MASSACHUSETTS INSTITUTE OF TECHNOLOGY

ARTIFICIAL INTELLIGENCE LABORATORY

WORKING PAPER 235 7 September 1982

Getting Started Computing at the AI Lab

by
Christopher C. Stacy

Abstract

This document describes the computing facilities at the M.I.T. Artificial Intelligence Laboratory, and explains how to get started using them. It is intended as an orientation document for newcomers to the lab, and will be updated by the author from time to time.

A.I. Laboratory Working Papers are produced for internal circulation, and may contain information that is, for example, too preliminary or too detailed for formal publication. It is not intended that they should be considered papers to which reference can be made in the literature.

© MASSACHUSETTS INSTITUTE OF TECHNOLOGY 1982

Table of Contents

List of Figures

1. Introduction

This document is written to help you get acquainted with the computational facilities at our laboratory, and to provide enough introductory information to get you started using them.

Although the use of various computer systems is described in this document, it is intended as an introductory guide, not a complete reference manual. If you would like a copy of a reference manual or an AI Laboratory Memo or Working Paper, you should visit the AI Publications office.

Alot of unique jargon is used by people at the AI Lab. As a matter of historical interest, copies of an online file containing a jargon dictionary are kept on the timesharing machines at the lab. You would probably find it useful and entertaining to take a look at this file. A copy exists in the file PS:<GLS>JARGON.TXT on the OZ Tops-20 system, and in the file GLS;JARGON > on the AI10 system. Here is a sample entry from the jargon file:

> **AUTOMAGICALLY** adv. Automatically, but in a way which, for some reason (typically because it is too complicated, or too ugly, or perhaps even too trivial), I don't feel like explaining to you. See **MAGIC**.

Finally, remember that people in the laboratory are generally willing to answer questions and help you get started. Frequently, having a more experienced user spend a little time at the terminal with you can be more helpful than pounds of documentation.

Suggestions for improvements and corrections to this document, or questions and requests for more help, can be sent by electronic mail to the author at the address "CStacy@MC" or "CStacy@OZ".

Numbers in this document are decimal, unless otherwise noted. In the example computer/user interactions in this document, the user's typein is underscored. It is usually obvious where a user pressed RETURN at the end of a line, so no special notation is made to indicate this.

1.1. Lisp Machines

The lab has a number of personal computers called Lisp Machines, which were designed at the AI lab.

The current Lisp Machine processor is a stack oriented order code machine called a CADR. With microcode to implement the Lisp primitives, and a sophisticated Lisp system on top, LispMs comprise the most powerful segment of our computing resources.

The Lisp system software for the Lisp Machine was originally written at the AI lab. It is being continually enhanced and improved by both people at the lab, and by outside companies who manufacture Lisp Machines.

Lisp Machines use special consoles with hi-resolution bit-mapped displays and fancy keyboards, and little boxes known as *mice* which are fun to roll around on tabletops. Some of the Lisp Machines are equipped with high resolution color displays.

Our current LispMs were constructed at the laboratory; there are plans to purchase many additional machines which feature an improved hardware design from outside the lab in the near future.

Some of the LispMs have Floating Point Systems hardware attached to them. These peripheral processors can be used for certain types of number crunching. For example, the FPS boxes are used for doing some of the computations required for speech analysis.

One Lisp Machine (CADR-27) has been dedicated as a central Filecomputer. The Filecomputer has several 300 megabyte disk drives attached to it, and represents a bulk repository for files.

1.2. Timesharing

Currently, the AI lab has two large timesharing computers available for general use by lab members. These two machines computers provide laboratory members with network access, central file storage, document preperation facilities, and software development tools.

MIT-AI10 is a modified PDP-10 KA computer which runs the powerful Incompatible Timesharing System (ITS). Along with the now- demolished PDP-6 computer, this old machine served as the primary computational resource for the first part of our history. Because of its old age, AI10 is not a very reliable computer, and most people do not store important files on it.

MIT-OZ is a PDP-10 KL model B computer (a "PDP-20"). Acquired in June of 1982, this machine is intended to serve along side and eventually replace the services of the KA. OZ runs an enhanced version of the Tops-20 operating system. Tops-20[1] is sometimes called "Twenex."

1.3. Other Computers

Various groups in the laboratory have facilities useful for doing special kinds of computing. Details about these facilities are very dynamic and beyond the scope of this document. To get specific details or to arrange to use this special equipment, contact someone from the appropriate group.

1.3.1. Field Engineering

The Field Engineering project has two VAX computers, an 11/780 called HTVAX, and an 11/750 called HTJR. HTVAX runs the Unix[2] operating system, and HTJR runs DEC's VMS. These machines are available to members of the group for their work in expert systems which debug complex computer hardware. Various faults will be introduced into HTJR's hardware, and programs on HTVAX will diagnose the failures.

1.3.2. Vision and Robotics

The Vision and Robotics groups at the AI lab have developed and obtained some interesting peripherals (usually controlled by a Lisp Machine). Briefly, some of these devices are:

- The PUMA robot arm, manufactured by Unimation. This device is mounted in a special work table whose configuration can be easily rearranged.

- The Frame Grabber, which can capture video images from a camera or videotape.

- The Photoscanner, which can do high resolution (say, 20 microns) digitization of photographic images.

---

[1] Tops-20 and VMS are registered trademarks of Digital Equipment Corporation

[2] Unix is a registered trademark of Bell Labs; thou shalt not take Ma's name in vain!

- The Convolution machine, a special purpose processor for rapidly computing zero crossings in a digitized image.

- The Linear Array Camera, which can take 1000x1000 pixel digital pictures. This is connected to the Convolution box.

1.3.3. Music

Marvin Minsky's AI Music group studies such areas as intelligent music composition tools, automatic transcription of music, and programs which understand improvisational techniques. The group has several Chroma synthesizers which are interfaced to the Lisp Machine.

1.3.4. Altos

The AI lab has a Xerox Alto minicomputer. The Alto is an early 32-bit workstation made by Xerox. The Altos was never made commercially available, but were used at Xerox's Palo Alto Research Center for a series of interesting experiments involving workstations on Xerox's Ethernet. Xerox donated a number of these machines to Universities.

The Alto is the predecesor of such Xerox computers as the Dolphin. Altos are usually programmed in the Mesa language or in Smalltalk. At the AI lab, Altos are primarily used to run the Alto Draw program. An Alto can be used to play various graphics-oriented games, such as: Pinball, Star Trek, Asteroids, and Space Invaders.

Alto documentation, and documentation for the DRAW program can be found in the Alto User's Handbook [Xerox 79].

Files can be transferred over the Ethernet using the Internet Trivial File Transfer Protocol. See the documentation on FTP in [Xerox 79].

1.4. Output Peripherals

There are several output devices for network users in Tech Square. There are two Xerox laser printers which can be used to print medium to high quality output, an electrostatc sheet printer, and a Tektronix screen copier.

- The Dover is a high-speed, high-quality laser printer located in the ninth floor computer room. It is driven by a Xerox Alto computer running a file spooler called Spruce. Files often go through another spooler on the MIT-MC timesharing computer before they get to the Alto. Files sent to the Dover are in PRESS format.

- The Xerox Graphics Printer (XGP) is the venerable ancestor of the Dover. The XGP is controlled by a PDP-11 and is spooled to by a program which runs on MIT-AI10. The fonts available for the XGP are decribed in [XGP].

- A Versatek printer is located in the third floor computer room near the VLSI lounge. This electrostatic printer is useful for producing medium quality wallpaper in bright cheery VLSI patterns. It is controlled by a Lisp Machine, and will soon be connected directly to

the Chaosnet.

1.5. Other Machines

On the ninth floor in building NE43, the AI Lab shares space with the Laboratory for Computer Science (LCS). LCS has several machines which are on the ARPAnet:

- MIT-XX is a Tops-20 machine with a hardware configuration similar to that of MIT-OZ.
- LCS also has two ITS systems running on KA-10 processors: MIT-DM, and MIT-ML (which is used primarily by the Clinical Decision Making group).
- The Macsyma Consortium has a KL-10 processor running ITS which is operated by the Mathlab group at LCS for Consortium members; this machine is called MIT-MC.
- LCS recently acquired many VAX 11/750 computers configured as single-user machines.
- The Ethernet IFS is operated by LCS; this filecomputer system is used primarily by Altos.

1.6. Terminals

There are several kinds of terminals you can use to access many of the computers in the laboratory. Which kind of terminal you use depends on which machine you want access to, and what is convenient for you.

Scattered throughout the lab are conventional terminals such as Concept-100s and Ann Arbos. These are usually attached to a Chaosnet TIP (decribed later when we talk about networks.) These are useful for accessing timesahring machines on the Chaosnet, like AI10 and OZ.

Lisp Machines, on the other hand, require you to use a special console which is wired directly to them.

Consoles connected to the Knight TV system allow you to connect to the AI10.

Most terminals have special shift keys on them called "Control" and "Meta".[3] Typing "control-A" means holding down the CONTROL key while simulataneously pressing the letter "A". One way to refer to the character control-A is to write ↑A. In many programs, control-A actually prints that way when you type it.

Sometimes you will find mention of keys which don't seem to be on your terminal's keyboard. Sometimes the key is simply labeled differently, while in other cases you must type to simulate the missing key.

For example, the RETURN key is almost always labeled RETURN. By contrast, the CALL key does not

[3] Labeled "Control" or "ctrl". On an Ann Arbor Ambassador terminal, "Meta" is labeled "Pause", while the Meta keys are completely unlabeled on a Concept-100

appear on many conventional terminals. When using an ITS computer via a conventional terminal, you can type "control-Z" to simulate a CALL.

The ALTMODE or ALT key is sometimes labeled ESC or ESCAPE on conventional terminals. On terminals which don't have enough graphics capability to print the altmode character, altmode is usually displayed as a dollar-sign ($).

The BACKNEXT character (used by ITS and Emacs) is labeled very differently on different terminals. It usually can be simulated by typing a "control-underscore" (^_).

2. Networks

At the AI lab, we have direct access to two computer networks: the DoD ARPAnet/Internet and the MIT Chaosnet.

A computer connected to a network is called a *host*. The computer networks allow users of hosts to transfer data between machines in a fast and easy way. For Lisp Machine users, access to files on other network hosts is transparent.

Each host on the network has one official host name, and might have additional host nicknames by which they can be referenced. For example: the Lisp Machine known as "MIT-LISPM-1" has the nicknames "CADR-1" and "LM1".

At the time of this writing, AI10 is the AI lab's official network machine, and uses the network host name "MIT-AI". Plans for the near future call for both AI10 and OZ to be put on the ARPAnet. At that time, OZ will assume the name MIT-AI.

Here is a partial list of some computer networks from the system network table on MIT-AI; many of these networks are part of the Internet: ARPAnet, Chaosnet, BBN-RCC, CMU-Ethernet, Cyclades, Datapac, DCN-Comsat, EDN, LCSnet, EPSS, Intelpost, Mitre, Parc-Ethernet-3, Satnet, Srinet, SU-Net, Telenet, Transpac, Tymnet, and Wideband.

2.1. The ARPAnet

The DoD Internet connects hundreds of computers across the world together over in a large "catenet" of smaller packet-swictched networks. The ARPAnet is the original component of the Internet. The ARPAnet is used primarily by the Government agencies other organizations sponsored by the Advanced Research Projects Agency. The ARPAnet is managed by the Defense Communications Agency (DCA).

The ARPAnet provides us with a relatively high speed digital link to many important research centers such as: Stanford University, Carnegie Melon Universty, Berkeley, Yale, SRI, Xerox PARC, and BBN.

Additional information about the ARPAnet can be found in the newtwork resource handbooks and protocol documentation from the Network Information Center (NIC) at SRI International [NIC 82a, NIC 82b]. The NIC maintains an online respository of Request For Comments (RFCs) and Internet Experiment Notes (IENs), and online directories of computers, software, and people on the network.

2.2. The Chaosnet

The Chaosnet is a high-speed packet switching coaxial network designed at the AI lab. The Chaosnet employs its native Chaos protocol, described in [Moon 81a].

The Chaosnet connects many computers around MIT and elsewhere. This includes all the Lisp Machines, the AI Lab's and many of LCS's timesharing machines, the EE Department, the Plasma Fusion group, and more.

Also connected to the Chaosnet are gateways onto MIT Ethernets. Ethernet is another, higher speed coaxial network, developed at Xerox. Some parts of the MIT Ethernets use Chaos software protocols instead of the Ethernet protocols.

There are many hosts on the Chaosnet. A few of them include: AI, Alcator, ARCMac, Bypass, CCC, CIPG, Cog-Sci, CSG, DSPG, Eecs, Franky-Mouse, Fusion, HTVAX, JCF, LNS-VAX, Math, MC, ML, Multics, OZ, PFC-VAX, Plasma, RTS, SIPB, Speech, and XX.

2.3. Services

The computer networks provide three basic services: Remote Access (Telnet), File Transfer (FTP), and electronic Mail.

Many more network protocols exist to do both low level and user-level things. For example, upon booting, Lisp Machines set their time of day clock by polling various other computers on the Chaos network for the time of day. Another example of a useful protocol is the SEND protocol, which allows a user on one machine to send an interactive message to someone on another machine. You can design and implement your own protocols for your programs and conduct interesting experiments if you like.

Here is a brief description of the frequently used common services.

2.3.1. TELNET/SUPDUP

The TELNET and SUPDUP services allow a user on one system to remotely connect through the network to another system and use it as though they were directly connected.

This is mostly useful for briefly visiting a machine other than the one you are using, or for reaching a machine who you cannot directly dial up. It is usually wasteful and inefficient to use a computer exclusively to TELNET or SUPDUP to another computer, when you could directly connect to it instead.

The SUPDUP protocol [Crispin 77] is a highly efficient display telnet protocol. The advantage over the TELNET protocol is that SUPDUP takes advantage of the full capabilities of display terminals, although it also has the ability to run printing terminals. When you use the SUPDUP protocol, you do not need to tell the remote host which you are connecting to what type of terminal you have or what the terminal's capabilities are. The host you are SUPDUPing from handles the actual display support for your terminal.

Additionally, SUPDUP defines a network graphics protocol [Stallman 78] which makes it easy for network hosts to draw pictures along in addition to text.

2.3.2. FTP

There are several sorts of File Transfer Protocols.

These protocols allow the transfer of files between two machines across a network. Some of these protocols include: various Internet File Transfer Protocols, the ITS MLSLV protocol, and the Lisp

Machine Chaosnet FILE and BAND-TRANSFER protocols. Often when someone uses the phrase "FTP", they are referring to the ARPAnet/Internet File Transfer Protocol.

2.4. Mail

Electronic mail plays a very big role at our lab; alot of daily communication with your coworkers both here and around the world is done via the computer. How to send mail and receive depends on which computer you are using, and is described a little later.

People on the machine are referred to by their *network address*.

When you need to send electronic mail, you can use the Inquire database to look up someone by their name or their network address.

When a message arrives for a user, it is placed in a file called a *mailbox*. Each user on a computer should have either a mailbox or an entry in the Inquire database saying on which machine the mailbox is located.

When you send a message to a non-existant network address, the mail system will complain to you to give you a chance to try re-addressing the message and sending it again.

The format of network addresses will be changing slightly soon to accomodate the expanded address space of the fully operational Internet. Under the present scheme, a mailing address usually looks like **Joe@Computer** where **Joe** is the user name of someone at the **Computer** site. Examples: Agre@MIT-AI, DLW@SCRC-TENEX, KDO@SU-AI.

2.4.1. Processing Mail

In order to send and read electronic mail, most people employ a program called a mail reader. Which mail reader you use depends on which computer you read your mail from, andyour personal taste. A powerful mail reader called ZMAIL exists on the Lisp Machine. On the ITS and Tops-20 systems, the most popular mail reader is called BABYL.

2.4.2. Ettiquette

It is considered illegal to use the ARPAnet for anything which is not in direct support of Government business. At the AI lab, we use the network to talk to other researchers about all kinds of things. For example, personal messages to other ARPAnet subscribers (for example, to arrange a get-together or check and say a friendly hello) are generally not considered harmful. This is one of the ways in which we adapt the network environment to our community. It is very clear that without that sort of freedom, the network could not have evolved to its current point of technical and social sophistication.

Sending electronic mail over the ARPAnet for commercial profit or political purposes is both anti-social and illegal. By sending such messages, you can offend many people, and it is possible to get MIT in serious trouble with the Government agencies which manage the ARPAnet.

2.5. Mailing Lists

A *mailing list* is a collection of individual network addresses called by a single name. Mailing lists are useful ways of dissemenating information to specific groups of people.

Through years of experimentation on the ITS computers and its advanced mail software, MIT pioneered the development of mailing lists. Mailing lists have come from the realm of novelty to be recognized as an important research tool used every day.

Some mailing lists are somewhat private, and are intended for communication between a specific group of people. Most mailing lists, however, are public and anyone can join them.

Some mailing lists are quite large -- a few mailing lists have several thousand recipients on them.

Mailing lists for sending announcements are usually named INFO-*something*. These are intended as receive-only lists, and you should not send anything but official announcements to them. Some examples of this are: INFO-EMACS and INFO-LISPM. However, note that this naming convention is not really always adhered to. For example, INFO-MICRO is a general discussion group for microcomputer enthusiasts.

Frequently, the users of a piece of software or hardware will create a mailing list for discussing how the program should evolve, announcing new features to users, and so forth.

Bug reports about system programs can be conveniently sent to the maintaner(s) of a program, if a mailing list is defined for that purpose. This alleviates the problem of knowing who is current ly responsible for each program. By convention, these mailing lists are name "BUG-foo", for example: BUG-EMACS.

A related feature for bug-reporting via the mail system is that if you send a message to a non-existant BUG- mailing list, the message will be delivered to a group of general maintainers. These people will forward the message to the correct maintaner (or perhaps fix the bug themselves).

Mailing lists are defined in files which are part of the mail system for each host. The details of adding an entry to a mailing list are different for each host, and are sometimes a little tricky. However, you do not need to deal with this problem yourself: you can send mail asking to be put on a mailing list.

To get on a mailing list, you should usually not send mail to the list itself. This is especially true of large mailing lists, where the many people on the list are normally uninterested in the comings and goings of the audiance. Except for BUG- mailing lists, most mailing lists of much size have a maintainer who is reponsible for adding and deleting people from the list as they request. By convention, the maintainer of the FOOBAZ list can be reached by sending mail to FOOBAZ-REQUEST.

Sometimes an archive file is set up as a recipient on a mailing list. If a mailing list has an archive, you can delete your copies of the messages (assuming you received them) and refer to the ones in the archive instead. This saves disk space, and preserves the discussions for posterity.

2.5.1. BBoards

Electronic Bboards are similar to electronic mail, but are used for one-way announcements instead of conversations.

BBoards can be conceptually divided into two categories: *system messages* which everyone should see, and other stuff. Certain BBoards are used for just System Messages. Good relations among users rely in part on people not being inundated with junk mail; system messages which go to inappropriate addresses are junk mail.

System Messages are items like listings of Lab seminars, and machine maintenance schedules. Other Stuff includes messages about requests for information, job offers, and housing searches.

To send mail to a BBoard, you simply address your message to a specially named address (just like a mailing list). Here is a description of some useful BBoards. You can mail to these addresses on MIT-OZ.

- BBOARD. All the people on MIT network computers can read these messages.
- *MAC. All of the AI lab and LCS read these system messages.
- *MIT. All the people on network machines at MIT read these system messages.
- *ITS. The users on the four ITS machines see a message sent to *ITS at any single machine.
- *AI. The users on the AI10 and OZ machines see a message sent here.
- *OZ Messages to the MIT-OZ Tops-20 system users.

2.6. Finger/Inquire

The Finger protocol is used to get a list of what people are using a computer on the network.

Inquire is the generic name of a program which manages an informal database of who our computer users are.

The Inquire databases have such information as your user-id, full name, network address (what user name on which computer your electronic mail should go to), your office telephone number, and other useful things. In addition to being used by the mail programs, Inquire provides everyone on the machine with an instant directory of people via the WHOIS and FINGER programs.

There are several Inquire databases, some of which are commonly shared by more than one machine. Which databases you are registered in depends on which computers you use the most; it is not a bad idea to have an entry in each of them.

2.7. TIPs and TACs

In addition to any direct dialup lines which a host might have, there are special hosts know as TIPs and TACs. The name TIP comes from the early days of the ARPAnet, and stands for "Terminal Interface Processor". On the ARPanet Internet, these minicomputers are now called TACs. Both TIPs and TACs serve the same purpose.

A TIP can use the TELNET or SUPDUP protocols to dynamically connect a terminal to some other computer on whatever network the TIP is on. For example, if you are at home and wish to use some ARPAnet host (say, MIT-AI, or Stanford's SU-SCORE) you could call the local ARPAnet TIP and have it connect your terminal to that system; if you needed to use the HTVAX (which is only on the Chaosnet), you could call a Chaosnet TIP and connect to it.

2.7.1. ARPAnet TAC

The ARPAnet TACs are useful for connecting to machines which are directly on the ARPAnet Internet. To use the TAC, obtain the dialup number from your supervisor, and connect your terminal to the phone.

To begin using a 300 baud line, type control-Q.

To begin using a 1200 baud line, type

```
@R RETURN
```

The TAC should respond like:

```
MIT-TAC 424 : #51
```

Commands to the TAC are preceeded by an atsign (@). While you are using the TAC, whenever you type the atsign, the next characters until you hit Return are treated as a command for the TAC. To send an atsign through, you need to type two of them (@@), and the host will echo one more, so it will look like: @@@.

You can change what the TAC's command ("intercept") character is by using the command:

```
@i number
```

Where number is the decimal number of the ascii character to make be the TAC command intercept character. A popular value is 126, which is a tilde (~); another is 36, control-uparrow. To reset the intercept character to an atsign, you can type:

```
@i esc
```

The ARPAnet Internet is still undergoing conversion to the new Internet protocols.[4] At the time of this writing, the TAC is normally in NCP mode when you first connect to it. When a TAC tries to open a connection for you it will mention which protocol it is trying to use. These two commands can be

[4] The older protocol is known as "NCP".

issued if you want to change which protocol the TAC is going to use:

@protocol Tcp

and

@protocol Ncp

To connect to a host on the ARPAnet, use the command:

@open *net:port/imp*

The network number is optional if you want to connect to a host on the ARPAnet. (The ARPAnet is network 10, incidentally).

The TAC will then say "TRYING...", and presently, you will be (hopefully) connected and speaking to the system you requested.

Here are the nunmbers of some common ARPAnet hosts:

MIT-AI	2/6
MIT-MC	3/44
MIT-ML	3/6
MIT-DM	1/6
MIT-XX	0/44
SU-AI	0/11
SU-SCORE	3/11
SRI-KL	1/2

Finally, to disconnect from the host you are using, type the command:

@Close

TIP commands can be abbreviated to their first letter.

2.7.2. Chaosnet TIP

The Chaosnet TIPs are PDP-11-based terminal concentrators connected to the Chaosnet. We currently have installed two Chaos TIPs named NE439A and NE433A, on the ninth and third floors, respectively. Chaos TIPs run an operating system called MINITS which understands a few simple commands.

NE433A is connected to the local terminals (such as Concept 108s and AnnArbors) inside the lab. NE439A is connected to Vadic 1200 baud dial up modems as well as local terminals

Chaos TIP commands begin with a control-backslash. (To send a control-backslash character to the host you are using, you must type the character twice.) After the ↑\ you might type a numeric argument, and then a single letter command. The numeric argument should be either the octal Chaosnet address or a host number to connect to. Here is a summary of the single letter commands:

↑Z	Connect using the SUPDUP protocol.
T	Connect using the TELNET protocol.
F	Connect using the Finger protocol.
K	Disconnect from the currently selected host.
?	List available hosts and their numbers, and these commands.

Some popular host numbers include:

0	MIT-MC
1	MIT-AI10
4	MIT-XX
14	MIT-OZ

Figure 2-1: Chaos TIP commands

3. Interesting Software

This chapter describes some useful programs and languages which are available on various machines at the lab:

3.1. Lisp

Lisp is the language most widely used in the AI lab. There are several dialects available, including:

- MACLISP - This dialect of Lisp was developed at MIT under Project MAC. A descendant of Lisp 1.5, Maclisp is available on the PDP-10 computers, including AI10 and OZ.

- LispM Lisp - This is the dialect of Lisp used by the Lisp Machines. Among its many features is support for an object oriented programming technique called *flavors*. Lisp Machine Lisp is descended from MacLisp.

- SCHEME - Scheme is a dialect of Lisp with lexical scoping. Originally created by Gerry Sussman and Guy Steele, SCHEME is the Lisp which is currently taught in the undergraduate programming course. As part of their research into AI techniques for VLSI design, Sussman's group has created a single chip SCHEME interpreter.

- NIL - NIL is a new lexically scoped Lisp for the VAX computer. It currently runs under VMS.

There are several other Lisp dialects (such as Franz Lisp, Rutgers UCI Lisp, and Yale's "T" language) floating around on various machines.

3.2. Emacs

Emacs is an extensible, customizable, self documenting, display editor. Emacs is very powerful and easy to learn and use, and is the most freqently used editor on all our machines. Once you have used Emacs, you will probably not wish to use another editor.

There are assorted documents concerning Emacs/Zwei, including: the Emacs Manual [Stallman 81a], and [Stallman 81b, Wechsler 82, Ciccarelli 78, Stallman 81c].

The original Emacs was written primarily by Richard Stallman (RMS@AI) under the ITS operating system. With the advent of the Lisp Machines, Dan Weinreb wrote an editing system called ZWEI, and implemented the Zmacs editor in it.

The word *EMACS* probably originally stood for "Editing Macros". Another popular theory says that it is a recusive ancronym meaning, "Emacs Makes All Computing Simple".

ZWEI stands for "Zwei Was Eine Initially". Eine was an early version of the LispM editor; "Eine Is Not Emacs".

There is also an Ersatz Emacs editor on the VAX timesharing machine.[5]

Emacs and Zwei can serve as a base for advanced, friendly programs, especially for applications involving text processing if some sort. Here are a few popular programs which are based on Emacs:

1. INFO (called XINFO on Tops-20) allows you to examine tree structured online documentation files. There is alot of useful information in INFO; you should check it out on the timesharing machine you use.

2. DIRED is a directory editor. It allows you to easily examine and modify your directory (changing file attributes, deleting files, etc.) from inside the editor.

3. BABYL is a program for reading and sending electronic mail. BABYL will reformat the memo headers of incoming messages to look nicer, can keep keyword tags on messages, and has a powerful Survey Mode for scanning through large mailboxes.

3.2.1. TECO

The original (PDP-10) Emacs is implemented in ITS TECO, an amazingly powerful and amazinlgy obscure language which most closely resembles line noise.

Here is some sample TECO from an Emacs init file; this code redefines what functions certain key-commands will invoke when typed in this user's editing environment:

```
q↑R↑?(q.↑R↑?u↑R↑?)u.↑R↑?    !* Switch RUBOUT and C-RUBOUT !
q.↑RWu:.x(↑↑↑K)              !* Make C-X C-K do like C-W       !
```

Luckily, you can do many customizations without learning TECO, and you can usually find a TECO wizard around to help you do more fancy things and write new editing commands.

3.2.2. ZWEI

With the larger address space available in Lisp Machines, the ZWEI editor (Zmacs) was able to be written in Lisp.

Here is some Lisp code from a ZMACS init file which does the same thing as the TECO code did in the PDP-10 Emacs above:

```
zwei:
(login-eval
 (set-comtab-return-undo
     *standard-control-x-comtab*
     '(#\control-K com-kill-region)))
zwei:
(login-eval
 (set-comtab-return-undo
```

[5] Some former MIT students developed and marketed an Ersatz Emacs editor for microcomputers called MINCE; "Mince Is Not Complete Emacs".

```
*standard-comtab*
'(#\control-rubout com-rubout
  #\rubout com-tab-hacking-rubout)))
```

3.3. SRCCOM

The SRCCOM source comparator on ITS and Twenex is a handy tool for doing automatic comparisons of source files to find the differences between them. The Lisp Machine has a SRCCOM program as part of the DIRED subsystem in Zmacs.

3.4. ATSIGN

The ATSIGN program generates program listings with nice cross references and symbol tables.for several languages, including Lisp. ATSIGN will handle multiple fonts, mutiple files, and makes nice crossreference charts.

3.5. Text Justifiers

There are three text justifiers which are popularly used by people at the AI lab.

1. Tex is a very powerful typestting system written by Don Knuth at Stanford. It is especially good for doing complex mathematical typsetting.

2. R is a text justifying system which is designed to be very flexible. Such features as multiple environments and macros are provided. R is available under ITS, Tops-20, Unix, and VMS. Documentation for R is available in INFO.

3. The SCRIBE document production system was developed at the Computer Science Department of Carnegie-Mellon University. *Getting Started Computing at the AI Lab* was prepared using SCRIBE, and camera-ready copy was printed on the Dover printer. SCRIBE is available under ITS, Tops-20, and on the VAX.

3.6. SPELL

The SPELL program on ITS (called ISPELL on Tops-20) will check the spelling of the words in a file. This is an amazingly handy tool for preparing papers. More information about SPELL can be found through INFO.

3.7. Macsyma

Macsyma is an expert program for doing powerful symbolic and numerical mathmatical manipulations.

Macsyma was originally written in Maclisp and has been ported to the Lisp Machine. At the time of this writing, Macsyma is being installed on the MIT-OZ computer; it is also available on the Macsyma Consortium's ITS computer, and on the MIT-Multics system.

3.8. Electronics Design Tools

- DAEDELUS is a program for assisting engineers with circiut design at the chip level. DAEDELUS is a program which runs on the Lisp Machine. The Design Procedure Language (DPL) is layout language often used in conjunction with Daedelus.

- DRAW is a program for assisting engineers with circiut design and layout at the pc-board level. DRAW runs on MIT-AI10 with a Knight TV.

3.9. FED

FED is the Lisp Machine Font Editor. With FED, you can modify and design the character fonts which the Lisp Machine display uses.

4. Using Lisp Machines

This chapter will briefly introduce you to the Lisp Machine, but will tell you very little about actually using one. Lisp Machine documentation has been published elsewhere, and much more documentation in preperation. For more detailed documentation, good places to begin include *The Lisp Machine Manual* [Weinreb 81], and *Operating the Lisp Machine* [Moon 81b].

Lisp Machines are very powerful single-user computers which are well suited for use in AI research. Lisp Machines, called "LispMs" for short, are general purpose computers which implement Lisp in microcode. Naturally, all the system and user software is also written in Lisp. Lisp Machines feature dynamic paging over a large address space, and a very sophistiated programming environment. Each Lisp Machine is connected to the Chaosnet, which was originally designed to allow them to communicate.

The idea behind the Lisp Machine is that one big Lisp environment is made available to the user. Inside this environment are various system programs such as the compiler and editor which are useful for writing Lisp programs. This environment can provide the same unlimited flexability as the Lisp language itself.

Although most LispMs have a 300 megabyte disk attached to them, users do not generally store their files there. The disk is used primarily for storing copies of the of the Lisp Machine system (world load and microcode files), and for dynamic paging space.

The Lisp Machine has a *pathname system* which makes it possible to reference file systems on many other computers. Most users store their files on a timesharing machine (such as OZ) or on the Lisp Machine Filecomputer.

Each Lisp Machine is said to be *associated* with a particular timesharing machine. This association determines such things as where the machine will defaultly assume that the user's files are.

4.1. Window System

The Lisp Machine has an advanced *window system*. A window is a specific portion of the screen where a program can interact (type out or read input) with the user. The window system makes it possible for the user to make many windows of various types and helps to manage them on the screen. Different programs can run in different windows, or a program could use multiple windows, depending on its design. Often, a window will take up the entire screen.

A useful device for interacting with windows is the mouse. The mouse is a small hand-sized box which can be rolled on the tabletop to control a pointing cursor on the screen. On the mouse are three buttons on it which can be clicked; what the clicks mean depends on the program reading them.

4.2. booting

Resetting a Lisp Machine is called *booting* it. The most frequent kind of reset is a *cold boot*, which completely reinitializes the machine and starts it up fresh. When you are done using a Lisp Machine, it is best to cold boot it for the next person. When a machine has been cold booted, the words COLD BOOTED appear on the screen in the lower right hand corner.

When the Lisp Machine starts up, it leaves you in a window called the Initial Lisp Listener. This is a Lisp interpreter which runs in a window filling the entire screen.

4.3. Logging in

Lisp Machines are currently accessable only from the special console connected to each machine. The Lisp Machines themselves are located in a computer room; many of them are located in a corral on the ninth floor in Tech Square. LispM consoles are located in offices throughout the lab.

A LispM console consists of a high resolution video monitor, and a keyboard. You will probably notice that the keyboard has alot of keys on it. One useful key is labeled HELP; it can be pressed to get helpful information. Also, attached to the console is a mouse. Mice are purported to work best when rolled around on the surface of a Lisp Machine Manual.

Before you can login to a Lisp Machine, you need to locate one.

It would be tiring to walk around to each LispM console to see if someone was logged into it. A quicker way of finding a machine is to either look on the Free Display of a Knight TV, or ask someone on the nearest Lisp Machine to generate a list of free machines. The LispM command to get a list of which Lisp Machines are in use and which are free is TERMINAL 1 F. That is, press the TERMINAL ey, followed by the number 1 and the letter F.

Once you have found a Lisp Machine, be sure that it is really not in use. A Lisp Machine which is not in use has mostly likely been completely reset cold booting. On the screen will be a herald telling what the name or number of the Lisp Machine is, some information about what version of Lisp Machine software it is running, and which machine the LispM is associated with.

The user interface to the Lisp Machine does not include a command interpreter. Instead, you type Lisp expressions for evaluation and for side effects.

To login, use the ***login*** form, which requires a single argument: your user name. If you do not wish to use the associatd timesharing machine, yous should include a second argument: the name of the system where your files can be found.

```
(login 'cstacy 'mc)
```

When you are done using the machine, the usual practice is to cold boot it. This is done by holding down both the left and right CONTROL and META keys while striking the RUBOUT key.

4.4. Editing and Compiling

To begin editing your programs, you need to enter the editor. This is done with the ***edit*** form, which requires no arguments. ***Edit*** will put you into Zmacs, an editor which is very much like Emacs.

```
(edit)
```

If you know how to use Emacs, you should have no trouble: the simple commands all work the same. When you want to exit back to Lisp, use the control-Z (↑Z) command.

One way to compile some Lisp code into the environment is to mark off a region in the editor, and use the M-X Compile Region command. There are many, many more commands for interacting with Lisp from the editor.

4.5. Communication

The Lisp Machine is on the Chaosnet, so LispM users can send messages to each other and to people on timesharing machines. To send a message to someone, use the ***qsend*** form. Qsend requires as its argument a string containing the name@site of someone to send the message to. The Qsend program prompts you for the message text, and then transmits it to the foreign user.

```
(qsend "zvona@CADR-22")
```

4.6. Additional Documentation

Some other useful documents about the Lisp Machine include:

- *Lisp Machine Summary*, Symbolics, by Janet Walker, May 1982.
- *System Release Notes*, which are available online on MIT-OZ.
- *Introduction to Using the Window System*, AI Lab Working Paper 210, by Daniel Weinreb and David A. Moon, May, 1981.
- *Lisp Machine Choice Facilities*, AI Lab Working Paper 208A, by David A. Moon, Revised June, 1981.
- *Scroll Windows*, Symbolics, by Bernie Greenberg, August, 1981.
- *Chaosnet FILE Protocol*, Symbolics, by Eliot Moss and Daniel Weinreb, September, 1981.

5. The Knight TV System

For many years, the primary way to access AI was from the Knight TV terminal system, which was designed by Tom Knight (TK@MIT-AI).

The Knight TVs are bit-mapped frame buffer displays controlled by a small PDP-11. The TV-11 is connected to the PDP-10 via a special PDP10-to-PDP11 interface which allows the PDP-11's memory to be mapped into the PDP-10. A Knight TV console consists of a CRT display, and a keyboard similar to those on the SAIL TV system used at Stanford.

The TV-11 is connected to a Tektronix electrostatic copier in the eight floor playroom. It is possible to copy your screen image to this printer. The TV-11 is also connected to the elevators in NE43, and the ninth-floor door opener. There are commands to summon the elevator and open the door.

A program on the PDP-10 called the Name Dragon prepares and updates a screen of information called the Free display. When a TV is not in use, its screen displays the Free display.

This display will show you who is logged in to what terminal, what programs they are running, and some information about the system (such as the date, the system version number, how much core is available, and how many jobs are paging).

There are more physical consoles than there are memory buffers. The Free display tells you how many are functional and how many are in use. The Free display also lists which Lisp Machines are in use, and the location of free ones.

Other things displayed include birthday notices, an amusing picture, and the time of day the Free display was updated.

When a TV is being used, the bottom line on the screen contains an interesting status report called a *wholine*. The wholine shows the user name, job name, date and time, status (running, paging, etc), the system "fair share" and other interesting data.

```
7/27/82 4:13:28  3 CSTACY E      CSTACY ECHOIN  1/32% 0:12:07.4  63/94K
```

The ESCAPE key (different from ALTMODE) is used to send special commands to the PDP-11 controlling the TV display. Commands are single letters, which may be preceeded by numeric arguments. The commands are not echoed.

```
ITS 1279 7/27/82 3:38:05 TVS: 8/11 USERS:14 0/23 42% SB=0 PG=0 CORE:0/365 43

-User-     --Full name--          Jobnam Idle TTY -Console location-
XGP     O Xerox Graphics Printer XGPSPL    26 T24 Datapoint Near XGP (9th)
DCB       Dan Brotsky            TEX         T43 NE438A: 814 Brotsky
KLH     + Ken Harrenstien        MIDAS       T44 Net site SRI-NIC
KLOTZ     Leigh L. Klotz         TECO        T55 926 Sealy x6765
KEW       Karen E. Wieckert      OZ         .T56 819 Davis x5879
SHIM      Shimon Ullman          HACTRN  1:53.T57 822 Ullman x5033
MINSKY    Marvin Minsky          OS          T60 750 Minsky x7807
GJS       Gerald J. Sussman      SCHEME      T61 356 Sussman x6874
RICH      Chuck Rich             HACTRN      T62 342 Rich x7877
PHW       Patrick H. Winston     ANALOG   12 T63 817 Winston x6218
RMS       Richard M. Stallman    EMACS       T64 914 Stallman x2076
POGGIO    Tomaso Poggio          LISP        T66 822 Poggio x5226
CARL      Carl Hewitt            HACTRN  *:**.T67 813 Hewitt x5873
PGS       Patrick Sobalvarro     CADR25          902 Robot room x6765
KENT      Kent Stevens           CADR24          903 Vision x6765
KMP       Kent M. Pitman         CADR22          339 Sorcerer's Apprentice
DPH       Daniel Huttenlocher    CADR21          722 FPS Express x6765
GREGOR B  Gregor Kiczales        CADR20   2      706 Educational Computing
ECC     H Eugene C. Ciccarelli   CADR18          907 CADR-1's room  x6765
LEVITT    David Levitt           CADR-9          721 Music Hacker's Hangout
PGS       Patrick G. Sobalvarro  CADR-7          936 LispM Factory x6703
CSTACY    Christopher C. Stacy   CADR-6          726 Spacy x7710
HENRY     Henry Lieberman        CADR-4   14     912 Lounge Lizard Lispmacho
NIS       Keith Nishihara        CADR-3          903 Vison x6765
AGRE      Phil Agre              CADR-2          751 VLSI Lounge x7837
DANIEL    Daniel Weise           CADR-1          907 [Son of CONS]  x6765

System up time = 3:15:48   Last updated at  3:49:26    XGP: Idle
Free Lisp Machines:  CADR-8 747, CADR19 301,  CADR23 822

Happy Birthday to SUNNY and MOBIUS
```

Figure 5-1: The Name Dragon's Free Display

Knight TV commands

Command	Description
F	Show the Name Dragon display (Finger). This command will switch your TV so that instead of looking at your own display buffer, it sees the Free display.
S	Switch frame buffer (Spy). This command will switch your TV to your own display buffer, which is useful if you were looking at someone else's. If you type a number before the "S", your TV will be switched so that you see that TV buffer number.
C	Complement white-on-black (actually, it's green-on-black) mode.
D	Buzz open the ninth floor door. Your screen flashes when the door has been buzzed.
E	Summons the elevator to your floor. Your screen flashes when the elevator has been called.
Q	Copies your screen to the 8th floor Tektronix copier. Your screen flashes when the TV has grabbed the image for the copier.
I	Clear your scroll register. You should never need to do this.
A	Select an audio channel to listen to. This usually doesn't work anymore.
L	Clear screen without telling the PDP-10.
W	Control the wholine.
U	Control the wholine.

Figure 5-2: Knight TV commands

6. Using ITS

This chapter will tell you hw to get started using the MIT-AI10 timesharing system.

There are three other ITS computers in the world: MIT-MC (a KL machine) owned by the Macsyma Consortium; MIT-ML, the Mathlab computer at the Laboratory for Computer Science; and MIT-DM, the LCS machine where the game of ZORK was created.

MIT-AI10 is a modified PDP-10 model KA processor with Stardust Memories[6] , a homebrew pager, PDP10/PDP11 computer interfaces, a 7 track tape drive, and eight Calcomp disk drives. MIT-AI10 runs the Incompatible Timesharing System (ITS), developed and maintained at the MIT AI lab. At the time of this writing, MIT-AI10 is the AI Lab's ARPAnet machine and has the official host name MIT-AI. The system is also on the MIT Chaosnet.

6.1. Logging in

In order to login to ITS, a user needs an identifier called a uname. Unames are unique six character names which do not end in a digit. People who need to access ITS from outside the building need an account with a password.

To log in to AI10, get to a terminal connected to the machine (perhaps a Knight TV console, or a Chaosnet TIP terminal) and press the CALL key. The first thing you probably want to do after hitting CALL is log in. Logging in identifies you to the system: it allows your home directory to be set, and runs an login-init file you may have.

If you are logging in from inside the building or over a network connection from another AI lab computer, you will not be asked for your password. When connecting to the machine from outside the building (over a dialup or network) you will need to type a password to prove who you are.

To log into ITS, type the LOGIN command:

```
AI ITS.1268. DDT.1388.
TTY 62
8 Lusers,  Fair share = 69%
*:login uname
  [OK]
*
```

All programs which run on ITS run in *jobs*. Jobs have names up to six characters long. When you first press the CALL key and get the attention of ITS, ITS creates a job called HACTRN for you and starts up a copy of DDT in it.

When started, DDT says hello with its version number, the ITS version number, your tty line number, and how many users are on the machine and how loaded it is. Finally, DDT will print its prompt, an asterisk (*), to let you know it is ready for commands.

---

[6] Custom MOS memory designed by Howard Canon (HIC@MIT-MC). The Stardust is often referred to as the HICMEM.

6.1.1. Logging Out

When you are done using ITS, you should type the LOGOUT command:

```
*:logout
AI  ITS 1268 Console 37 Free.  09:32:21
```

The logout command will destroy any jobs you were running, including your HACTRN, at which point you will be logged out.

If you are on a dialup line or a net port and are disconnected without first logging out, your HACTRN and other jobs will become "detached" instead of destroyed. When you next login again, DDT will offer to "re-attach" them for you.

If you answer no to this offer, you can later get your detached job back by typing the command:

```
*:reattach uname/K
```

It is important to not leave extraneous job sessions lying around on the system, since they consume job slots which could be used by other people.

6.2. Commands

The usual top level user interface on ITS is DDT, the Dynamic Debugging Tool. Originally designed for debugging assmebly langauge programs, DDT has a pretty obscure command language.

There are two sorts of DDT commands: "short form" and "long form".

Most beginners use the long form of commands, because they are easier to remember. All long form commands are begun with a colon (:) and end with a RETURN. Long form commands are single words, such as LOGIN, COPY, and KILL.

The way to run programs is similar to the way to run commands. To run a program, type a colon, the program's name, and press RETURN. Example (running the "WHAT" program):

```
*:what

You tell me

:KILL
*
```

When commands (and most programs) run to completion, they indicate their termination by typing out ":KILL".

A short form exists for many commands. This is usually a short, obscure sequence of control characters and the (ALTMODE) character. An example of a short form command is:$^X. (ALTMODE[7] ALTMODE control-X period) and ^R (control-R). While these are pretty cryptic, many users find that the

[7] We use the dollar sign to represent the ALTMODE character.

short forms are convenient to type.

There are many short form commands used for debugging assembly code; you will probably never need to type them. We mention such commands here to ease your confusion because you are likely to someday inadvertantly type one by accident. For example, if you type a slash (/) you will usually make DDT type the contents of some memory location.

6.2.1. Control Characters

DDT understands several control characters for editing your input and interrupting the system.

The CALL key (or ↑Z) is used at almost any time to get the attention of ITS, interrupt the running job, and return you to DDT.

Control-G is an interrupt similar to CALL, but it is more drastic and is not guaranteed to leave things in a consistant state.

Control-S is used to stop the output of a DDT command.

The RUBOUT key deletes the last character you typed.

Control-D deletes the current line being typed.

Control-L clears and refreshes the screen, retyping the current line.

6.2.2. --More--

When you are using ITS an a display terminal, the system will pause at the end of each page of output and wait for you to tell it to continue. This ensures that you have a chance to read the screem before it is rewritten. When it pauses, it will type

```
--More--
```

and wait for you to press either SPACE to continue, or RUBOUT to flush the output. If you type a character other than RUBOUT or SPACE, it will be treated as a RUBOUT.

6.3. INFO

Extensive online documentation about ITS and the programs on it can be found with the INFO program. INFO is self documenting and includes a built-in tutorial.

If you are really having trouble, you can request a system hacker to come and help you by typing the LUSER command.

```
*:luser
Help has been sent for, please wait.
:KILL
*
```

LUSER will tell you if it cannot find anyone to ask for help.

6.3.1. System Load

The load on the machine is measured as the "fair share". Considering the number of jobs in the system and the amount of system overhead, the fair share is the percentage of the machine's computrons that each of the jobs which wants to run should get. The larger the fair share's value, the less loaded the machine is.

You can see what the current fair share is by looking on your wholine, or by entering the SSTATUS command.

The LOADP program can also be used to print some system load information. LOADP tells how many users are on the machine, how idle they seem to be, and how many network ports (STYs) are free.

6.4. System Security

ITS does not have any "security" to prevent users from doing things. As a result, it is incumbant on ITS users to take care not to damage other people's files. All users may freely access any file they choose, and can run programs which might be considered privileged on some systems.

6.5. Jobs

As mentioned before, a job is the entity in which programs run A job in the system is uniquely identified by its two names: its *uname*, which identifies its owner, and its *jname*. When dealing with DDT to manipulate your own jobs, the uname (which is the same as your uname) is usually not mentioned.

The thing that DDT is good at is manipulating jobs. A user can have several jobs extant concurrently, arranged in a tree which with a HACTRN (containing DDT) at the apex. More than one job may be running at a time, but only one job has control of the console to type out and read typein from.

With the understanding that DDT is the name of the program, and HACTRN is the name of the job in which it runs, we will use the terms interchangably hereafter to refer to your toplevel job.

Initially, your DDT has control of the console. When you start up another job, DDT passes the console to the new job (called the *inferior* job, because of its place in the tree relative to DDT).

Here is a representation of a job tree with three inferior jobs named LISP, X, and EMACS.

Control of the console can return to DDT if the program decides to interrupt its superior (HACTRN), or if you press the CALL key to force the interrupt yourself.

To resume a job which has been stopped with the CALL key, type the CONTINUE command. The short form of continue is:$P (ALTMODE P).

To start a job, simply type its name as though it were a long form command. For example, to start up a LISP interpreter, type:

```
*:lisp
LISP 2122
Alloc?_
*
```

If you typed the above command, you would be left inside a Lisp interpreter, ready to start typing in Lisp s-expressions. You could temporarily exit the LISP job and get back to DDT by typing the CALL key.

Most users have more than one job in their job tree: typically there is an EMACS and some other job, perhaps a Lisp interpreter.

DDT thinks of your inferior jobs as belonging in a ring where one of the jobs is the currently selected job. You can give commands to switch between jobs and make which ever one you choose be the current job.

To list all your jobs and see which one is the current job, use the LISTJ command. An asterisk is printed next to the current job.

```
*:listj
  *LISP  P - 2
   EMACS P - 33
*
```

In the above example, to the right of the jname is a single letter indicating the state of the job, followed by the job's internal number in the system. The "P" state means "proceedable"; you can use the CONTINUE command on this job to resume it. Other common states include: "R" for "running", the job is running but doesnt have control of the console; and "W" for "waiting", meaning that the job was running but is stopped because it needs back control of the console to do further work.

To make a job named FOO be the current job, type the JOB command:

```
*:job foo
*
```

Instead of explicitly saying what job to make current, you can rotate through the ring of jobs by omitting the argument to the JOB command. In this case, DDT will print out "FOO$J" where FOO is the jname of the current job, to let you know which one you picked. The short form of the JOB command is $J" (ALTMODE j).

Once you have selected a job and made it be the current job, here are some things you can do with it:

- The CONTINUE command will proceed the job and give it control of the console.

- The PROCEED command will proceed the job without control of the console. If the job is named FOO and it wants to type something on your console, it will make DDT say, "JOB

FOO WANTS THE TTY". Making FOO the current job and typing the CONTINUE command to DDT at that point would then give FOO the TTY. The short form of the PROCEED command is control-p (↑P).

- The KILL command gets rid of the current job.

When you attempt to create a job with the same name as an existing job of yours, DDT will offer to clobber the existing job and start a fresh one. You should type a SPACE to say "yes", and clobber the job, or press RUBOUT to not kill the existing job. There are customization varibles to change the behaviour of this feature.

The MASSACRE command KILLs all of the jobs in your HACTRN.

6.6. Files

On ITS, users store data in *files*. To access a file, you must know its name. File names consist of several component fields. The syntax of a file name looks like:

Device:Directory;FN1 FN2

Each of the fields can be up to six characters long.

The Device specifies which logical device the file lives on; this is usually DSK, the primary disk. You can also specify the two letter name of another ITS (ML, MC, DM), or the Filecomputer (FC).

The FN1 and FN2 are the names of the file; most people use the FN2 to indicate the file type (such as LISP) or version (such as 69).

The FN2 ">" and "<" have special significance to ITS; they refer to the newest and oldest version number of a file, respectively.

Each user on ITS has a home directory; many users have a directory whose name is the same as their uname. If your home directory is not the same as your uname, DDT tells you what it is when you log in. The HSNAME command will also print this for you:

```
*:hsname chris
CHRIS==>CSTACY
```

When you use file names in DDT, they are sticky. This means that file manipulating commands will tend to properly default the various fields in the file name to their value the last time you issued a command. Because of this, you do not need to type the entire file spec to refer to the same file or similarly named files in successive commands.

If you are typing a filespec and want to see what the resultant file name would be (with the defaults merged in), you can type an ALTMODE to get DDT to print it out.

Files are stored in directories. To list all the files in a directory, use the LISTF command. You can specify to LISTF which directory to list, or let it default.

```
*:listf mycrft

AI MYCRFT
FREE BLOCKS  #3=400 #1=130  #13=1200 #2=200  #6=212  #15=2000
  1   MYCRFT LOGIN  1       6/16/82   06:02:20
  1   MYCRFT EMACS  2       6/16/82   07:32:20
  1   MYCRFT MAIL   18      6/17/82   05:16:01
  13  MISPU  14     20      7/28/81   09:12:40
  3   MISPU  20     22+103  7/28/81   02:07:08
  6   FROB   BAZ    1       1/12/82   04:22:34
  1   THESIS TEX    45      6/16/82   12:02:01
  1   THESIS PRESS  102     6/16/82   14:37:00
  15  URSLA  LEGUIN 1       5/16/81   01:50:23
*
```

Figure 6-1: ITS directory listing

The above sample directory listing (of a mostly empty directory) illustrates the LISTF command. Listed is: the name of the directory, the number of free disk blocks on each pack, the pack which each file is on, the name of the file, its length in blocks, and its creation date. A disk block on ITS is around 4K characters long.

6.6.1. Useful File Manipulation Commands

The short form of LISTF is ↑F (control-F). To list the KMP directory, type the command KMP↑F.

A variant on LISTF is useful for listing related files in a directory. To list only those files which have an FN1 of "JUNK", you can type the command: JUNK$$↑F (junk ALTMODE ALTMODE control-f).

To print the contents of a file on your terminal, you can use the PRINT command. Example:

```
*:print foo;bar baz
```

The short form of PRINT is ↑R (control-R); to print the file named FOO;BAR BAZ, type ↑R FOO;BAR BAZ and press the RETURN key.

To delete a file from the system, you can use the DELETE command. Example:

```
*:delete foo;bar baz
```

The short form of DELETE is ↑O (control-o), which works like ↑R does for PRINT. Because you could easliy delete a file accidently, this short form does not work unless you set a customization variable (explained later) which enables the command.

To rename a file, you can use the RENAME command. It takes two file names seperated by commas: an initial file name and a new file name. Example:

```
*:rename foo;bar baz,foo;new name
```

A related command is the MOVE command. The MOVE command will rename a file between directories or across devices. This is sort of like renaming, except that it actually copies the file and then deletes the original.

To copy a file, you can use the COPY command. Example:

```
*:copy foo;bar baz,fred;bar baz
```

Both MOVE and COPY will copy across directories or devices, but RENAME will only rename a file in the same directory.

6.6.2. Hardcopying files

The DOVER program and the XGP program can be used to print files on the repsective hardcopy peripherals. Each of these prgrams expects a file name as an argument.

```
*:dover james;sprite >
[Attempting Chaosnet transmission]

:KILL
*
```

6.6.3. FIND

You can find files whose names match certain patterns by using the FIND program. FIND is documented in INFO. Example:

```
*:find sys***;p*****
AI:SYS;
  2   PDP10   DDT     3          9/2/71 18:46:00    (6/27/82)
  5   PURQIO  2122   67 +307     2/19/82 04:23:48   (8/23/82) JONL
AI:SYSBIN;
  4   PEEK    BIN     9 +297     8/9/82 03:02:07    (8/23/82) PGS
SYSENG;
  3   PEEK    474    22 +615     10/30/81 00:58:18  (4/13/82) MOON

:KILL
*
```

6.7. Terminal Support

ITS is a designed for use with display terminals. If you are using a hardwired terminal, a Knight TV, or are using the SUPDUP protocol across a network, ITS will usually already know your terminal's characteristics. However, coming from a dialup line or a TELNET connection (such as an ARPAnet TAC), ITS will need to be informed about your terminal.

ITS provides support for a variety of terminals through two programs: TCTYP and CRTSTY. The TCTYP command is for terminals which ITS knows about natively, while CRTSTY is used for more complicated support and display optimization.

TCTYP will accept a simple syntax which specifies the type of terminal you have, such as a Concept-100:

```
*:tctyp c100
:KILL
*
```

TCTYP also understands more complicated specifications. A very simple example of this is telling the system how many lines your screen has. On some terminals, for instance an Ann Arbor Ambassador, giving this specification will cause ITS to transmit the codes necessary to actually set the size you want:

```
*:tctyp aaa height 48
:KILL
*
```

Giving the command

```
:tctyp ?
```

will cause TCTYP to list the terminals and options which it understands.

The CRTSTY program works by logging you in on an additional job and intercepting and translating your input/output stream to the system. This means that it uses up an additional job slot and an additional virtual port (STY) on the machine. There are a limited number of STYs on the machine.

Entering the command

```
:crtsty ?
```

will make CRTSTY print out what options it supports.

6.7.1. Wholines

If you are not using a Knight TV and still desire a wholine on your screen, there are some programs to provide one. The WHOLIN program works on most terminals, while the H19WHO puts a wholine on the 25th line of an H19 terminal. The VTTIME program will put a time of day clock in the lower right corner of your screen. Finally, an argument of "wholine" to the CRTSTY program will provide a CRTSTY user with a wholine.

6.8. Customization

Customization involves some slight magic. If you find this section confuing, don't hesitate to get a more experienced user to help you.

There are a number of variables in your DDT which control the exact behaviour of some of your commands. You can set these by telling DDT to make itself the current job with the SELF command. To poke the desired value into the variable, type the name of the variable, a slash, the value, and a carriage return.

One of the interesting variables is called DELWARN. A non zero value for DELWARN will enable the short form delete command. By setting it to the value two (2), you will also make DDT print a brief explanation of what it will do if you press return.

In this example, we will set the DELWARN variable, type a short form PRINT command, and then cancel the PRINT command by typing control-D instead of carriage return.

```
*:self
#!
*delwarn/ 1 2
:kill
*↑R PRINT FILE
DSK:CSTACY;STUFF > ↑D XXX?
```

Another useful variable to set is the MORWARN variable. Setting this variable to zero (0) will cause your --More-- message to be terse.

By setting the BYERUN variable to -1, you can get a cute message or saying when you use the LOGOUT command.

There are many customization variables in DDT. Full documentation on them can be found in INFO.

6.8.1. Login inits

You will probably want some of the customization variables set for each of your sessions, and you will probably want the same ones set each time. You can do this by writing a file in your home directory which has an FN1 the same as your uname, and an FN2 named "LOGIN". This is your *login init file*, which is a series of DDT commands to be executed each time you log in.

In order to not see the DDT commands as they are executed from the file, you can enclose the commands in the file inside ↑W (off) and ↑V (on) characters, which control command echoing in this mode.

The following sample init file does these things: Unless you are on a SUPDUP, CRTSTY, or hardwired connection, it asks you if you are on a Concept-100 or VT-100 terminal and sets the terminal type correctly; sets some DDT variables; notifies you of incoming personal and system mail, and exits.

```
↑W
:ddtsym ttytyp/
:if e $q&<%tysty+%tydil>
  $(:jump mail
  $)
:ddtsym tctyp/
:if e $q-%tnsfw>
  $(:jump mail
  $)
:$↑VOn a Concept?↑W$
:if more 0
    $(:$↑VYes↑W$
      :tctyp c100
      :jump hack
    $)
:$↑VNo↑W$
    $)
:$↑VVT100?↑W$
:if more 0
    $(:$↑VYes↑W$
      :crtsty vt100
      :jump hack
     $)
:$↑VNo↑W$
:tag hack
:self
delwarn/ 2
morwarn/ 0
sndflg/ -1
byerun/ -1
:kill
:gmsgs * /D
:vk↑V
```

Figure 6-2: Sample DDT init

6.9. Inquire

To make an Inquire entry on ITS, run the INQUIR program.

The WHOIS command will print on your terminal all the information associated with a user in the ITS Inquire database. A single Inquire database is shared between all the ITS machines; if you are known on one ITS, you are known on all of them. Example:

```
*:whois kwh


KWH A  Ken Haase                    Last logout 6/23/82  No plan.
 (Ken) [OZ]  Hacking amateur systems for Minsky
 Birthday December 3; NE43-352; 3-1728; Home phone 491-0012
 Senior House, 4 Ames St., Cambridge, MA 01239
 Smile.
```

Numerous commands and programs exist to see who is logged in. The most common ones are WHOJ and FINGER. Example:

```
*:whoj
T24 XGP     XGPSPL
T43 DCB     TEX
T53 CSTACY  EMACS
T56 KEW     R
T67 CARL    HACTRN
:KILL
*:finger

-User-     --Full name--          Jobnam Idle TTY -Console location-
XGP     O Xerox Graphics Printer XGPSPL    26 T24 Datapoint Near XGP
DCB       Dan Brotsky            TEX         T43 NE438A: 814 Brotsk
CSTACY    Christopher C. Stacy   EMACS       T53 914 Greenblatt x67
KEW       Karen E. Wieckert      R          .T56 819 Davis x5879
CARL      Carl Hewitt            HACTRN  *:**.T67 813 Hewitt x5873
PGS       Patrick Sobalvarro     CADR25          902 Robot room x67
KMP       Kent M. Pitman         CADR22          339 Sorcerer's App
DPH       Daniel Huttenlocher    CADR21          722 FPS Express x6
GREGOR B  Gregor Kiczales        CADR20    2     706 Educational Co
ECC     H Eugene C. Ciccarelli   CADR18          907 CADR-1's room
LEVITT    David Levitt           CADR-9          721 Music Hacker's
PGS       Patrick G. Sobalvarro  CADR-7          936 LispM Factory
HENRY     Henry Lieberman        CADR-4    14    912 Lounge Lizard
NIS       Keith Nishihara        CADR-3          903 Vison x6765
DANIEL    Daniel Weise           CADR-1          907 [Son of CONS]
```

6.10. Communication

On ITS, there are three basic ways to communicate with other users: comlink, sends, and mail. Each of the three ways is best suited to a particular type of interaction.

6.10.1. SENDs

The SEND command is available for sending user a quick interactive (the other user must be logged in) message.

To get the fancy version of SEND which understands about talking to multiple networks, be sure to set the SNDFLG variable as shown in the sample DDT init file above.

Simply type the command SEND, followed by the uname of the person you wish to send to, a SPACE or carriage return, and your message. Your message may be as long as you like; to end it, type control-C. Example:

```
*:send levitt Is the music room free right now? ↑C
*
```

If KWH typed the above command, on LEVITT's console would come the message:

```
*
[MESSAGE FROM KWH at MIT-AI  4:56PM]
Is the music room free right now?
```

A send will interrupt whatever was printing on the receiver's console. If you were not in DDT at the time you received a send, DDT will repeat the message when you exit to make sure you saw it. There are customization varibles to control this repeating.

You can also specify a *uname@host* in the SEND command, if the person you want to talk to is on another machine.

If the person you want to send to is on a non-ITS machine on the ARPAnet, you should use QSEND in lieu of SEND. QSEND is cross between SEND and MAIL, and works with many computers on the net.

If you do not want to receive sends from other users, you can use the GAG command.

```
*:gag 0
```

prevents incoming sends, and

```
*:gag 1
```

re-enables them.

It is usually considered anti-social to gag yourself.

6.10.2. Comlinks

Comlinks are the most interactive way of communicating with another user on ITS. When two or more users are in a comlink, they will each everything which is being displayed on any of the consoles. Although there is no limit to the number of people in a comlink, due to the psychology of human conversation, comlinks usually work best when only two people are linked at once; more people in a comlink tends to be confusing.

Comlinks are controlled by commands beginning with the special character called "backnext". This character actually exists only on Knight TV keyboards, but it may be simulated by typing control-underscore or MACRO. Note: on certain ascii terminals with non-standard keyborad encoding, the backnext character is obtained by typing control-questionmark.

To establish a comlink with another user, type the command:

BACKNEXT *Cuname*

This will cause ITS to print "OK" and link you together. If the person is in another comlink with someone already, ITS will ask you ("y" or "n") if you want to break in and have a multi-way comlink.

To end a comlink to another user, type the command:

BACKNEXT N

You do not have to be in DDT to use the comlink feature of ITS. Comlinks work "on top" of any program. While you are in a comlink, none of your typein is seen by DDT (or whatever other program you were running when you entered the refuse).

There is an etiquette for talking to people in comlinks.

- Because a comlink is a pretty violent interruption, you should usually ask someone with a send for permission to enter a comlink.

- Each person in the link takes turns typing. When you are done saying something, type a blank line to let the other person know you are finished.

- If you are in a multi-way comlink, you should sign your name or initials before you start typing, so that it is easy to inedntify who is saying what.

- The way to yell "shut up!" or otherwise get someone's immediate attention in a comlink is to beep their bell by typing control-G.

If you are in a comlink and wish to type something at the program (such as your HACTRN) which is running "underneath" the link, you can toggle "input" mode on and off with the command BACKNEXT I.

It is also possible to type at the programming running on the other person's console; this is called "slaving", because it allows you to take control of someone else's terminal. The command to toggle "slave" mode is BACKNEXT E.

Sometimes you may not wish to be interrupted with an incoming comlink. By putting your console into "query" mode, you can restrict who you will link with. If you are in query mode and someone tries to link with you, you will see the message:

```
LINK FROM KMP - QUERYING
```

You can answer with BACKNEXT Y for "yes" to allow the comlink, or BACKNEXT N to disconnect the person. If the person gets bored of waiting for your response, they can use the backnext-N command to unlink. To put your terminal in comlink query mode, type:

```
*:tctyp query
*
```

There are times when you will want to completely disallow comlinks, and there is no point in even being asked. If you put your terminal into "refuse" mode, all incoming comlinks will be rejected with the message REFUSED on the console of the person trying to link. To get into refuse mode, type the command:

```
*:tctyp refuse
*
```

To get out of query or refuse mode, use the command:

```
*:tctyp accept
*
```

There are additional comlink commands described in the file INFO;ITSTTY >.

6.10.3. TALK

If you want to link to someone who is on another ITS machine, you can use the TALK program. The program is run by typing:

```
*:talk uname@host
```

To exit TALK terminating the conversation, the user who initiated TALK must type ↑C.

This method of comlinking is less versatile than the backnext commands, and only works across ITS machines (not locally).

Another useful comlink program is UNTALK. UNTALK is similar to TALK but does not work across machines. However, on a display terminal, UNTALK splits the screen horizontally and allows the two people to type at the same time on their part of the screen.

6.10.4. Mail

ITS has several programs for sending and receiving mail. These programs do not actually perform the sending of mail; this is handled by a deamon called COMSAT.

The easiest way to send mail is to run the MAIL program. Example:

```
*:mail
To: rich@mit-oz
Text:
Hi,
Will you be able to give a Programmer's Apprentice
talk at the next Revolving Seminar?  I need to know
by Wednsday.
↑C

:KILL
*
```

If you type an ALTMODE (on an ascii terminal, an "ESC"), MAIL will prompt you for a command. Type "?" at it to begin using the program's self documentation.

There are several ways to read your mail. The simplist is the PRMAIL command. Typing the PRMAIL command} at DDT will print your mail and offer to delete it after you have seen it. For more flexability you should use a mail reading program such as BABYL.

The MSGS program will examine the system bboards and offer to print the messages in it on your terminal. The GMSGS program is similar, but puts the messages into your mail file instead.

6.11. Using Network Programs

The FTP program for transferring files from the ARPAnet Internet is called FTP. The TELNET program is called TELNET and the SUPDUP program is called SUPDUP. These programs are mostly self documenting.

7. Using TOPS-20

This chapter will tell you how to get started using the MIT-OZ timesharing machine.

OZ consists of a DEC-2060T (KL model B) processor, two megawords of 36-bit MF-20 memory, eight 177MB RP06 disk drives, two TU77 tape units, and various network interfaces including ARPAnet and Chaosnet.

MIT-OZ runs an enhanced version of the DEC TOPS-20AN operating system which is maintained by the MIT computing community. Tops-20 is also sometimes called Tops-20.

7.1. Logging in

In order to login to Tops-20, a user needs a loginable directory on the system's primary disk structure (the PS: device).

Currently, Tops-20 is set up so that a person's username is the same as the name of their login directory. This means that if you do not have a toplevel directory[8] your username is probably a cumbersome string.

Most users have toplevel directories, so this problem is often avoided. However, as the number of people in login-subdirectories grows, this username convention becomes uncomfortable. An uncoming rewrite of the Inquire database system will change the naming convention so that each person has a unique, short username which is not necessarily the same as the person's directory name.

To log into OZ, get to a terminal connected to the machine (perhaps a hardwired terminal, or a Chaosnet TIP terminal) and type control-C. The system should respond by identifying itself. Now, type the login command:

`@login` *username password*

If you like, you may actually omit the command name and simply type your username and password. To logout, type the word LOGOUT and press RETURN.

7.2. Commands

When you login to Tops-20, you are typing at a program called the EXEC. The EXEC understands many commands for such things as: managing your file directories, getting information about the system, starting programs, and talking to other users on the machine.

To type a command, simply type its name, and any parameters for the command, and press RETURN.

The EXEC usually prompts you with an atsign (@) when it is ready to receive a command. Some comands put you in modes in which only certain subcommands are valid; in that case your prompt will be a double-atsign.

---

[8] A directory whose immediate superior in the file system hierarchy is the ROOT-DIRECTORY

7.2.1. Control Characters

Tops-20 programs generally understand the following control characters for editing your typein and controlling your status.

RUBOUT deletes the last character you typed, while ↑W rubs out the last word. To flush the line you are typing, use ↑U. You can redisplay the line you are typing with ↑R, and redisplay the line on a clear screen with ↑L.

You can get a quick status report telling what program you are running, what instruction it is executing, and how loaded the entire system is, by typing control-T ↑T. This feature does not work if you are in Emacs, since ↑T is an Emacs command.

7.2.2. Pausing

If Tops-20 knowns the length of your terminal (if you told it what kind of terminal you had, or connected using the SUPDUP protocol), it can pause at the end of each screenful of output if you like. The command:

```
@ter pause end
@
```

Will enable this feature. At the end of a page, Tops-20 will beep your console's bell and wait for you to press contol-Q.

In the latest release of Tops-20 (which at the time this document was prepared was not yet in use, but should be soon) there is a new way of telling the system to proceed to the next screenfull. The command sequence:

```
@ter pause end
@ter its-style
@ter verbose
@
```

will make the system pause with the message [***More***], and wait for a SPACE to be typed before proceeding. Any character besides a SPACE will flush the output. Many users consider this behaviour a significant improvement.

7.3. Recognition and HELP

The user interface on Tops-20 has an interesting feature called completion and recognition. Many programs, including EXEC, use this feature.

Recognition is Tops-20's attempt to help you remember the arguments to a command. If you type a question mark (?), all the possible completions for the command you are typing are displayed. If you type the ALTMODE key at the end of the unambiguous abbreviation of the part of the command you are typing, Tops-20 will try to complete what you are typing. If there are more components to the command you are typing, you will be shown some "guide words" to suggest what the next component is. Tops-20 will also try to complete unambiguous file names for you when you type ALTMODE. If the system cannot figure out how to recognize your command or complete your file name, it will beep your console's bell.

You are encouraged to login and experiment with command recognition. A good way to start is to run the TOPS20 program, which will let you practice this. To run the TOPS20 program, just type it's name and press RETURN.

Online documentation for Tops-20 is available from the Help files (displayed by the HELP command) and the XINFO (Emacs INFO) program.

7.4. System Security

Vanilla Tops-20 tries to provide file security and other forms of restriction protection. Like most forms of computer security, these features can be circumvented in various ways.

MIT-OZ provides an atypical security environment. Instead of preventing users from doing things, the system tries to make users responsible for their actions. This is done by requiring certain functions to be done via a special mechanism which keeps audit trails.

A privileged system deamon called the Protector is always running on MIT-OZ. Unprivileged programs (available to all users) contact the Protector and request some special function which the user could otherwise not do. The Protector performs the requested function, and leaves an appropriate audit trail.

The most common uses of the Protector mechanism are: changing the protection on a file which you need to access, creating new accounts, managing user/directory groups, and performing such operations tasks as shutting down the timesharing.

Under Tops-20, each user's account on Tops-20 has a set of *capabilities* associated with his account. These capabilities determine which "privileged" operations the user is allowed to perform. Before using special capabilities, a user must *enable* then with the ENABLE command. When done doing whatever it was that required the capability to be enabled, it is usually a good idea to disable the capabilities again. When enabled, your prompt changes to remind you that you are enabled. When you are done using your enabled capabilites, you should use the Disable command to revert to your normal capabilities.

Currently, the Protector is not fully implemented.[9] To ensure that system security does not get in the way of users, all lab members have accounts with capabilities allowing them to bypass any system security. When Protector is released for general use, these accounts will be modified appropriately.

7.4.1. File Security

Most users at the AI lab feel no need to protect their files from other lab members. There is generally no information stored online which is not public to at least the lab community, and lab members are encouraged to be responsible when using the machine, and careful with other people's files.

Tops-20 allows users to make their files and directories inaccessable to other users. You can

[9] A remedial Protector deamon is operational now, but is very limited in the functions it provides.

restrict particular kinds of access (read, write, delete, etc.) to a file or directory from three individual sets of users.

The default file system protection on OZ allows all users to access each others' files. User's are encouraged to stick to the default protection, which is the simplist and friendliest thing to do.

When setting the restrictions on a file or directory, you specify which access will be allowed to three types of users: the owners, the people in *groups* with you, and all the rest of the users. Protection for files is specified as a mask of three two-digit octal numbers. Each number in the protection code corrosponds to one of the three categories. (Protection for directories is specified as four numbers, since directories recognize an additional category of users: those connected to the superior directory.)

A *group* of users on Tops-20 is a collection of users who will all access a file or directory under the same restrictions. A directory can be placed in multiple groups, but there is no way to set up different restrictions for different groups.

Sadly, group ids are numbers, not symbolic names. As such, the information about groups given in your directory information will look pretty cryptic. A member of user group N has the number N in the User-Groups field of his or her directory's information. If a directory is accessible to users in group N, it will have have the number N in its Directory-Groups field of its directory information.

To see a user group correspondance table, look under at the USER-GROUPS topic in HELP. Here is a sample listing of the table:

```
Number          Group
----------      ------------------------------------------------
10              System source directories
100             Music Hackers
120             Programmer's Apprentice
210             Analogy System Development
300             MUMBLE system
401             Brady: shape description
503             LISP sources and libraries
504             EMACS distribution sources
507             LSB sources and libraries
1730            SPICE Circuit Simulator
2000            ACTOR system
```

Figure 7-1: Sample User-Groups Table

File protection is very useful for safeguarding against accidentally destroying files. For example, you can set up your directory so that you cannot delete your files without changing the mode of access you are using.

More information about file security can be found under the HELP topics "Protection", and "Security".

7.5. Manipulating Forks

A user and all of his programs run in a single Tops-20 *job*. Each of the programs in a job usually runs in its own *fork*. The EXEC manages these job forks for you, and allows you to switch between them.

To start a program in a fork, simply type its name. Example:

```
@emacs
```

The command INFO FORK prints out the names of all your forks and some information about their state. (You can abbreviate this to INFO.) Here is an example with an currently selected MM fork, and an EMACS fork:

```
@inforMATION ABOUT foRK-STATUS
     EMACS (1): Kept, HALT at 61364  00:00:15.6
 ==> MM (2): ↑C from IO WAIT at 35410 00:00:00.1
@
```

When you exit most programs, their forks stays around until you explicitly flush them. The RESET command gets rid of a fork:

```
@reset emacs
```

Typing RESET with no argument gets rid of the currently selected fork. Sometimes a RESET with no argumnet doesnt want to get rid of the current forl. Giving the argument "." to a RESET command will always get rid of the current fork.

Giving the argument "*" to a RESET gets rid of all your forks:

```
@i fo
     EMACS (1): Kept, HALT at 61364  00:00:15.6
     MM    (2): ↑C from IO WAIT at 35410 00:00:00.1
     SEND  (3): ↑C from IO WAIT at 35410 00:00:00.1
 ==> CFTP  (4): ↑C from IO WAIT at 35410 00:00:00.1
@rese *
@i fo
@
```

To create a new fork which contains the same program as an existing fork, type the name of the program and use the ALTMODE key to complete it. The EXEC will name the new fork *FOOn*, where *n* indicates how many other forks there are with the same basic name. Example:

```
@emacS.EXE.62203
[EMACS0]
[Keeping]
```

A quick status report showing the current fork's status can be usuually be obtained with ↑T at any time.

In most cases control-c (↑C) will suspend the current process and give control to the superior EXEC.

After suspending a process with ↑C the "continue" command will resume the process from where it

left off.

```
@continue
```

CONTINUE can be abbreviated to just "C".

You can also tell CONTINUE which fork to resume. Thus,

```
@continue cftp
```

would continue your CFTP fork.

Certain programs, notably EMACS, sometimes work better if they are continued with the command START, instead of CONTINUE. START restarts the program at a special entry point.

7.6. Files

On Tops-20, users store data in *files*. To access a file, you must know its name. File names consist of several component fields; the syntax of a file name usually looks like:

Device:<Directory>Name.Extension.Generation

All the fields in the file name are optional; only enough need exist to uniquely identify the file. Example:

```
PS:<CSTACY>FOO.BAR.3
```

The *Device* component tells on what peripheral device on the machine the file resides. This is usually the name of some {disk structure} or a *logical name*. A disk structure may refer to either a single disk pack or to a set of packs. For example: PS (public structure), the primary disk structure on the machine, consists of several mounted disk packs.

Typing the command

```
@help structures
```

Will provide you with information about what structures are currently mounted on the system, and a brief description of what they are for.

The *Directory* identifies who owns the file.

The *Name* component is the field which most uniquely identifies a file. The Name component of a filename can be pretty long.

The *Extension* component is the field which dALTMODEribes what sort of data is stored in the file; this is the file "type". Extensions are usually three or four characters. Some common extensions include: TXT (text), LISP (Lisp source), and EXE (executable binary image).

The *generation* is the version number of a file. When a new version of the file is written out, the generation number is incremented. Here are some examples of legal file names:

```
PS:<AGRE>SCDPCM.QFASL
PS:<CSTACY>MAIL.TXT
SS:<EMACS>MODLIN.TEC
```

7.6.1. Directories

Files are stored in directories.

Directories are per-structure; having a directory on one structure does not imply having one on whatever other structurea are mounted. In order to log in, a user on Tops-20 must have a directory (which has the same as her login name) on PS: structure.

A feature of files on Tops-20 is that they do not necessarily go away when deleted. To make a file really go away from disk, you need to *expunge* the directory the file is in. Sometimes, when the system gets low on disk space, it will expunge your deleted files for you. The EXPUNGE command expunges directories:

```
@expunge <mumble>
 PS:<MUMBLE> [33 pages freed]
@
```

In Tops-20 there is the notion of a *connected directory*. When a user is connected to a directory, he enjoys all the access rights of the directory owner. The directory you are connected to when you log in is your own directory on the PS: structure.

7.6.2. Subdirectories

Tops-20 supports subdirectories, which allow you to group related files together and help manage your directory. A subdirectory is pretty much like your toplevel directory, and has a name which looks something like:

```
<YOURNAME.FOO>.
```

Subirectories make files on the system appear to be tree-structured. The top node in the three is the <ROOT-DIRECTORY>. From this root at the apex, individual directories belonging to users emerge. Underneath a user might be additional directory nodes. Many users find that subdirectory nodes help them to organize their files.

7.6.3. Logical Names

A *logical name* can be thought of as a way of referring to a particular set of files by an alias. Logical names are especially useful if you dont want to specify the name of a directory.

A logical name can provide a generic reference to some directory whose actual name might change from time to time; the logical name is a pointer to the directory even though the directory's name might change.

Logical names may be system-wide or job-wide. Job-wide logical names apply only to one user, and system-wide logical names are defined for all users. To define a job-wide logical name, use the define command:

@define *logical name: actual pathname*

7.6.4. Wildcards

You can refer to more than one file at a time. This is done by using an asterisk (*) as a pattern matching component of the name. The asterisk is a "wildcard character"; it matches any name. Here is an example of a filename which specifies all the files in the <KMP> directory on the PS: structure which have the Name "Foo":

```
PS:<KMP>Foo.*.*
```

The above file name refers to all these files if they exist:

```
PS:<KMP>FOO.LISP.1
PS:<KMP>FOO.LISP.2
PS:<KMP>FOO.QFASL.2
```

Referring to sets of files using wildcards is especially useful when you are performing a listing, renaming, or deleting operation.

7.6.5. Useful File Manipulation Commands

There are many commands which are useful for manipulating files and directories.

The TYPE command will print out the contents of a file on your terminal.

The CONNECT command is used to connect to a directory as though you were its owner.

The DIRECTORY, and VDIRECTORY ("verbose directory") commands are usful for seeing which files are in a directory.

The DELETE command deletes files. The EXPUNGE command gets rid of deleted files in a directory.

The COPY command makes copies of files, and the RENAME command changes a file's name. You can move files around in the file system with these commands, but the RENAME command will not move a file across a disk device.

7.6.6. Hardcopying files

To send a file to the Dover printer, use the DOVERSEND program. If the file is not already in "press" format, you will need to use the PRESSIFY program first. These programs are self-documenting.

7.6.7. Creating Directories

A directory has a name and some attributes. These attributes determine such things as whether the directory can be used to login with, how much space the files in it can consume, and what groups it is part of.

To list the attributes of a directory, use the INFO DIR command. Example:

```
@info dir <pa>
 Name: PS:<PA>
 Working disk storage page limit +inf
 Permanent disk storage page limit +inf
 Creator-of-login-subdirs
 Login-subdirs-allowed
 Number of directory 720
 Account default for Login - none set
 Maximum subdirectories allowed 987
 User groups 120
 Directory groups 120
@
```

The *quota* attributes, such as the amount of disk used, are typically some very large number (displayed as "+inf"). This is because most users on OZ have no interest in arbitrary limits to the amount of storage that lab members use.

The BUILD command allows you to make new directories, and to alter the attributes of existing directories.

```
@build
ps:<jdo.thesis>
@@
```

Once inside BUILD mode, you can enter Build subcommands. Type LIST VERBOSE at BUILD to see what the different characteristics of the directory you can modify. When you are done in BUILD mode, type RETURN twice to exit back to the EXEC.

One interesting paramter to set with BUILD is GENERATIONS-TO-KEEP. This variable says how many versions of a file to let you write to disk in the directory. If this number is zero, any time you write a new version of a filem, it is stored on disk. If this number is a positive number *n*, only the most recent *n* versions of your file will be available; the old generations are automatically deleted when you write out a new one.

7.6.8. Changing Passwords

To set the password on a directory, use the SET command.

```
@set dir pass <danny>
Old Password:
New Password:
Retype new password:
@
```

When the Protector facility is fully operational, the capability to create "toplevel" directories and login-able subdirectories will be restricted from users, and new commands will be introduced.

7.7. Terminal Support

The version of Tops-20 which runs on OZ has been enhanced to provide some support for display terminals. The TERMINAL command is used to tell Tops-20 what sort of terminal you are on.

Example:

```
@terminal c100
@
```

Giving the command

```
terminal ?
```

to the EXEC will get you a list the terminals and options which Tops-20 understands.

7.8. Inquire

To make an Inquire entry on OZ, run the INQUIR program.

The WHOIS command will print on your terminal all the information associated with a user in the OZ INQUIR database. Example:

```
@whois kwh
KWH  Ken Haase Last logout Wed 24-Mar-82  01:23 from TTY212
 (Ken) Hacking amateur systems for Minsky  (Music)
 Work: NE43-352  (617) 253-1728
 Home: 4 Ames Street

KWH has no new mail,
[No plan]

@whois danny

DANNY Danny Hillis  Last logout Wed 24-Mar-82  00:23 from TTY216
 (Danny)  Hacking connection machine (CM)
 Work: NE43-711  (617) 253-6765
 Home: 111 Ivy St., Brookline, MA

DANNY has new mail from KWH at Wed-24-Mar-82 00:20,
last read Tue 23-Mar-82 20:28
[No plan]
```

Numerous commands and programs exist to obtain system status reports, FINGER and SYSTAT being the most commonly used.

```
@finger

 User     Name                  Job Subsys Idle TTY Console Location
HANSON    Christopher P. Hanson  9  EXEC          2 825 Hanson x5848
CARL      Carl Hewitt           15 EXEC    *:** . 3 813 Hewitt x5873
MINSKY    Marvin Minsky         17 BABYL        12 1200b dialup
PHW       Patrick H. Winston    20 EMACS        14 1200b dialup
GREN      Ian G. Mackey         21 FINGER       50 NE437A: Playroom
DAVIS     Randy Davis           25 UNPROTECT    52 HTJR: 819 Davis
```

```
PGA      Phillip G. Apley       26 TEX          53 CHAOS site Sportdeath-1
GJS      Gerald J. Sussman      31 SCHEME       66 CHAOS site MIT-MC
PGS      Patrick G. Sobalvarro  32 EMACS        67 LISPM-25: Puma Room

@systat

 Mon 30-Aug-82  06:59:11  Up 37:21:48
 8+5 Jobs   Load av  1.26   1.31  1.37

  Job     Line     Program User
   9       2        EXEC    Hanson
   15      3        EXEC    Carl
   17      12       BABYL   Minsky
   20      14       EMACS   Phw
   21*     60       SYSTAT  Gren
   25      52       UNPROT  Davis
   26      53       TEX     Pga
   31      66       SCHEME  Gjs
   32      67       EMACS   Pgs

    1      26       SYSJB1  Operator
    2      27       PTYCON  Operator
    3      30       CHARFC  Operator
    4      31       OPR     Operator
    5      32       PROTEC  Operator

@
```

7.9. Communication

On Tops-20, there are three basic ways to communicate with other users: comlink, sends, and mail. Each of the three ways is best suited to a particular type of interaction.

7.9.1. SENDs

The SEND command is available for sending user a quick interactive (the other user must be logged in) message. A send will interrupt whatever was printing on the receiver's console.

Simply type the command SEND, followed by the name of the person you wish to send to, a SPACE or carriage return, and your message. Your message may be as long as you like; to end it, type control-Z. Example:

```
@send levitt
Message (end with ↑Z or ALTMODE):
Is the music room free right now? ↑Z
@
```

If KWH typed the above command, on LEVITT's console would come the message:

```
KWH, TTY20, 27-Jul-82  4:56PM
Is the music room free right now?
```

You can also specify a *name@host* in the SEND command, if the person you want to talk to is on

another machine.

7.9.2. Comlinks

Comlinks are the most interactive way of communicating with another user on Tops-20. When two users are in a comlink, they will each everything which is being displayed on any of the consoles.

Since a comlink will interrupt whatever a person is doing, you should probably SEND to the person asking if it is OK to link first.

To establish a comlink with another person, use the talk command:

```
@talk name
```

This will link your terminals together. Each person in the link takes turns typing. When you are done saying something, type a blank line to let the other person know you are finished.

Since you probably do not want to type commands to your EXEC during a comlink, you should type the REMARK command:

```
Link from JFC, TTY10
@remARK (mode)
Type Remark. End with ↑Z.
this is a comment
This is another comment.
@
```

The person who you linked to will also have to type the REMARK command. When you are done conversing, type ↑Z and the break command:

```
↑Z
@breAK (links with) *
@
```

There is an etiquette for talking to people in comlinks.

- Because a comlink is a pretty violent interruption, you should usually ask someone with a send for permission to enter a comlink.
- Each person in the link takes turns typing. When you are done saying something, type a blank line to let the other person know you are finished.
- If you are in a multi-way comlink, you should sign your name or initials before you start typing, so that it is easy to inedntify who is saying what.
- The way to yell "shut up!" or otherwise get someone's immediate attention in a comlink is to beep their bell by typing control-G.

7.9.3. Mail

Tops-20 has several programs for sending and receiving mail. These programs do not actually perform the sending of mail; this is handled by a deamon called XMAILER. Probably the best way to process your mail is with the BABYL program.

The SYSMSG and FORUM programs will examine the system bboards and offer to print the messages in it on your terminal.

In order to receive mail on OZ, you need to have a file named MAIL.TXT in your directory. This file is created automatically whenever a directory is created, so unless you have somehow deleted it manually, you should not need to worry about it. Unlike some Tops-20 sites, you only need to alter your Inquire entry with INQUIR to have your mail forwarded to another address.

7.10. Customization

When you issue the LOGIN command, the file LOGIN.CMD in your directory is run, if it exists. This file should contain a series of EXEC commands to set up your session.

Here is a sample login init file which sets up some terminal characteristics and starts up the bboard reading programs:

```
term page
term wrap
sysmsg
forum
take
```

Figure 7-2: Sample EXEC init file

If you have many subdirectories, another useful thing (not shown here) to do in an init file is to define some handy logical names for accessing your files.

7.10.1. PCL

Some users like using an alternate EXEC called the PCL EXEC. This EXEC has several features, its main virtue being the ability to define new commands and command abbreviations. For more information, read the HELP information for PCL.

7.11. Using Network Programs

The FTP program for transferring files from the ARPAnet Internet to OZ is called FTP. For transferring files to OZ from the Chaosnet, use the CFTP program. The TELNET program is called TELNET, and the SUPDUP program is called VCHTN. These programs are more or less self-documenting.

8. References

[Christman 80]
Christman & Sjobrg.
The Last Whole XGP Font Catalog.
AI Memo 197, Artificial Intelligence Laboratory, Massachusettes Institute of Technology, March, 1980.

[Ciccarelli 78]
Ciccarelli.
An Introduction to the EMACS Editor.
AI Memo 447, Artificial Intelligence Laboratory, Massachusettes Institute of Technology, January, 1978.

[Crispin 77]
Crispin, M.
SUPDUP Display Protocol.
RFC 734, Network Information Center, SRI International, October, 1977.
NIC 41953

[Knuth 80]
Knuth.
Tau Epsilon Chi, a system for technical text.
American Mathematical Society , 1980.

[Mathlab 77]
Mathlab.
Macsyma Reference Manual.
Technical Report, Laboratory for Computer Science, Massachusettes Institute of Technology, December, 1977.

[Moon 81a]
Moon.
Chaosnet.
AI Memo 628, Artificial Intelligence Laboratory, Massachusettes Institute of Technology, June, 1981.

[Moon 81b]
Moon & Wechsler.
Operating the Lisp Machine.
AI Working Paper , Artificial Intelligence Laboratory, Massachusettes Institute of Technology, April, 1981.

[NIC 82a]
NIC.
ARPANET DIRECTORY.
Defense Communications Agency, 1982.

[NIC 82b]
Internet Protocol Transition WorkBook.
Defense Communications Agency, 1982.

[Reid 79]
Reid, B.K. & Walker, J.H.
SCRIBE Introductory User's Manual.
Technical Report, Carnegie-Mellon University, July, 1979.

[Shapiro 81]
Shapiro.
R-Dover Font Sampler.
Technical Report, Laboratory for Computer Science, Massachusettes Institute of Technology, September, 1981.

[Snyder ??]
Snyder, A. and Moss, Eliot.
R Reference Manual.

[Stallman 78]
Stallman, Richard M.
The SUPDUP Graphics Extension.
RFC 746, Network Information Center, SRI International, March, 1978.
NIC 43796

[Stallman 81a]
Stallman.
Emacs Manual for ITS Users.
AI Memo 554, Artificial Intelligence Laboratory, Massachusettes Institute of Technology, April, 1981.

[Stallman 81b]
Stallman.
Emacs Manual for TWENEX Users.
AI Memo 555, Artificial Intelligence Laboratory, Massachusettes Institute of Technology, April, 1981.

[Stallman 81c]
Stallman.
EMACS: The Extensible, Customizable, Self-Documenting Display Editor.
AI Memo 519, Artificial Intelligence Laboratory, Massachusettes Institute of Technology, March, 1981.

[Wechsler 82]
Wechsler.
Zmacs User's Guide.
Symbolics, Inc., 1982.

[Weinreb 81]
Weinreb & Moon.
Lisp Machine Manual.
Artificial Intelligence Laboratory, Massachusettes Institute of Technology, 1981.

[Xerox 79]
Alto User's Handbook.
1979.

Index