote site name

To abort output to the terminal type tO (control-O). If MSG does not understand your input, it will return to command input mode, or reprompt you. The following are symbols and their associated meanings used in the command descriptions:

<FILE-NAME>

Stands for any TENEX file descriptor, including TTY: or LPT:. If you are requested to input a file name, the appropriate TENEX confirm will be given (e.g., [Old version]).

<MSG-SEQUENCE>

This input is prompted by the string (message sequence) in verbose typeout mode. A sequence of message numbers has the following format.

(1) Any single message number.

(2) Any two numbers separated by ">" or ":". This means message numbers delimited by the two outside numbers (e.g., 2>5 means messages 2,3,4,and 5 in that order). NOTE: if the first number is greater than the second number, it means the sequence in reverse order (e.g., 5>2 means messages 5,4,3, and 2).

(3) A pair of numbers separated by "-". This is so that the standard interpretation of the string "21-4" (that is not "21-24") means message numbers 21, 22, 23, and 24. Using this interpretation, the string and "24-1" is an error.

(4) Any sequence of the previous three types separated by commas. This is the way to group several non-adjacent messages together. For example: 1,3,5:7,10 means messages 1 and 3 and 5 through 7 and 10.

<MSG-SEQUENCE> of the types described above are ALWAYS terminated by <return>.

(5) However, there are special types of message sequences. All are determined by the first character that you type in the <MSG-SEQUENCE> stream. The following are the thirteen possibilities:

1. <escape> is typed, which causes the current message number to be echoed to you and the relevant process performed on that message only.

2. <control-I> is typed, which causes the previous completely specified <MSG-SEQUENCE> to be echoed and processing performed on that message stream.

3. R which stands for "Recent messages" only.

4. O standing for "Old messages" only.

5. A standing for "All messages" and which is equivalent to 1:(last message number).

6. D standing for "Deleted messages". This is valid ONLY in the context of the Headers, Undelete, and Delete commands. Everywhere else, the headers of the deleted messages will be printed. Of course, you can delete the typeout of those headers by typing control-O.

7. U standing for "Undeleted messages".

8. I standing for messages in inverse order. This is the opposite of the A (for all messages) sub-command.

9. S for "Subject field search for string" which asks you to provide a string which will be used as a mask match on the subject field of the message headers.

10. F for "From field search for string" which is like S but searches the Author field of the message headers instead. NOTE: the header command prints the initial part of the To: line of the message (if it exists) is the message was sent by the login-directory. Therefore, to search for messages sent by yourself, specify the string "To:" rather than the login directory name.

11. E standing for "Examined messages", i.e. all messages which have been completely typed (with the T command) or listed (with the L command).

12. N standing for "Not examined messages", which is the opposite of the E sub-command.

13. L standing for the "Last message sequence" that was completely specified.

Types (i) and (j) require you to type a string terminated by <return>. Typing just a <return> (i.e. the null string) means that searching is not to be performed. Otherwise, the search will be performed on the string typed up to (but not including) the <return>. The string you type must be an exact match to some substring of the appropriate field, but all alphabetic characters are treated as

being upper case. (Note: carriage-returns in the subject field of the header listing are ignored.)

(6) (Note: This is an experimental feature which may change in the future.) If you type comma or "M" as the first character of the message sequence that you are specifying, you will be able to specify more than one of the options drawn from the first five items mentioned here. You will then be entered into a sub-command mode. Any of the standard message sequences are acceptable as input. To terminate the specification of the list of message sequences, just type a carriage return in response to the prompt. If you wish to abort the acquisition at any time, type "Q" (for Quit) or control-N (↑N). To abort the acquisition of a single message sequence (like 3:14), type rubout. Typing rubout at the sub-command level (i.e. at the prompt without typing anything first) will have the same effect as typing control-N.

The default message sequence is 'All messages'. Any message sequence specified causes an intersection to be taken between that single message sequence (like 'Examined'), and the previous total. For example, the sequence:

```
<- Headers ,
<<- Examined
<<- From string: SYSTEM
<<-
```

would cause only the headers corresponding to messages from SYSTEM which have already been typed to get listed on your terminal.

If you just want to add a message sequence to the list, preface the actual message sequence with a "P" (for Plus) or "+". If you want to just subtract a message sequence from the list, preface the actual message sequence with an "M" (for Minus) or "-". For example,

```
<- Headers Multiple message sequences
<<- Examined
<<- From string: SYSTEM
<<- Plus: Subject string: MSG
<<- Minus: Deleted
<<-
```

will list the headers for all undeleted messages about MSG or which are examined messages from SYSTEM. No further associations between msg-sequence specifications are currently allowed.

In the command format below, everything that the program types will be lower case and everything you type will be in UPPER CASE. This is not the case when using MSG, but is used here for clarity.

7.1 MSG COMMANDS

<- Headers (message sequence) <MSG-SEQUENCE>

The headers for messages will be typed out for those messages defined by the message sequence typed. Headers corresponding to deleted messages have an asterisk printed before the header for that particular message. The headers for recent messages are preceded by a plus sign (+); messages which have not yet been typed are preceded by a minus sign (-), and deleted messages are preceded by an asterisk (*). If the message was sent by the user of the LOGIN directory, the initial part of the To: field of the message will be printed in the author field of the header, if the To: field exists in the message. In order to get the length of the message typed out along with the header, use the I command (which stands for Inclusion of length in header).

<- Delete (message sequence) <MSG-SEQUENCE>

This command will indicate (by a preceding asterisk) in the header information for the messages specified by <MSG-SEQUENCE> that those message are deleted. NOTE: This command marks each message in the actual message file indicating that it is deleted. If you reread the file for some reason, the messages will still be marked (and treated) as deleted (but not expunged). This command does, however, affect message numbers specified in later commands in the following way. If you have deleted message number 5 and then try to "Type" or "Put" message number 5 either directly or implied by the use of the ":" option, the deleted messages will NOT be included.

<- Undelete (message sequence) <MSG-SEQUENCE>

Of course! If you can delete a message, you certainly ought to be able to undelete it. This command undoes the action of the Delete command for the messages specified by this <MSG-SEQUENCE>.

7.1.1 Commands To See And Move Messages

<- Type (message sequence) <MSG-SEQUENCE>

This command will type on your terminal the messages specified by <MSG-SEQUENCE>. All messages which are completely typed are treated as having been 'examined'.

```
<- Put (message sequence) <MSG-SEQUENCE>
   into file name: <FILE-NAME>
```

This command will put the messages specified by <MSG-SEQUENCE> into the file specified by <FILE-NAME>. If the file does not exist, it will create that file and write the messages into it. If the file already exists, it will append the messages to the messages already in the file. This command is useful if you want to keep separate files containing messages concerning different topics.

```
<- Move (message sequence) <MSG-SEQUENCE>
   into file name: <FILE-NAME>
```

This command is a convenient combination of the Put and Delete commands. As the messages are put into the file, they are marked for deletion. If any of the messages are already deleted, you will be informed, and those messages will NOT be moved to the file.

```
<- List (message sequence) <MSG-SEQUENCE>
   on file: <FILE-NAME>
```

This command lists all the specified messages on the file specified. All messages specified by the <MSG-SEQUENCE> are treated as having been examined (typed). If you are listing more than one message, a preface page on the file containing the headers for those messages will be created. You will be asked if you want each message on a separate page. This command is intended to allow a user to obtain a reasonable hard copy listing of some messages. (Note: the preface page of headers might have the length of each message included depending on the setting by the I(nclusion of length in header) command.)

7.1.2 Commands To Update Your Message Files

```
<- Overwrite old file <FILE-NAME> [confirm]
```

This command will overwrite the current file (specified by <FILE-NAME>), reflecting the fact that you have deleted messages. That is, if you delete message 2 and then "overwrite" your file, message 2 will disappear from that file. It also rereads your file, renumbering your messages. You are warned if any unexamined messages (which are also not deleted) exist in the file that you are overwriting.

```
<- Quit [confirm]
```

This command returns you to the TENEX EXEC without rewriting any file (almost equivalent to typing control-C). You are warned if any unexamined messages (which are also not deleted) exist in the current message file.

```
<- Exit and update old file <FILE-NAME> [confirm]
```

This command is another way to Overwrite your old message file, but instead of rereading the file it returns you to the TENEX EXEC. This is equivalent to doing an overwrite followed by a Quit, but without the overhead of rereading the file. You are warned if any unexamined messages (which are also not deleted) exist in the file that you are overwriting.

```
<- Write file <FILE-NAME> sorted by message arrival time
```

This is similar in nature to the Overwrite command, except that the messages are sorted into ascending sequence by their arrival time before the overwriting is attempted. The file is then rescanned. You are warned if any unexamined messages (which are also not deleted) exist in the file that you are sorting.

7.1.3 Commands To Read Other Message Files

```
<- Read file name: <FILE-NAME>
```

You can use MSG on any file which has a "message format." This means you can peruse or modify files created with the "Put" or "Move" commands (but NOT the "List" command). If, for example, you have a file containing messages pertaining to MSG problems, you can read it to make sure you've taken care of them. Read is the command which lets you read files other than your standard mailbox file (MESSAGE.TXT;1 or MAIL.TXT.1). It also prints out the recent header information for that file. If that file has old messages which have not yet been 'examined', you will be informed. You will also be told if any of the old messages in the file are deleted.

7.1.4 Commands To Sequence Through The Messages

```
<- Current message is nn of mm messages.
   in file: <FILE-NAME>
```

This command tells you (1) the number of the current message, (2) the total number of

messages, and (3) the file name of the currently active file. The current message is either the last message typed on your terminal or, if you have not typed one yet, either after the last message if the file had no recent messages, or before the first recent message. This command will let you know where the Next and Backing up commands will start, i.e. the first message they will type if used. Finally, it will tell you what the currently active message file is.

<- Go to message number: <NUMBER>

This will allow you to change the Current message number explicitly. If <NUMBER> is not in the range of acceptable numbers (i.e. it is less than 1 or greater than the number of messages in the file), or you did not type a number, you will be told and the Current message number will not be changed. However, there are several other options which are specified by the FIRST character typed:

a. E for the end of messages (the last message)
b. L for the last message (same as E).
c. B for the beginning of messages (message number 1)
d. escape (alt-mode) for current message number

<- Next message is:

This command types the message following the current message (if one exists) and sets the "current message" to be that message. Deleted messages will not be typed, but the "current message number" will still be incremented.

<- <line feed>

Same as Next. Types the message following the current message, and sets the current message to be that message.

<- Backing up -- previous message is:

This command always types the previous message (i.e., Current message number - 1). It is the inverse of the Next command. It always decrements Current message number.

<- ↑

This is equivalent to the Back command. It types the previous message and sets the current message to be that message.

<- ↑H

The <control-H> (or New-line) command is equivalent to the Back command. It types the previous message and sets the current message to be that message.

7.1.5 Other Commands

<- Verbose

This is a binary switch which causes the program to go into either 'Short typeout mode' or 'Long typeout mode', and tells you which is the setting that it changes to. The default is 'Short typeout mode'. Long typeout mode gives additional prompting about what is expected to be typed in.

<- Koncise

This is a binary switch which causes the program to go into either 'Concise typeout mode' or 'Short typeout mode' (the default), and tells you which is the setting that it changes to. Concise typeout mode shortens some of the typeout that MSG gives when it is interacting with the user. It is meant for 'advanced' users only.

<- Inclusion of length in header

This command is a binary switch which causes the program to go into a mode where header listings caused by the Header command will have the number of characters in the message included as part of the subject field. The default is that the length will not be included. Note that when you read a file initially, the length of 'recent' messages will always be included in the initial listing of recent headers.

<- Zap profile [Confirm]

The Zap profile command will allow you to set up a user profile file for yourself without having to know the format of such a file. For the time being, the profile information will be limited. Typing control-N will exit you to the command level of MSG. Typing E at any point will 'Exit' the dialogue and ask you if you want the changes made permanently. At any point, type "?" to determine the appropriate responses. The following is a summary of the questions posed:

1. Normal, Verbose or Koncise typeout mode?
2. Always include the length of messages in all headers listings?
3. When in SNDMSG (from any of the Sndmsg, Forward or Answer commands), when you type control-N, do you want to abort Sndmsg without confirmation?

4. Do you want to be required to confirm all commands with a single carriage return?
5. Do you want to be told that some messages have been 'not-examined' whenever you try to quit MSG (by any of the Quit, Overwrite, or Exit and Update commands)?
6. Do you want to receive copies of your 'answers' to messages?
7. If the answer to (6) is yes, then you will be asked if you want to save all your 'answers' on the file SAVED.MESSAGES.
8. Do you want a list of headers for all messages:

    a. being deleted with either the "Move" or "Delete" commands
    b. being moved with the "Put" command
    c. being listed with the "List" or "Xerox" command
    d. being marked or unmarked with the "'" or "-" commands.

At the end of the dialogue, you will be asked if you want these changes in mode settings to be made permanent. If you answer 'Y' then each time you start up MSG, the settings of the modes noted here will be set to the values you indicated. Otherwise, the settings are set only for this session. They are NOT permanent, and can be changed any time.

<- : (prints current time and date)

<- ' Mark messages as examined (message sequence) <MSG-SEQUENCE>

This command will mark all the messages specified by <MSG-SEQUENCE> to be "examined", so MSG will think they have been typed or listed even though they may not have been.

<- - Unmark messages to be Not examined (message sequence) <MSG-SEQUENCE>

This command will mark all the messages specified by <MSG-SEQUENCE> to be "NOT examined", so MSG will think they have NOT been typed or listed, even though they might have already been seen.

<- ; <COMMENT>

This command is mainly intended to allow you to talk with somebody over a link while you are in MSG. It eats all characters except <return>, control-Z (↑Z) and control-N (↑N), which return you to the command level of MSG. Two other characters have special effects. <delete> (<rub-out>) will type the string ' XXX ' and is useful in indicating that the previous word (or phrase) should be ignored. <line-feed> will cause effectively a carriage return and tab sequence to be typed. This way you can type more than one line of text. NOTE: the

standard TENEX and TOPS-20 editing characters (e.g., ↑A or the DEL key) are treated like any other character and perform no special function.

7.1.6 Command To Run Other Programs

```
<- Sndmsg [confirm]
```

This command will start up SNDMSG and give control of the terminal to it. When SNDMSG is finished (i.e., when you have sent the message), it will turn control back to MSG in the same state as it was before you sent the message. Control-N (↑N) will ask if you wish to abort. If you provide a positive confirmation, then you will be returned to the top level of MSG. Otherwise, you will be returned to SNDMSG.

```
<- Answer message number: <MESSAGE-NUMBER>
   Reply to those whom the message is: <ANSWER SUB-COMMAND>
```

This facility allows you to send a message to the sender of a message, and (at your discretion) those people to whom that message was sent, without having to type their addresses to SNDMSG.

```
The <ANSWER SUB-COMMAND> can be any of the following:
 F -- From (indicating the sender of the message only)
 T -- To (indicating the sender of  the  message  and  those
      addresses in the To: list)
 C -- Cc (indicating all recipients of the original  message
      in addition to the sender in the message)
```

Typing anything else aborts the command.

```
The <MESSAGE-NUMBER> can be any argument that the Go command takes:

    a.  a message number
    b.  E for the end of messages
    c.  L for the last message (same as E).
    d.  B for the beginning of messages (message number 1)
    e.  escape (alt-mode) for current message number
    f.  <return> for current message number.
```

The header of the message specified is also typed so that you may be sure you are answering the correct message. In fact, the header is typed after you have specified the message number, but before you are asked to supply the sub-command.

When prompted for additional addresses, any that you specify will be passed to SNDMSG as part of the cc: list. Some of the SNDMSG conventions are NOT implemented. These are the control-B feature which allows specification of a file, and the feature which allows you to specify a global host name (which spreads across several user names). Also, control-N aborts the Answer command! Local user names and remote host names are checked for validity.

An attempt is made to insure that all addresses are valid (i.e., all host names on remote addresses, and user names on local addresses), and that no duplications are present. If clarification is necessary from the user, you may be asked some questions. If these questions are posed, all type-ahead is deleted. If relevant, MSG will issue a warning if either the To: or cc: destination fields of the message have a destination list as part of the field (like LISP-USERS:). When control is given to you to type your answer, you will be typing to the message acquisition portion of SNDMSG (i.e., that part which normally would prompt you by typing "Message (? for help):"). Control-N (↑N) will ask if you wish to abort. If you give positive confirmation, then you will be returned to the top level of MSG. Otherwise, you will be returned to SNDMSG.

There are two relevant profile mode settings. One is whether you wish to receive copies of the answers you send. It is initially assumed that you do. If you do not want copies of the replys you send, then your name will not appear in any of the destination lists unless you specify it as part of the additional carbon-copy list. However, if you do want copies of your messages, you will always receive one. In addition, if you have also indicated in your profile that you want all your responses to go to a file called SAVED.MESSAGES, then if that file exists, your responses will go in that file and NOT into your mailbox file (MESSAGE.TXT;1 or MAIL.TXT.1).

However, if you do not always want your answers to go to SAVED.MESSAGES, but do want copies of your answers, if there is a file named SAVED.MESSAGES in the login directory, you will be asked if you want your copy of the message to go to that file. If a positive response is given, then the login directory name will NOT appear in the destination lists.

<- Forward (message sequence) <MSG-SEQUENCE>

This facility will allow you to send copies of messages you have received to other people. The headers of the messages being forwarded are typed, after which you will be asked to provide the subject of this forwarded message. Then it will hand SNDMSG the subject and those messages you want forwarded, and leave you in SNDMSG in such a way that the message being forwarded can be edited, or your own comments added. You will be left in SNDMSG as though you had typed the forwarded message in yourself. When done, type a control-Z and then specify, in the standard way, to whom the mail is going. Once in SNDMSG, typing control-N (↑N) will ask if you wish to abort. If you give a positive confirmation in the standard way, then you will be returned to the top level of MSG. Otherwise, you will be returned to SNDMSG.

<- Jump into lower fork running: <FILE-NAME>

This command is an escape in MSG in case you wish to run another program such as TECO, PUB, the EXEC, and so on. It searches directories to try to find the program you are asking it to run. The search list is, in order, <SUBSYS>, <SYSTEM>, your connected directory, and the login directory (if it is different from your connected directory). This way, you can run EXEC without having to type the complete information (i.e., <SYSTEM>EXEC.SAV).

If you decide to leave the lower fork, but want to continue it at a later time, all you need do is type an escape as the first character of the file name you are requested to provide. This will cause the old file name (preceded by an appropriate message) to be printed, and then you will be asked to confirm in the standard way. If you provide a positive confirmation, you will be asked if you want to continue or start that program. Typing 'C' for continue will put you back in the lower fork at the place where you exited; typing 'S' for start will restart the program.

<- Xed (editor) [confirm]

This command will start up XED (a text editor written at ISI). It has the capability to give SNDMSG the text built while in the editor as the body of the message. When you "Quit" XED you will return to MSG. Each additional time that you execute the XED command, you will be returned to the SAME copy of XED that you previously left with the XED "Quit" command (the old text buffers are left intact).

<- ^Exec [confirm]

When you type control-E, the program will type "Exec" to you and ask for confirmation. This command is intended to give you a new copy of the EXEC with a minimum of hassles. To leave that EXEC and return to MSG, type "QUIT". If you decide that you want a copy of the EXEC again, and you use this command, you will be given the same EXEC with all of your context intact.

7.1.7 Command Summary

Cmnd. Char.	Meaning
A	Answer message number: <MESSAGE-NUMBER> Reply to whom the message is:

```
                        F -- From
                        <return> -- same as F
                        T -- To list plus original sender
                        C -- Cc list plus To: list plus original sender
  B         Backing up -- previous message is:
  ↑         Same as Backing up
 ↑H         Same as Backing up
  C         Current message is nn of mm messages
              in file: <FILE-NAME>
  D         Delete (message sequence) <MSG-SEQUENCE>
 ↑E         Exec [confirm]
  E         Exit and update old file <FILE-NAME> [confirm]
  F         Forward (message sequence) <MSG-SEQUENCE>
  G         Go to message number: <MESSAGE-NUMBER>
  H         Headers (message sequence) <MSG-SEQUENCE>
  I         Inclusion of length in header
  J         Jump into lower fork running file:  <program name> [confirm]
  K         Koncise -- provides shorter prompting
  L         List (message sequence) <MSG-SEQUENCE>
              on file name: <FILE-NAME>
  M         Move (message sequence) <MSG-SEQUENCE>
              into file name: <FILE-NAME>
  N         Next message is:
<lf>        (line feed) same as Next message is:
  O         Overwrite old file <FILE-NAME> [confirm]
  P         Put (message sequence) <MSG-SEQUENCE>
              into file name: <FILE-NAME>
  Q         Quit [confirm]
  R         Read file name:  <FILE-NAME>
  S         Sndmsg [confirm]
  T         Type (message sequence) <MSG-SEQUENCE>
  U         Undelete (message sequence) <MSG-SEQUENCE>
  V         Verbose -- provides more prompting
  W         Write file <FILE-NAME> sorted by message arrival time [confirm]
  X         Xed  [confirm]
  Z         Zap profile [confirm]
  '         '   Mark messages as 'examined' (message sequence) <MSG-SEQUENCE>
  -         -   Unmark messages to be NOT 'examined' (message sequence)
                                                    <MSG-SEQUENCE>
  :         :   (the time and date is then printed)
  ?         ?   Type command character for its description, ? for summary
  ;         ;   Comment -- <return> or ↑Z returns you to command level
```

Abort commands on typein and terminal output with control-N (↑N). Confirm with Y or <return>.

7.2 RECEIVING NEW MESSAGES WHILE USING MSG

MSG, on typing a command or returning from the execution of a command, checks to see if your currently active message file, usually MESSAGE.TXT;1 or MAIL.TXT.1, has been written into. If it has, it prints out that fact and the headers for the new messages. The "current message number" is not modified. It then executes your command or returns to command mode, accordingly.

7.3 ERRORS WHILE READING A MESSAGE FILE

When reading a file in MSG (either at startup or with the 'Read' command), the file MUST be in the so-called "message file format". If MSG recognizes that the file does NOT conform to this format, you will be told so. The following are the circumstances which might cause the file to become unreadable, and some suggestions for getting around the problems.

The file is a message file (that is, one or more valid messages have been read from it), but somewhere in the middle it does not conform to the message file format. It could be:

> (1) It has a hole in it. Read the file with a text editor to get rid of the hole, and write it back out, and reuse MSG. Try this first.

If this doesn't work, MSG will give you an error at the same place. Then you can try the second suggestion:

> (2) If suggestion (1) didn't work, then the file has internal byte counts which do not match the actual file. Either you used a text editor on your message file changing the number of bytes but not the byte counts or your file was mysteriously altered. The date of a message could not be read. Either the byte count for the last message read was wrong, or there is junk between the last message read and the one with the error. Using some editor, find the last message read. The first line of that message contains a date-and-time followed by a byte count indicating how many characters are in the message body starting on the following line. Skip that many characters of the message body. You should be at the date-and-time line of the next message. If there is junk there, delete it. Otherwise, try to fix the count so it is pointing at the date-and-time of the next message.

The beginning of the file does not conform to the message file format. It could be:

> (1) the file is not a message file -- sorry, we can't help you there.

(2) It is a message file with a bad first line -- probably a blank line. Read the file with a text editor. If the second line begins with a time and date, then delete the first line and reuse MSG on the new file.

(3) It is a message file with a hole at the beginning. Read it with a text editor to get rid of the hole, write it out and reuse MSG.

8. INTRODUCTION TO HERMES

The following was excerpted from Bolt Beranek and Newman, Inc. THE HERMES MESSAGE SYSTEM on-line documentation.

The HERMES Message System is a computer program for sending and receiving messages over a computer network. The HERMES System has features that help the user read messages, compose messages for sending, and create and manage files of messages. The HERMES Message System is one product of Project HERMES, a Bolt Beranek and Newman, Inc., research and development effort in advanced forms of message technology. Project HERMES is supported by the DoD Advanced Research Projects Agency, by the Naval Electronics Systems Command, and by the U. S. Army Development and Readiness Command. The HERMES program was first implemented for host computers using the TENEX Executive System developed at Bolt Beranek and Newman, Inc. for the DEC PDP-10 computer. In 1977, HERMES was modified to run on the TOPS-20 Operating System. In this documentation, the operating system is referred to as "TOPS-20" or the "Exec".

The HERMES Message System contains an indexed set of on-line documentation. Without entering any special subcommand mode you can obtain succinct information or extensive documentation about any topic. The documentation is available using four HERMES commands: DESCRIBE, OUTLINE, EXAMPLE, and DOCUMENT. The commands are used to request information about specific topics in the HERMES system. There is also a HELP facility for new users and a NEWS facility to keep users posted on recent modifications to the HERMES system. The HELP command is designed to aid users just getting started with HERMES. HELP presents basic topics and asks the user to choose the ones to be printed out. For basic instructions for using the HERMES system, type HELP:

```
>HELP<CR>
```

HELP tells you how to use the on-line documentation commands, OUTLINE, DESCRIBE, EXAMPLE and DOCUMENT, which provide detailed information about the system. The NEWS command provides users with a summary of recent modifications to the HERMES system. In addition to NEWS, there is a topic, NEW-FEATURES, which can be accessed by the *normal* documentation commands DESCRIBE, OUTLINE, EXAMPLE, and DOCUMENT. Any changes mentioned in NEWS are also available as subtopics of NEW-FEATURES. To obtain a complete list, type:

```
>OUTLINE NEW-FEATURES ALL<CR>
```

To print the latest News on your terminal, type NEWS.

```
>NEWS<CR>
```

You can also output News or Help to the line-printer or a file.

```
>NEWS LPT:<CR>
or
>NEWS <file-name><CR>
>HELP LPT:<CR>
or
>HELP <file-name><CR>
```

If you have suggestions or questions, please use the SUGGESTION command:

```
>SUGGESTION<CR>
```

This is a quick and easy way to send a message to the Hermes staff.

8.1 PANIC BUTTONS

<CTRL-O>	stops print-out of text anywhere; prints ↑O.
<CTRL-U>	stops a command before the final <CR>.
<CTRL-E>	stops a command at any time; prints ↑E.
<CTRL-C>	stops HERMES anywhere and returns you to the operating system; prints ↑C. You can recover by typing CONTINUE to the @ prompt.

To type a control character, e.g., <CTRL-C>, hold down the CTRL key while you type the letter C.

NOTE: Occasionally, by accident, you may get into a program that runs under Hermes in a "lower fork". Such a program will have a different prompt (such as "*"). In this situation, <CTRL-C> will return you to HERMES.

8.2 EDITING CHARACTERS

```
<RUBOUT>   deletes a single character.
<CTRL-W>   deletes a single word.
<CTRL-U>   deletes a line of text.
<CTRL-R>   retypes a line of text.
<CTRL-Z>   ends the Text:-field; prints ↑Z.
```

8.3 TYPING COMMANDS

You can use either the space character or the Escape-key (or ALT MODE) to separate the different sections within a command. The Escape-key <ESC> causes HERMES to print out completely any word used in command and then to print "noise words" to introduce the next word, if it is possible to type one. At such a point, typing <ESC> causes HERMES to insert the "default" word. Most commands have a set of defaults, chosen to be most useful to the beginning user. For example, PRINT prints the current message in the form specified by the PTEMPLATE onto your terminal (symbol TTY:). If the current message is No. 10, and you type "PR", followed by a series of <ESC>'s you will see:

```
>PRINT (messages) 10 (using template) PTEMPLATE (on file) TTY:
```

You may type "?" at almost any point in the HERMES system to see what words you are allowed to type in next.

```
>?          gives a list of top-level HERMES commands.
>SU?        lists commands that begin with "SU".
>SURVEY ?   lists all the things that you may type in at
            that position in the "SURVEY" command.
```

Use ? and <ESC> freely.

```
?           tells what choices you have.
<ESC>       tells what defaults the system has set up.
```

To find out what can be typed at a given point, type "?". The system prints all possible choices on your terminal. Use the "?" and <ESC> features freely; they reduce what you have to remember and help you understand the range of the system.

8.4 ENTERING HERMES

When you see the Exec prompt "@", type HERMES, followed by a carriage return. The HERMES Message System responds with the HERMES prompt ">", and prints brief surveys of any messages that have arrived since your last session. The message-file you see when you enter the HERMES system is your INBOX, which has the file-name MAIL.TXT.1.

When you enter, you can tell HERMES to input a different message-file by typing

```
@HERMES,<CR>
 (using message-file)  <file-name><CR>
```

To direct the attention of the HERMES system to another message-file, after you enter HERMES, type

```
>GET <file-name><CR>
```

When you first enter HERMES, your FILENAME-INPUT switch is set to TENEX or TOPS-20, and your file-names are handled as in the operating system. For simplified file-handling within HERMES, you can set your FILENAME-INPUT switch to HERMES.

```
>Edit SWITCHES<CR>
>>Set FILENAME-INPUT HERMES<CR>
>>Done<CR>
```

All files now have single-word names, which are extended and recognized like other words in HERMES commands. For example, your INBOX is named MAIL.TXT. If you have a file named NEWFILE, and no other files beginning with NEW, you can now type:

```
>Get NEW<CR> (CONFIRM)<CR>
```

To see a list of your message-files, type

```
>Show MESSAGE-FILES<CR>
```

For more information, type

```
>DESCRIBE FILENAME-INPUT-SWITCH<CR>.
```

8.5 LEAVING HERMES

When you want to leave HERMES, type QUIT (or simply Q) to the HERMES prompt ">", followed by a carriage return. QUIT<CR> returns you to the Exec.

```
>QUIT<CR>
```

If you have deleted any messages, HERMES asks whether you want to expunge them (i.e., take them away forever). If you see the prompt ">>" or ">>>", you must first type "DONE".

```
>>DONE<CR>
>QUIT<CR>
```

After you QUIT, you may continue by typing CONTINUE to the @ prompt:

```
@CONTINUE<CR>
>
```

If you wish to logout directly from HERMES, type LOGOUT, followed by a carriage return. This is equivalent to typing QUIT<CR> to the HERMES prompt ">", and then typing LOGOUT<CR> to the TOPS-20 prompt "@".

```
>LOGOUT<CR>
```

Examples:

```
@HERMES<CR>
HERMES 4.0.17 26-AUG-77
-+8  252  29 SEP 77 Wolf at BBN-TENEXD MIT Press annual book sale
[HERMES session]
>QUIT<CR>
@
```

or

```
>LOGOUT<CR>
[TOPS-20 logout message]
```

8.6 READING MESSAGES

For a quick survey of all messages in your message-file, type

```
>SURVEY<CR>   or   >S<CR>
```

To SURVEY specific messages, e.g., 3, 1, and 4 through 6, type

```
>SURVEY 3,1,4:6<CR>    or   >S 3,1,4:6<CR>
```

To print messages in full, one at a time, type the LINE-FEED-key. The HERMES System steps through the messages, always printing the next message in the message-file. To back up and print the previous message, type the up-arrow character "↑". To print the current message, use the PRINT command. You need type only

```
>PRINT<CR>    or    >P<CR>
```

To PRINT other messages, specify as in SURVEY:

```
>PRINT 3,1,4:6<CR>    or    >P 3,1,4:6<CR>
```

The PRINT command resets the CURRENT message to the last message printed. If you now type LINE-FEED, HERMES prints message 7. HERMES tells you when you have messages that have arrived recently. HERMES prints a "survey" of each message, when you log in or when a new message arrives:

```
-+    1  536 15-Jul-77 MOOERS at BBN-TENEXA HERMES HELP INFORMATIO
```

For a quick survey of all messages in your current message-file,

```
>SURVEY<CR>
```

To print the text of a single message:

```
>&lt;LF&gt;                The LINEFEED key (NOT followed by <CR>)
                    prints the NEXT message, AND
                    sets it to be the CURRENT message.
```

When a new message arrives, it is generally the NEXT message, unless you have been skipping around in your file.

```
>↑                  The UP-ARROW or CARET key (NOT followed
                    by <CR>) prints the PREVIOUS message
                    AND sets the CURRENT message.
>PRINT<CR>          prints the CURRENT message.
```

To print a survey of all messages from JONES, type

```
>SURVEY FROM JONES<CR>
```

To SURVEY all messages that are from Jones and to Smith, type

```
>SURVEY FROM JONES/TO SMITH<CR>
```

To PRINT all messages that are from Jones and have the words "message service" in their subject fields, type

```
>PRINT FROM JONES/SUBJECT "MESSAGE SERVICE"<CR>
```

If you search for only one word, you can omit the quotes.

8.7 SENDING MESSAGES

To send a message, type "COMPOSE" to the Hermes prompt ">":

```
>COMPOSE<CR>
```

The COMPOSE command gives you a series of prompts that guide you through composing a message. The To: and Cc: fields must be filled in with names of "directories" on ARPANET computers. The To, Cc and Subject fields end with <CR> but the Text field ends with <CTRL-Z>. When you are done composing, HERMES allows you to format and send the message. You should check your PTEMPLATE using the SHOW command to find out what switches are set. If the format switch is set to NO you may change it to either ASK or YES using the SET command. After you type <CTRL-Z>, the system asks

```
SEND?:                    You may answer YES<CR> or NO<CR>.
```

If you answer YES<CR>, HERMES SENDS your message. If HERMES is able to deliver it immediately, it tells you that the copy of the message to an addressee is "delivered". If not, HERMES queues the message for another program,named MAILER, which picks the message up a few minutes later and delivers it. HERMES does not tell you which messages are queued. If you type NO<CR>, HERMES does not SEND the message. Instead, you can do more work on the message. For example, you can add another name to the CC: field:

```
>>CC: YOURNAME<CR>    Notice the double prompt ">>".
```

When you want to send the message, type SEND<CR> and "confirm" with a second <CR>.

```
>>SEND<CR> (CONFIRM) <CR>
>
                          Using the SEND command takes you back
                          to the ">" prompt.
```

If you change your mind about sending, after the computer prints (CONFIRM), you can stop it with <CTRL-U>. Some commands, such as SEND and ERASE, cause irreversible actions to take place. When you terminate such a command with the usual <CR>, Hermes asks you to confirm by printing "[Confirm]" on your terminal. You may then either

a) confirm with another <CR>, or
b) abort the command:

2) Type <CTRL-U>. The <CTRL-Q> key has the same effect as <CTRL-U> unless you have a scope terminal which you set to "terminal page" mode.

Type either Yes<CR> or No<CR> in response to a question. Yes and No may be abbreviated. Examples:

```
>>SEND <CR>
 [Confirm]<CR>          The message is sent.
>>SEND<CR>
 [Confirm]<CTRL-Q>      The SEND command is aborted.
>FILE 1:3,5 SAVED<CR>
Delete messages after filing? Y<CR>
1:3,5
                        Messages are deleted from your
                        message-file.
>FILE 1:3,4 SAVED<CR>
Delete messages after filing? N<CR>
1:3,5
                        Messages are not deleted.
```

To reply to a message in your message-file:

```
>REPLY<CR>              gives you prompts for your reply to the
                        CURRENT message.
>REPLY 5<CR>            replies to Message 5.
```

The REPLY command automatically sets up fields in the outgoing message (typing them out as it does so) so all that is required of the user is to enter the Text:-field. The user is given the option of replying just to the author of the original message or sending copies to its other recipients as well. To forward a message in your message-file to someone else:

```
>FORWARD <CR>           forwards the CURRENT message, and
                        prompts you for addressees and
                        comments.
>FORWARD 5<CR>          forwards Message 5
```

The FORWARD command prompts for the To: and Subject: fields, sets up the Subject:

field to reference the forwarded messages following the user's subject and then gives the user a chance to enter comments. Following any comments, FORWARD will "package" the messages to be forwarded in the message just built, then request permission to send it.

8.8 NAMED SEQUENCES

To organize your messages, HERMES provides facilities for grouping messages into named-sequences. A named-sequence is an ordered set of messages. Individual messages can appear in as many different named-sequences as you wish. For example, to create a sequence named ANSWERED, you type:

```
>CREATE SEQUENCE ANSWERED<CR>
```

This enters the sequence-editor and HERMES prompts you with a double arrow. To add messages 3,4,5 and 7 to this sequence, type:

```
>>ADD 3:5,7<CR>
>>DONE<CR>
```

Once a sequence is created, you can add messages without entering the sequence-editor. You use the ADD command and specify both the messages and the named-sequence:

```
>ADD 9 ANSWERED<CR>
```

To find out which sequences a message is in, use the WHEREIS command:

```
>WHEREIS 9<CR>
  9 is in ANSWERED
```

To see a list of all messages in a sequence, for example, in ANSWERED, type:

```
>SHOW ANSWERED<CR>
  3:5,7,9
```

Sequences can be used in all of the message-handling command mentioned above. To survey all messages in ANSWERED, type:

```
>SURVEY ANSWERED<CR>
```

8.9 MESSAGE MANAGEMENT

Within a message-file, messages can be organized into NAMED-SEQUENCES, which can be thought of as overlapping "file folders". The command for filing messages is FILE. You must specify the messages and the file you want them filed in. If the file you name does not exist, HERMES creates it. The file-name may contain upper-case letters, numbers or the hyphen "-", and may not be more than 30 characters long.

```
>FILE 3,1,4:6 SAVED-MSGS<CR>
```

If the file is from a different directory than your current one, the file-name consists of the directory name enclosed in angle brackets, i.e., "<SMITH>", followed by the file-name itself.

```
>FILE 3,1,4:6 <SMITH>SAVED-MSGS<CR>
```

The FILE command copies messages from your current message-file, then asks whether you want to delete the original messages. To shift HERMES' attention to another file of messages, for example, SAVED-MSG, type:

```
>GET SAVED-MSG<CR>
```

All subsequent commands will now work with this new file. Examples:

```
>FILE 3,1,4:6 SAVED-MSGS<CR> [New file]<CR>
Delete messages after filing? Yes<CR>
3,1,4:6 filed and deleted.
```

You may answer "Y" instead of "Yes". If you do not want to delete the message, answer "No" or "N".

```
>FILE 7,9,11 SAVED-MSGS<CR> [Old version]<CR>
Delete messages after filing? N<CR>
7,9,11 filed.
```

8.10 MESSAGE DELETING

The command for getting rid of messages is DELETE. To delete the current message, type

```
>DELETE<CR>  or  >D<CR>
```

To delete other messages, type

```
>DELETE 7,9,11:15<CR>   or   >D 7,9,11:15<CR>
```

This marks the messages as DELETED. You cannot SURVEY or PRINT them but you can UNDELETE them:

```
>UNDELETE 7,9,11:15<CR>
```

At the end of each HERMES session, when you QUIT or LOGOUT, HERMES asks you whether you want to have your deleted messages EXPUNGED. When message are expunged, they are physically removed for the file and the remaining messages are renumbered. Examples:

```
>SURVEY<CR>
>DELETE 3,5<CR>
>QUIT<CR>
Messages 1,3:5 deleted.
Expunge and renumber? Yes<CR>
@
@HERMES<CR>
>SURVEY<CR>
```

8.11 ON-LINE DOCUMENTATION

The on-line documentation material is basically a reference manual arranged as short topics with *associated examples*. The topics are organized in a table of contents or OUTLINE.

```
>OUTLINE<CR>
                    shows the first two layers of topic names,
                    and is equivalent to the command
>OUTLINE HERMES (to depth) 2<CR>
```

You may use the OUTLINE command with any topic, and with the depth argument set to 2, 3, 4 ... All. To see the contents of a topic, type

```
>DESCRIBE <topic><CR>
```

The topic defaults to "HERMES" at top command level, to "MESSAGE-EDITOR" when you are creating a message, and to other appropriate topics in other parts of the system. For more information, type

```
>DESCRIBE DOCUMENTATION<CR>
```

Please note that when bugs are discovered in the production version of Hermes, which is installed as <SUBSYS>HERMES, we will try to put up a fixed version as <NEWSYS>HERMES, as soon as possible. Watch the HERMES herald line, or type NEWS<CR> for details.

8.12 HERMES BASIC COMMAND SUMMARY

This summary gives the basic commands for reading and composing messages. HERMES has many more commands and features than appear here.

Type a Carriage-Return, <CR>, after all commands except Line-Feed, <LF>, ↑ and "control characters". Sometimes you may type additional words before the <CR>, to modify the action of the command. Only the most frequently used of these words are shown on this list.

ENTERING and LEAVING HERMES

Prompt: @, >, or >>

@HERMES	Gives you access to the HERMES program.
>QUIT	Leaves HERMES; returns you to operating system

SPECIAL TERMINAL KEYS

The key marked ESC is the Escape-Key, <ESC>. The key marked CTRL is the Control-Key, <CTRL>. To type a Control-Character, *hold down* the <CTRL> and simultaneously type the letter key.

<ESC> and ? -- Use them to find your way around HERMES

<ESC>	Causes HERMES to complete a word, to print "guide words", or to insert a default word. Type <ESC> repeatedly to learn about a command.
?	Causes HERMES to show the words you are allowed to type at at any point, even within a word.

PANIC BUTTONS

Can be used anywhere.

<CTRL-E>	Aborts a command at any time.
<CTRL-O>	Stops printout on your terminal.
<CTRL-C>	Stops HERMES. Returns to the operating system. Recover by typing CONTINUE to the @ prompt. If you are in Teco, Xed or Neted, returns to HERMES.
<CTRL-T>	Shows system is working, tells system load.

EDITING CHARACTERS

For typing commands and editing fields.

<RUBOUT/DEL>	Deletes a single character.
<CTRL-U>	Deletes a line in text; aborts a command.
<CTRL-W>	Deletes a word in text, or a part of a command.
<CTRL-R>	Retypes a line of text.
<CTRL-Z>	Ends the Text:-field, or any TEXT-type field.
<CTRL-I>	Inside a field, inserts a tab. Tab stops are set every 8 spaces. One tab alone on a line turns OFF FORMAT. Two tabs alone turn ON FORMAT.
<CTRL-B>	Inside a field, inserts the contents of a file. Prompts for file-name.
!	Escape from Hermes file-names. Type ! before name.

USEFUL SYMBOLS

.	CMESSAGE (the current message)
*	ALLMESSAGES (all the messages in a message-file)
%	LASTMESSAGE (the last in a file) or the last line in a field.

INFORMATION COMMANDS

```
Prompt:  >

>HELP                  Prints the HERMES MINIMAL REFERENCE SUMMARY
>NEWS                  News of latest changes.
>STATUS                Shows your directory and current message-file.
>DESCRIBE    topic     Information about a large number of topics.
>EXAMPLE               Prints example of topic just DESCRIBEd.
>OUTLINE     topic     Shows a structured outline of topics.
```

READING MESSAGES

```
Prompt:  >

>(LF) or LineFeed      Prints NEXT  message.  Do not type (CR).
>(↑)                   Prints PREVIOUS message.  Do not type (CR).
>SURVEY      messages  Prints a summary of messages.
>PRINT       messages  Prints in full.  Resets current message.
```

MANAGING MESSAGES

```
Prompt:  >

>DELETE      messages  Marks messages deleted.
>UNDELETE    messages  Restores deleted messages.
>FILE  messages file   Copies message into another message-file.
>GET         file      Changes the current message-file.
```

COMPOSING MESSAGES

```
Prompt:  >

>COMPOSE               Prompts you to compose a new message.
>REPLY       message   Prompts a reply. (Default: current message.)
>FORWARD     messages  Places messages in Text: field of a new draft;
                       prompts you. (Default: current message.)
```

COMPOSING A DRAFT MESSAGE: The Draft Editor

```
Prompt:  >>

A Draft is an unsent message, which consists of a collection fields.
You can create a draft with the top-level commands COMPOSE, REPLY or
FORWARD.  HERMES prompts you for the fields.  If you do not SEND the
draft, you are placed in the DRAFT-EDITOR, with >> prompt.
```

SECOND-LEVEL COMMANDS FOR LEAVING THE DRAFT-EDITOR

```
Prompt:  >>

ABORT          Erases draft and returns you to top level.
DONE           Does not erase draft.  Returns you to top level.
SEND           Distributes draft as a message to addressees in To:,
                Cc: and Bcc: fields, and to files in Fcc: field.
```

FIELDS THAT CAN BE USED AS SECOND-LEVEL COMMANDS

```
Prompt:  >>

HERMES recognizes a number of standard fields, in five types.

Data-Type     Contents of Field
  ADDRESSEE   Network addresses, of the form NAME or NAME@HOST.
               Only the To:, Cc: and Bcc: fields  cause SEND to
               distribute messages to addressees.
  DATE        Accepts a single date, in any reasonable format that
               includes day, month and year.
  LINE        Plain text, ending with a <CR>, like the Subject:.
  TEXT        Plain text, ending with a <CTRL-Z>, like the Text:.
  FILE        Fcc: only.  SEND distributes messages to files in Fcc:.


Field Name       Data-Type
  Bcc:           ADDRESSEE  Blind Carbon Copy.  Bcc names are seen
                             only by others on the Bcc list.
  Cc:            ADDRESSEE  Carbon Copy.
  Fcc:           FILE       File Carbon Copy.
  From:          LINE       SEND fills in From: if you do not.
  Keywords:      LINE
  Subject:       LINE
  To:            ADDRESSEE
  Text:          TEXT       Allows <CR>s; ends with <CTRL-Z>.
```

Some fields are seldom used as commands but are filled in by HERMES:

Date:	Date message was sent.
In-Reply-To:	Filled in by the REPLY command.
Message-ID:	Unique ID composed of Host, Date and Sender.
Redistributed-By:	Filled in by the REDISTRIBUTE command.
Redistributed-Date:	" " " " " "
Redistributed-By:	" " " " " "
Sender:	Login directory of user who SENDs the message.

Some standard fields are seldom used: Start-Date, End-Date, Suspense-Date, Class-Char, Message-Class:, Special-Handling.

SIMPLE SECOND-LEVEL COMMANDS FOR EDITING THE DRAFT

Prompt: >>

You may omit the "line" and type only the field name. Two switches control the CHANGE-BELL (bell or !) and the SHOW-FIELD-DEFAULT (lines in single fields shown numbered or unnumbered).

ERASE	lines field	Erases specified lines. (Default: entire draft. You may type a field name or HEADER, too.)
FORMAT	lines field	Adjusts lines in a field to 65-character maximum. (Default: Text:)
INSERT	line field	Lets you insert text before a specified line in a field. Ends with carriage-return or <CTRL-Z>.
SHOW	lines field	numbered/unnumbered Displays specified lines of a field on your terminal. (Default: entire draft. You may type a field name or HEADER, too.)
SPELL		Runs program to correct spelling. (Text: only.)

THE CHANGE COMMAND

CHANGE has a set of single-character subcommands. These are not printed on your terminal as you type them, but they cause some conventional signs /, and ! (or a bell) to be printed, similar to the signs printed by the <CTRL-A> and <CTRL-W> characters.

CHANGE line field numbered/unnumbered

Lets you change a line by copying from an "old" version into a "new" version.

Subcommands of CHANGE (used within line, not echoed on terminal):

<space>	Copies one character from old line to new.
n<space>	Copies n characters.
<CR>	Copies the rest of line and ends CHANGE.
E	Copies to End of line.
Sc	Searches for any character c. Attempts to match case (upper or lower) of c, then searches for other case. Copies up to c.
S<space>	Searches for a space; thus copies to the end of the next word.
nSc	Searches for nth c, then copies up to nth c.
B	Breaks line into two lines. (To combine two lines, leave CHANGE and use FORMAT.)
I	Begins an Insertion. End with <CR> or <CTRL-Z>. (Rings bell or prints !s.)
D	Deletes next character to right. (Prints /s.)
nD	Deletes n characters to right.
<RUBOUT/DEL>	Deletes character to left. (Prints s.)
n<RUBOUT/DEL>	Deletes n characters to left.
<CTRL-U>	Deletes line to left.
<CTRL-W>	Deletes one word to left. (Prints <s.)
n<CTRL-W>	Deletes n words to left.
<CTRL-R>	Retypes line.
<CTRL-D>	Retypes entire field.

9. MAILSTAT PROGRAM

MAILSTAT lists all queued and undeliverable mail in the connected directory. It also accepts commands to manipulate the undeliverable messages - they can be deleted or put back on the queue to be mailed (with a different address if desired).

Example:

```
@mailstat
MAILSTAT 1B(11)
Type ? for help
*?

MAILSTAT lists existing queued and undeliverable mail in the
connected directory.  It also permits manipulation of
undeliverable mail.

Commands are:

Q - list queued mail
U - list or manipulate undeliverable mail
H - halt - exits program
? - types this message
carriage return - lists both queued and undeliverable mail

*Queued mail:

COLE     6-2-78 16:37:57
No undeliverable mail.

*u
type ? for help
**?
```

Commands are:

L - list undeliverable mail
carriage return - same as L
H - halt - exit to higher level (back to *)
? - types this message
M - manipulate undeliverable mail -
    Same as L, but after each item is listed, waits for you to choose one of the following options. Except as noted below, the option must be confirmed with a carriage return, or may be aborted with rubout (del).
S - save - does nothing - status of mail is unchanged
carriage return - same as S, but no confirmation required
D - delete message
Q - requeue message
A - address change - allows you to specify a different address for the mail and requeues the mail for the modified addressee.

9.1 USING MAILSTAT IN HERMES

Mailstat lists queued and undeliverable messages in a logged-in or connected directory and allows you to manipulate undeliverable messages.

If undeliverable mail exists, MAILSTAT automatically enters you into a set of subcommands, similar to an editor. The subcommands are SHOW, PRINT, REQUEUE, READDRESS&REQUEUE, DELETE, and NO-ACTION-FOR.

Also available are the general subcommands ABORT, DONE, DESCRIBE, OUTLINE, EXAMPLE, and DOCUMENT.

Example:

```
>mailstat
<HOLG>
Undeliverable mail:
  to              date queued  action
1 Anybody@ISIB      1-Jun-78
2 Cole@ISIC         1-Jun-78
```

```
>>readdress&requeue (undeliverable message number) 2
To: Cole@ISIB
>>show
Undeliverable mail:
  to              date queued  action
1 Anybody@ISIB      1-Jun-78
2 Cole@ISIC         1-Jun-78   READDRESS/REQUEUE to Cole@ISIB
>>delete 1
>>show
Undeliverable mail:
  to              date queued  action
1 Anybody@ISIB      1-Jun-78   DELETE
2 Cole@ISIC         1-Jun-78   READDRESS/REQUEUE to Cole@ISIB

>>No-action-for (undeliverable message numbers) 1
Message 1 marked for deletion.  Cancel
  deletion and leave as undeliverable instead?: yes
>>requeue (undeliverable message numbers) 1
>>show
Undeliverable mail:
  to              date queued  action
1 Anybody@ISIB      1-Jun-78   REQUEUE
2 Cole@ISIC         1-Jun-78   READDRESS/REQUEUE to Cole@ISIB

>>done
>mailer

No acknowledgements for HOLG
Queued mail from HOLG
Anybody@ISIB    , FTP ok No such mailbox at this site.
, RENAMED.
Cole@ISIB, FTP ok, mail allowed, sent ok, deleted.
>mailstat
<HOLG>
Undeliverable mail:
  to              date queued  action
1 Anybody@ISIB      1-Jun-78

>>done
>
```