

MASSACHUSETTS INSTITUTE OF TECHNOLOGY  
ARTIFICIAL INTELLIGENCE LABORATORY

A.I. Working Paper No. 307

May, 1988

**The Novice's Guide to the UNIX at the AI Laboratory**  
**Version 1.0**

Liz A. Highleyman

**ABSTRACT:** This is a manual for complete beginners. It requires little knowledge of the MIT computer systems, and assumes no knowledge of the UNIX operating system. This guide will show you how to log onto the AI Lab's SUN system using a SUN III or similar workstation or a non-dedicated terminal. Many of the techniques described will be applicable to other computers running UNIX. You will learn how to use various operating system and network features, send and receive electronic mail, create and edit files using GNU EMACS, process text using  $\gamma$ TeX, and print your files.

© Massachusetts Institute of Technology (1988)

A.I. Laboratory Working Papers are produced for internal circulation, and may contain material that is, for example, too preliminary, too detailed, or too site-specific for formal publication. It is not intended that they should be considered suitable for reference in scientific papers.



## CONTENTS

PREFACE . . . . .	5
ACKNOWLEDGEMENTS . . . . .	5
1. INTRODUCTION . . . . .	6
1.1 The UNIX Operating System. . . . .	6
1.2 Getting an Account . . . . .	6
1.3 The AI Laboratory Network . . . . .	6
1.4 Note on Terminals and Keyboards . . . . .	7
1.5 Fail-Safe Key Commands . . . . .	7
2. GETTING STARTED . . . . .	9
2.1 Logging-In to a SUN Workstation . . . . .	9
2.2 Logging-In to a SUN from a Non-Dedicated Terminal. . . . .	9
2.3 Accessing a SUN Using <code>supdup</code> and <code>telnet</code> . . . . .	9
2.4 Preliminaries . . . . .	9
2.5 The Shell . . . . .	10
2.6 Accessing Other Computers from a SUN. . . . .	10
3. ADMINISTRATIVE PROCESSES. . . . .	11
3.1 Checking for Availability . . . . .	11
3.2 Listing Processes and Jobs . . . . .	11
3.3 Foreground and Background Jobs . . . . .	12
3.4 Killing Processes and Jobs . . . . .	12
3.5 Logging Out. . . . .	13
3.6 The Reboot Operation . . . . .	13
4. SOME BASIC OPERATIONS . . . . .	15
4.1 Getting On-Line Help . . . . .	15
4.2 System Security . . . . .	15
4.3 Identifying Yourself and Other Users. . . . .	15
4.3.1. Using <code>watson</code> . . . . .	15
4.3.2. The <code>finger</code> and <code>whois</code> Commands . . . . .	16
4.4 Your History. . . . .	16
4.5 Date, Time and Calendar Display. . . . .	16
4.6 Calculator. . . . .	17

5. DIRECTORIES AND FILES . . . . .	18
5.1 The UNIX Directory and File Structure . . . . .	18
5.2 Directories and Pathnames. . . . .	18
5.3 Identifying Your Current Directory. . . . .	19
5.4 Changing Directories . . . . .	19
5.5 Creating and Removing Directories. . . . .	19
5.6 Listing Files. . . . .	20
5.7 Types of Files. . . . .	21
5.8 Setting Access Masks - Permission Options . . . . .	22
6. MANIPULATING FILES . . . . .	24
6.1 Creating Files . . . . .	24
6.2 Saving Files. . . . .	24
6.3 Viewing Files . . . . .	25
6.4 Removing Files. . . . .	25
6.5 Concatenating Files . . . . .	26
6.6 Moving Files . . . . .	26
6.7 Copying Files . . . . .	26
6.8 Remote Copying from Other UNIX Systems . . . . .	27
6.9 Linking Files. . . . .	27
7. TRANSFERRING FILES . . . . .	28
7.1 File Transfer Program Usage. . . . .	28
7.2 OZ <---> SUN Transfers. . . . .	28
7.3 Transfers from Reagan . . . . .	29
8. REDIRECTION AND PIPING. . . . .	30
8.1 Redirection . . . . .	30
8.2 Piping . . . . .	30
9. USING ELECTRONIC MAIL . . . . .	32
9.1 Mail Basics . . . . .	32
9.2 The UNIX Mail Program . . . . .	32
9.3 Reading Your Mail . . . . .	32
9.4 Sending Mail. . . . .	34
9.5 Other Sending Methods: talk and write . . . . .	34
10. EDITING WITH GNU EMACS. . . . .	36
10.1 Introduction . . . . .	36
10.2 On-Line Help . . . . .	36
10.3 Using Buffers . . . . .	37

10.4	Using Windows Within GNU EMACS. . . . .	37
10.5	Entering GNU EMACS to Edit or Create a File . . . . .	37
10.6	Basic Editing Commands . . . . .	38
10.6.1	Moving Within the File. . . . .	38
10.6.2	Inserting and Transposing Text . . . . .	39
10.6.3	Deleting and Undeleting Text. . . . .	39
10.7	Searching for Strings of Text . . . . .	39
10.8	Moving and Copying Blocks of Text. . . . .	40
10.9	Some Special Character Commands . . . . .	40
10.10	Extended ( M-x) Commands . . . . .	41
10.11	Using Keyboard Macros . . . . .	43
10.12	Saving the File and Exiting GNU EMACS. . . . .	43
11.	WORKING WITH FILES. . . . .	45
11.1	Locating Strings within Files . . . . .	45
11.2	Counting Elements in a File. . . . .	45
11.3	File Comparison . . . . .	45
11.4	Pruning, Sorting and Splitting Files . . . . .	45
11.5	The UNIX Spelling Checker . . . . .	46
12.	TEXT FORMATTING WITH Y <sub>T</sub> E <sub>X</sub> . . . . .	47
12.1	Introduction . . . . .	47
12.2	Using Y <sub>T</sub> E <sub>X</sub> : The Basic Idea . . . . .	47
12.3	Y <sub>T</sub> E <sub>X</sub> Opening Commands . . . . .	48
12.3.1	Setting Sizes . . . . .	48
12.3.2	Inputting Auxiliary Files . . . . .	48
12.3.3	Defining Macros . . . . .	48
12.4	Stylistic Commands . . . . .	49
12.4.1	Layout Commands . . . . .	49
12.4.2	Spacing Commands . . . . .	50
12.4.3	Font Changing Commands . . . . .	51
12.4.4	Special Characters . . . . .	52
12.5	Formatting Equations . . . . .	52
12.5.1	Horizontal Mode . . . . .	52
12.5.2	Vertical Mode . . . . .	53
12.5.3	Some Basic Equation Commands . . . . .	53
12.6	Creating Tables . . . . .	54
12.7	Closing Commands . . . . .	55
12.8	Running Y <sub>T</sub> E <sub>X</sub> . . . . .	55
12.9	The .dvi and .log Files . . . . .	55
12.10	Debugging the .y <sub>T</sub> e <sub>X</sub> File. . . . .	56

12.11	Previewing the .dvi File . . . . .	57
12.12	A Sample-Document. . . . .	57
13.	PRINTING FILES. . . . .	60
13.1	Printers Available at the AI Laboratory . . . . .	60
13.2	Defining a Printer . . . . .	60
13.3	Checking the Printer Status and Queue . . . . .	60
13.4	Printing a File. . . . .	61
13.5	Stopping a Print Job. . . . .	61
13.6	Printer Errors . . . . .	61
13.7	Unwedging a Printer . . . . .	62
14.	OTHER PROGRAMS AND PACKAGES. . . . .	63
14.1	C . . . . .	63
14.2	Lisp. . . . .	63
14.3	Wreq . . . . .	63
14.4	Games. . . . .	63
15.	APPENDIX A: LIST OF RESOURCES. . . . .	64

## PREFACE

This is a manual for users who are unfamiliar with the UNIX operating system. You will learn how to log onto the AI Lab SUN system using a SUN III or similar workstation or a non-dedicated terminal. You will learn how to use various operating system features, send and receive electronic mail, create and edit files using the GNU EMACS editor, process text using  $\gamma$ T<sub>E</sub>X, and print your files. The GNU EMACS editor and  $\gamma$ T<sub>E</sub>X text formatting program will allow you to do advanced word processing and equation formatting with professional looking results. Another text formatting program, LaTeX, is also available on these systems and will be described briefly.

This manual is organized so that topics are presented in the likely order in which you will use them. For the sake of coherence, all material regarding a given topic is presented together as much as possible.

There are certain conventions you should keep in mind as you use this manual. The *control key* is designated by the notation `ctrl-`. The *meta key* is represented as `M-`. The meta key will be either the `<pause>` key or the `<esc>` key, depending on your terminal, when you are using an Ann Arbor; it will be the `<esc>`, `<left>`, and `<right>` keys on a SUN workstation keyboard. A carriage return is designated by `<CR>`. Computer commands, file names, and computer output will be shown in **typewriter font**. The UNIX system is generally sensitive to case. Most UNIX commands consist only of lower-case letters; commands that combine upper-case and lower-case letters will be shown as they should be entered by the user. [Square brackets] indicate that the words within should be replaced by a specific value. For example [username] should be replaced by an appropriate user ID, omitting the brackets. *Italics* indicate new terms, which are defined in the glossary at the end of the guide. Because it is not yet certain what will become the *de facto* window standard at the AI Lab, use of windows on the SUNs will not be discussed in detail, though some points about window usage are mentioned. I have tried to keep these as general as possible, using the X window system as a point of reference. Since mouse functions may be customized by the user, there is little point in describing a specific mouse command configuration. For more information about window usage, consult the X11 Manual (O'Reilly and Associates, Newton, MA).

As a resource for beginners, this manual is necessarily incomplete. At the end of the guide is a list of more advanced manuals and resources that you may wish to consult once you have mastered the material in this guide.

Please report any errors or suggestions for improvement to `liz@wheaties.ai.mit.edu`.

## ACKNOWLEDGEMENTS

This manual was written at the MIT Artificial Intelligence Laboratory, and borrows liberally from **The New Idiot's Guide to OZ** (Liz A. Highleyman, AI Laboratory Working Paper 298, ©Massachusetts Institute of Technology (1987)). Many thanks to those who offered helpful advice and comments, especially Diane Dustman, Brain Fox, David Seigel and Laurel Simmons.

## 1. INTRODUCTION

### 1.1 The UNIX Operating System

UNIX is an *operating system*, the main set of programs which run the computer, control resources, and allow multiple users to access the system. UNIX was developed at AT&T Bell Laboratories. The AI Laboratory SUN system currently runs Berkeley Standard Distribution (BSD) release 4.2.

UNIX is a “programmer’s” operating system, written in C, which facilitates complex program-oriented usage. The basics of UNIX are, however, easy to learn, and the system is quite efficient for administrative and word processing uses as well. A small subset of UNIX commands will allow you to get started and perform numerous tasks.

UNIX is different in some major respects from an operating system such as TOPS-20. In general, command operation is silent; the system usually will not tell you when an operation has been performed, although it will let you know if it cannot do what was requested. UNIX also tends to be a bit less forgiving than TOPS-20, for example, in regard to single version file backup and the lack of a “soft delete” capability. If the user applies recommended safety procedures and develops cautious habits, however, these “features” should not be a problem. UNIX also offers important advantages, most notably its nested directory structure and easy subdirectory usage, and its plethora of commands for a broad range of uses. Because UNIX is a popular and widely used operating system, new applications and improvements are constantly being developed.

### 1.2 Getting an Account

Before you begin to work on the SUN system, you will need to have an account set up for your use. An account will give you a *username*, also known as a *login name* or *user ID*, and a certain amount of allocated resources. You will choose your username when your account is set up (usually your initials or a nickname). You will also select a password, although it will not be required on many machines. Your supervisor or sponsor will be responsible for authorizing your new account, and one of the system administrators or technical support people at the Lab will set up the account for you.

### 1.3 The AI Laboratory Network

The SUN System at the AI Lab is a network consisting of numerous individual workstations, four general file servers, and two multi-user SUNs with connections to non-dedicated terminals. Most of the individual SUN workstations are “private” machines, ranging from SUN 3/50s to SUN 3/110s. The file servers are SUN 3/280s used to store files and public programs; these are named *wheaties*, *alpha-bits*, *grape-nuts* and *trix*. The multi-user SUNs are *wheat-chex* (for the 7th floor) and *rice-chex* (for the 8th and 9th floors).



The Lab's UNIX machines are connected to one another via the Internet. The Internet is connected to the campus Ethernet, allowing access to Project Athena machines, ARPANET, etc. The network uses a protocol called NFS (Network File System) to allow file access and transfer. Programs are also available to allow transfers to and from machines not using NFS.

#### 1.4 Note on Terminals and Keyboards

There are currently a number of different types of terminals and workstations in use at the AI Lab, including SUN workstations, Lisp Machines, and non-dedicated Ann Arbor Ambassadors and XLs. The keyboard layout on these machines varies quite a bit. Associated with your personal directory will be a *login profile*, a file called `.login`, which contains commands to set the parameters specific to your terminal. This file will be set up when your account is initiated.

There are certain special keys that you should be familiar with. The `<delete>` key, labelled `<del>` or `<rubout>` on most keyboards, will delete the text preceding the cursor. The `<back space>` key does not always delete: it may either back up without erasing or print `ctrl-h` (the backspace character).

Many commands require the use of the *control key*. This is to the left of the letter keys on most terminals. In this manual, the control key is represented by the symbol `ctrl-`. For example, `ctrl-c` indicates that you should press the control key, and **simultaneously** press the desired character key, in this case `c`. Other special keys are the *meta* and *escape* keys. These keys will vary by keyboard; they are used to give different meanings to character commands. The Ann Arbor Ambassador terminals use the `<pause>` key, located in the lower left-hand corner, as the meta key. The Ann Arbor XL keyboard uses the `<esc>` key at the upper left. The SUNs use the `<esc>` key, as well as the `<left>` and `<right>` keys on either side of the `<space bar>`. On Lisp Machines, the meta key is labelled `<meta>`. In this guide, the meta key will be represented by the symbol `M-`. For example, `M-v` indicates that you should press the meta key **followed by** the desired character, in this case `v`; this is contrast to the escape sequence, which is achieved by pressing the `<esc>` key and the desired character key **simultaneously**.

#### 1.5 Fail-Safe Key Commands

If you ever type a command that UNIX does not understand, it will reply: `Command not found`. If you are using the T shell (see Section 2.5), the computer will display the incorrect command on the next line, allowing you to edit it; this allows you do avoid re-typing the entire command. Typing `ctrl-u` will erase the command and allow you to start over.

Most of the Lab UNIX machines also have a feature that will complete commands, as well as directory and file names, for you. Type enough letters of the command so that it is unambiguous (i.e., cannot be confused with any other), and press `<tab>`. The full command will then appear on the screen. If it is correct, hit `<CR>` and the command will be executed. If not, use the editing commands to amend it.

The `ctrl-c` sequence can be used to do an emergency interrupt of whatever is currently in progress (try it twice if once doesn't work); this will return you to the prompt. Typing `ctrl-c` will cause whatever you were currently working on to be lost. A less drastic measure is the `ctrl-z` sequence. This command will stop the current job, allowing it to be resumed intact at a later time. Typing `ctrl-d` may be tried if the machine appears stuck and all else fails. This is the EOF (End Of File) sequence. On some machines, using this command will log you out, although a command may be added to your `.login` file to prevent this. Generally, the command `ctrl-l` will redraw the screen. If this does not work on your machine, you may set up an alias in your `.login` file that will enable it. If you are using a SUN with an X window system, there is an option in the root window mouse menu that will "repaint" (X10) or "refresh" (X11) your screen.

In some cases, you may encounter a situation in which none of the above commands work. If your machine is entirely frozen, a system-wide problem may be at fault, and you should try again later. If you are using an individual SUN workstation, a *reboot* operation may be necessary to restart the computer from scratch (see Section 3.5).

## 2. GETTING STARTED

### 2.1 Logging-In to a SUN Workstation

When you approach an individual SUN workstation, you should see the `login:` prompt on the screen. If this is not the case, or if there is a window system currently running, it is likely that another user is logged in to the machine. It is generally possible to use such a machine without re-logging in. Use the `cd` command (described in Section 5.4) to change to your home directory. You should be able to access your files even if the machine is logged in under another username. If you wish to log out the current user and log in as yourself, see Section 3.6 for a description of the logout procedure. Once the previous user is logged out, the `login:` prompt should reappear. At this point, type `[username] <CR>`. Most of the individual workstations will not request a password when you log in. If one is required, you will see the `Password:` prompt after you have entered your username; type `[password] <CR>` following this prompt.

### 2.2 Logging-In to a SUN from a Non-Dedicated Terminal

It is also possible to access the SUN system from a non-dedicated terminal such as an Ann Arbor; these terminals are networked through the two multi-user SUNs. When you approach the terminal, you should see a notice showing the name of the SUN (`wheat-chex` or `rice-chex`) and the `login:` prompt. Type `[username] <CR>`. A password should not be necessary; if one is requested, type `[password] <CR>`.

### 2.3 Accessing a SUN Using `supdup` and `telnet`

If you do not have an individual workstation or a terminal that is networked to the SUN servers, you may access the SUN system from another machine using the `supdup` or `telnet` protocols. While logged into the other machine, type `supdup [sun-name].ai.mit.edu` or `telnet [sun-name].ai.mit.edu`. Replace `[sun-name]` with the name of one of the individual workstations or multi-user SUNs, but not a file server. When you see the `login:` prompt, type `[username] <CR>`. It may be necessary to use a password for a `supdup` connection even if you do not normally need one when logging into a workstation directly. You may also be asked to provide a parameter for the terminal you are using (e.g., `aaa`, `amb`, `heath`). If the screen display is not working correctly, enter the command `unsetenv DISPLAY <CR>` (UNSET ENVIRONMENT) to negate your current display environment.

### 2.4 Preliminaries

Once you have logged into a SUN using one of the above methods, you will see a notice on your screen stating what UNIX release is running, the date and time, and current machine-related messages. You will also receive system-wide mail messages (“bboard”) here. There will be a display showing the content of the message and the prompt `[ynq]`. Type `y` to view the message, `n` to flush

the current message and display the next one, or `q` to save the current and any following messages for later viewing. The format of the message display can be altered by changing the `msgs` line in your `.login` file. After the messages have been read, flushed or postponed, the login process will be complete and you will see the *shell prompt*, indicating that the machine is ready to receive commands.

## 2.5 The Shell

The *shell* is the environment running on your machine that acts as an interface between the UNIX operating system and the user. The shell interprets the commands you enter and invokes the various programs. The C Shell is the basic shell on the BSD UNIX systems. There is a file in each user's home directory called `.cshrc` which contains the default shell script and any special customizations you or the system administrator may add. Another available shell is the T Shell. This shell includes extra features such as EMACS-like command line editing. Users have files named `.tcshrc` and `.tcshrcf` which contain T Shell parameters. Each user also has a `.login` file that sets up local parameters (similar to the `LOGIN.COMD` file on TOPS-20), and possibly a `.logout` file as well. It is common practice to set up a file called `.aliases` which contains renamed commands. Many of the UNIX machines at the AI Lab have frequently used TOPS-20 (TWENEX) commands aliased to run as UNIX commands; you may also set up abbreviated aliases to execute long commands. Whenever you log in to a UNIX system, these special files residing in your directory (called *dot files*) will be executed automatically, and the defaults and aliases contained therein will become a part of your environment.

You will know that the shell is ready to receive commands when you see the shell prompt (also called the *system prompt*). On most machines at the Lab, this will be a combination of your username and the machine name, for example `liz@fruit-and-fibre.ai.mit.edu>`. This prompt indicates that you may initiate various jobs or programs. You may abbreviate your machine address or otherwise customize your prompt by altering the `set prompt` option in your `.cshrc` file.

## 2.6 Accessing Other Computers from a SUN

Because the SUN network file servers provide access to the Lab machines, it is no longer possible to access various computers via a concentrator that gives a choice of machines. You can still access some of these computers, as well as many off-site machines, using the `telnet` protocol. While logged into the SUN, type `telnet [machine-name]`. Replace `[machine-name]` with the name of the desired computer, for example `oz.ai.mit.edu` or `hermes.ai.mit.edu`. Use the complete machine address to ensure a proper connection. Once you have entered the `telnet` command, you should see the login prompt specific to the other machine, and can proceed as you normally would when using that machine. The machine name in the top level prompt should help you keep track of which machine you are actually working on at any given time.

### 3. ADMINISTRATIVE PROCESSES

#### 3.1 Checking for Availability

There are a large number of SUN workstations available at the AI Lab. The individual workstations are basically single-user machines, although they can support a few users (with a decrease in speed and efficiency) if those users are running low intensity jobs. It is recommended that you avoid, if possible, using a workstation which is currently in use. The multi-user SUNs should be used for most routine tasks, but not for large, computationally intensive jobs.

To find a free machine for a large job, first log onto one of the multi-user SUNs (**wheat-chex** for the 7th floor; **rice-chex** for the 8th and 9th floors) and issue the command **qfinger -msun-3 <CR>**. This will give you a listing of all the AI Lab SUNs. Select a machine which is "free" or has been "idle" for a considerable length of time, then log off the current machine, and log into the one you have just located.

If **wheat-chex** and **rice-chex** are above capacity and will not allow new users to log in, you may select a SUN name at random in order to log in and check for availability. Most breakfast cereal names are applicable.

Users should not log on to the file servers (designated in the listing by "Sun File-Server" or "root" in the user column of the **qfinger** listing), since this slows down the administrative processes that run on these machines. The file servers are **wheaties**, **alpha-bits**, **grape-nuts** and **trix**.

#### 3.2 Listing Processes and Jobs

There are two ways to keep track of your work in progress. The first is the process listing. To see a list of all currently active processes, type **ps <CR>** (Process Status). The result will be similar to the list below:

```

PID TT STAT  TIME COMMAND
5524 co IW   0:00 xinit x11
5526 co IW   0:00 xtools
5527 co IW   0:01 uwm
5529 co S    0:09 xclock
5534 p0 S    0:11 -csh (tcsh)
5636 p0 TW   0:04 emacs ltr1.tex
5653 p0 T    0:00 mail bfox
5654 p0 T    0:00 ytex ltr1.tex
5655 p0 R    0:00 ps

```

Each process has a unique number, called the *PID* (Process ID). The listing shows you the type of process, its status, the amount of time it has been inactive, and the command that was used to

initiate the process. Processes will include the current shell (`cs`h), and may also include `mail`, GNU `EMACS`, and `YTeX` jobs. If you are using a window system, the window management functions will also appear in this listing (in this case prefixed with `x` for X windows).

Another type of listing is the job listing. To see this, type `jobs <CR>` at the shell prompt. The listing will look like the following:

```
[1]   Stopped   emacs ltri.tex
[2] - Stopped   mail bfox
[3] + Stopped   ytex ltri.tex
```

This listing shows the jobs by number and displays their current status and the command that was issued to invoke the job; shell and window management jobs are not included. The `-` and `+` indicate the most recent and second most recent jobs, respectively. *Stopped jobs* are those that are currently inactive, but which may be resumed intact at any time until you log out. Jobs may be stopped by issuing the command `ctrl-z` while within a program.

### 3.3 Foreground and Background Jobs

In addition to stopping jobs using `ctrl-z`, it is also possible to make a job or process a *background* job. The command `bg %[job-number] <CR>` will put a specified job into the background (deactivate it) and allow you to work on something else. The command `fg %[job-number] <CR>` will return a job to the *foreground*, or reactivate it; the `fg` command may be used to reactivate both backgrounded and stopped jobs. Replace `[job-number]` with a number from the jobs listing, not a PID. The default job for both commands is the most recent job worked on (designated by `+` in the listing).

If you are using a SUN workstation running a window system, you may keep active jobs running in each window. The jobs in the various windows remain “on hold” while you are working in another window. Jobs become active when you move the mouse into the window in which they are running; this allows you to work on numerous tasks simultaneously.

### 3.4 Killing Processes and Jobs

It is advisable to kill any processes or jobs that you are not using in order to conserve system resources. You may kill a process by issuing the command `kill [PID] <CR>`, replacing `[PID]` with the appropriate PID number from the `ps` listing. The computer will pause momentarily, and you will see a message stating that the process has been terminated.

It is also possible to kill the stopped jobs listed in the `jobs <CR>` listing. Again, the command is `kill`. Enter the command `kill %[job-number] <CR>` (in some cases it may be necessary to include the option `-9` after the `kill` command in order to kill a stopped job).

If you are using a multi-user SUN from a non-dedicated terminal, you should kill all your processes and jobs before logging out. To kill all jobs, enter the command `kill 0 <CR>`. You can also kill all stopped jobs automatically by issuing the `logout <CR>` command twice in succession.

If you are using a workstation that is running a window system and is logged in continuously, kill any `mail`, `GNU EMACS`, `YTeX` and similar jobs when you end your work session. Do not kill your shell processes or window management jobs (`-csh`, `xinit`, `xtools`, `uwm`, etc.), since this will destroy your environment and window set-up.

### 3.5 Logging Out

To log out of a SUN accessed from a non-dedicated terminal, simply type `logout <CR>` at the shell prompt. If you have jobs currently in progress, you will see the message: **There are stopped jobs**. If this occurs, you may either kill the jobs explicitly as described in Section 3.4 above or enter the `logout <CR>` command a second time; this will kill any outstanding jobs and log you out. Use the same procedure if you are using a SUN via a `supdup` or `telnet` link; doing so will log you out of the SUN, but you will remain logged into the machine from which you initiated the connection.

If you are using a SUN workstation, it will be necessary to enter the *login shell* in order to issue the `logout <CR>` command. If no window system is currently running, you are in the logout shell, and the `logout <CR>` command will work directly. The process is a bit more complex if a window system is running. If there is a *console window* on the screen, move the mouse to that window and issue the `logout <CR>` command. If using an X10 window system, press the middle mouse button from within the *root window* to display the root menu, move the mouse to the "quit" option, and click the middle button; if using an X11 window system, the equivalent menu option is "restart". Doing this will bring you out of the window system and back to the login shell. Enter `logout <CR>` at this point to complete your session.

If you use a "private" workstation, you may leave the machine logged in continuously; if you use a "public" workstation or multi-user SUN, log yourself out when you have completed your work in order to free up computational resources for other users.

### 3.6 The Reboot Operation

Sometimes it will be necessary, in the case of a system wide failure or a serious workstation problem, to *reboot* your machine, or resart it from scratch. Occasionally a reboot of all the machines on the network will take place, initiated by one of the system operators. Under no circumstances should a user attempt to reboot a file server or one of the multi-user SUNs. If you are working on an individual workstation, use the command `f <CR>` to see if any other users are currently working on the same machine (unless, of course, you are in a situation where you cannot interact with the computer at all due to a freeze-up); warn any other users before you initiate a reboot. Press the keys `<L1>` (on the left-hand keypad) and `a` simultaneously to take down the machine. You will then see a `>` prompt. Type `b` here, and wait for the computer to re-initialize all systems and re-mount

all the file servers and peripherals. When this process is finished, the `login:` prompt will appear. Complete the login procedure as described in Section 2.1. Whenever possible, you should enter the login shell and log out of the machine as described above rather than rebooting while processes are running; this is much kinder to the machine.



## 4. SOME BASIC OPERATIONS

### 4.1 Getting On-Line Help

Numerous on-line help features are available on the UNIX systems at the AI Lab. The command `man command-name <CR>` will give you an on-screen printout of the pages of the official UNIX manual that describe the specified command. Unfortunately, you must know the exact command name in order to utilize `man`. Alternatively, you may use the `-k` (Keyword) option on the `man` command or the `apropos` command. These will give you all available information on a requested topic. For example, typing `man -k dir <CR>` or `apropos dir <CR>` will give you a screen full of commands relating to directories. You can then select the command you (think you) want, and use the standard `man` command to find out more about it.

Many programs, such as GNU EMACS and the Mail program have their own built-in help features and tutorials. Often typing `?<CR>` or `help<CR>` while within one of these programs will show you what kind of on-line documentation is available and how to get it. This type of help will be described in more detail in the sections that cover the individual programs.

### 4.2 System Security

Due to the academic environment at the AI Lab, it is general practice to keep the computer system open and accessible. It is, however, possible to make your files more secure if you are dealing with sensitive material.

The first level of system security is the user password. Although you generally do not need a password to log into an individual SUN workstation, you may need one to access the SUN system using `supdup` or `telnet`. You will choose or be assigned a password when your account is set up; subsequently, you can change your password at any time by using the command `passwd <CR>`. Enter this command, and you will be prompted for your old password. After entering it, you will receive a prompt asking for your new password; you will be asked to enter this twice to avoid typing errors since the password will not echo on your screen.

Further security may be obtained by altering the *access masks* or permissions on your files and directories. There is an array of permission modes, made up of three user access levels (owner, group and all users) and three capabilities (read, write and execute). These file permissions can be configured as you wish. The procedure for changing protections, including a mode number translation diagram, is presented in Section 5.8.

### 4.3 Identifying Yourself and Other Users

#### 4.3.1 Using `watson`

When you first log in as a new user, you should run the `watson` program (also known as `inquire`) and answer the questions it asks you. This will provide the system with some basic information about

yourself which will be useful to other users. To run this program, type `watson <CR>` at the shell prompt. The program is easy to use, and it provides fairly complete instructions. Start by entering your username, then follow the directions as you go along. The program will ask for information such as your MIT address and phone extension, birthday, supervisor, and projects you are working on. The `Remarks` entry allows you to enter a personal comment or witticism. If you get stuck, enter `?<CR>` to see a list of possible options. If you enter `all <CR>`, you will be asked to supply information for every entry (although you may skip those you do not wish to answer by typing `<CR>`). When finished, type `done <CR>`, and your entry will be updated.

### 4.3.2 The `finger` and `whois` Commands

To see the results of your `watson` session, type `finger [your-username] <CR>`, or simply `f [your-username] <CR>`, at the system prompt. This command will list your personal information, as well as your terminal location and what processes you are currently running. You may also use the `finger` command with another person's username to get the same information about that user. On most of the AI Lab machines, the TOPS-20 command `whois` has been aliased to the `finger` command and will work the same way.

Typing the commands `finger <CR>` or `whois <CR>` without a username will show you a list of the users currently using the machine you are working on. To see a listing of the users of all the AI Lab SUNs, enter the command `qfinger -msun-3 <CR>`. The resulting listing will show which machines are currently in use (including amount of idle time), which are "free", and which are "not responding" (or down). The file servers can be recognized by the designation "Sun File Server" or "root" in the user column. To get a machine listing that also includes the Lisp Machines and the LCS computers, type `qfinger <CR>`.

### 4.4 Your History

When you use the UNIX operating system, a file is created which contains the commands you enter in the course of your work session. This information is held in a file in your home directory called `.history`. This feature is useful if, for example, you get an unexpected result and wish to see what caused it. Use the command `h <CR>` to list your most recent command history. You can also use the history file to re-execute commands without re-typing them. Use the `set history` option in your `.cshrc` file to customize the number of lines saved.

### 4.5 Date, Time and Calendar Display

You can display the date and time on your screen by typing `date <CR>` at the shell prompt. The clock uses twenty-four hour time. If you are using an X window system, it is possible to set up a clock icon on your screen that displays the time continuously. To do so, type `xclock -t <CR>` at the shell prompt, or add this command to your window system set-up file (`.xinit`).

UNIX also has a program that will display the calendar for a given month. Enter the command `cal [month] [year] <CR>`. Replace `[month]` with a number between 1 and 12 corresponding to the desired month; replace `[year]` with either all four digits (i.e., 1988) or the last two digits (i.e., 88) of the desired year. The `cal` command is distinct from another command, `calendar`, which can be used as a reminder for important dates and appointments. Consult the full UNIX documentation or type `man calendar <CR>` for details on how to use this.

#### 4.6 Calculator

There is a program available on UNIX systems that allows you to use your computer as a simple calculator. The command to invoke this program is `bc <CR>` (Basic Calculator). When this command is entered, your cursor will move to the next line, and you may begin entering expressions to be evaluated using the four basic mathematical operators (+, -, \*, /). Type one complete expression per line; entering `<CR>` will cause the answer to be displayed on the following line. When using the / operator for division, the remainder will not be included in the result. Expressions should be in the common form (i.e., `3+4`), not prefixed by the operator as in Lisp. If you are using an X window system, you may wish to set up a small window that runs the `bc` program continuously.

## 5. DIRECTORIES AND FILES

### 5.1 The UNIX Directory and File Structure

One of the great advantages of the UNIX operating system is its nested directory structure and the ease with which one can create and manipulate subdirectories. UNIX directories are arranged in the form of a tree. At the base of the system-wide directory structure is a directory called */root* (or simply */*). Subsidiary to the *root directory* are the user directories and directories that contain public files.

When you log in to a UNIX machine, you will be in your *home directory*; the name of this directory will be the same as your username. This directory is the top level of your personal file structure. All of the files you create will be stored in your home directory or in one of its subsidiary subdirectories.

On the AI Lab SUN system, users' files reside in one partition of the disks on the file servers (*/wh* on *wheaties*, */ab* on *alpha-bits*, */gn* on *grape-nuts*, and */tx* on *trix*) There are copies of the public files residing on each server subsidiary to the */usr* directory. Public directories include */bin*, which contains programs to execute the various commands; */etc*, which contains miscellaneous files such as machine lists; and *games*, which contains public game programs.

Each individual workstation is served by one file server, and draws its public and utility files from that source. This server may or may not be the one on which the user's personal files are stored. If a certain file server is down, all the workstations that are served by that machine will be inoperable. If the machine on which you store your files is down, retrieval and storage will not be possible, although you may continue to perform other operations from your workstation. Ask a system expert to determine which file server stores your personal files and which one serves the workstation you use.

### 5.2 Directories and Pathnames

UNIX files may have *pathnames* of up to 1024 characters. A pathname can be thought of as the "address" of a file. Personal file pathnames are composed of the designation */* for the system-wide root directory, the file server partition (*wh/*, *ab/*, *gn/* or *tx/*), the user's home directory (*fred/*), the names of any applicable nested subdirectories (*text/letters/*), and the specific file name (*bob.ltr*). A typical complete pathname would be: */wh/fred/text/ltrs/bob.ltr*. The pathname allows the machine to search progressively narrower spaces until it locates the specified file.

In general, the entire pathname must be used to access a file outside of your own directory structure. If you are unsure which file server another user's files reside on, you can access the files by replacing the file server specification with a tilde, using a pathname of the form: *~fred/text/ltrs/bob.ltr*. The *~* tells the system to find the file wherever it may be located.

You can access files residing in the directory or subdirectory in which you are currently working by specifying the file name without a path (i.e., `bob.ltr`). Files within subsidiary subdirectories may be accessed by specifying the appropriate nested subdirectory path plus the file name (i.e., `text/ltrs/bob.ltr`).

### 5.3 Identifying Your Current Directory

Use the command `pwd <CR>` (Print Working Directory) at any level to see which directory you are currently working in. This is quite useful in helping you avoid getting “lost” when you are using multiple nested subdirectories or working with other users’ directories.

### 5.4 Changing Directories

The command `cd <CR>` (Change Directory) may be used at any level to change your current working directory. To enter another user’s directory, you must use the full directory pathname: `cd /wh/pat/drafts/ltrs`. Do not include a specific file name as part of the pathname when you are changing directories.

Within your home directory, use the command `cd subdirectory-name <CR>` to enter a subdirectory listed at the top level. Within each subsequent level, typing `cd subdirectory-name <CR>` will allow you to enter subdirectories at that level. You may also use the `cd` command with a nested directory path (for example, `cd drafts/ltrs/old`) to avoid stepping through each level explicitly.

Using the `cd <CR>` command by itself with no associated directory name will bring you back to the top level of your home directory. The command `cd .. <CR>` will bring you progressively up to the next higher level from your current working directory. You may use this command to access directories on levels higher than your home directory.

### 5.5 Creating and Removing Directories

Within UNIX it is possible to have an arbitrary number of nested subdirectories. The subdirectory creation and manipulation capabilities provided by UNIX are far more intuitive and easy to use than the TOPS-20 BUILD program, and users are encouraged to organize their files in trees of subdirectories.

To create a new directory, use the command `mkdir` (MaKe DIRectory). Type `mkdir directory-name <CR>` from either your home directory or an existing subdirectory to create a new directory or subdirectory at that level. Once a subdirectory is created, you can enter it using the `cd` command as described above. Within the new subdirectory, you can again use the `mkdir` command to create a sub-subdirectory, *ad infinitum*. You may also create a subdirectory at a deeper level by giving a nested directory path (e.g., `mkdir /drafts/ltr/old`).

The command to remove a directory is `rmdir` (ReMove DIRectory). Enter the command `rmdir directory-name <CR>` at any level to remove a directory or subdirectory at that level. You may

remove a subdirectory from a higher level by using a nested directory path. Before you attempt to remove a directory or subdirectory you must remove all the individual files contained therein. You may remove all such files at once by typing `rm * <CR>` (file removal is discussed in Section 6.3). There is also a recursive remove operation that can be used to eliminate a directory as well as all the files it contains; the command is `rm -r [dirname] <CR>` (ReMove -Recursive).

## 5.6 Listing Files

There are a variety of ways to list files within UNIX. The most basic is the `ls <CR>` (LiSt) command. This command will show you a simple alphabetical listing of the files and directories at the top level of your home directory, as shown below:

```
admin      drafts  list.todo  personal
biol-cyber lisp    mail       ytex.local
```

Another version of this command is `ls -F <CR>`. This command produces a similar listing, but the directories are marked with a `/`, as shown below. Many of the UNIX systems at the Lab have the `ls` command aliased to execute `ls -F`, since this is generally a more useful command.

```
admin/      drafts/  list.todo  personal/
biol-cyber/ lisp/    mail/      ytex.local
```

The listings given by the two commands above are not complete. They only show those files and directories created by the user, not the hidden files. To see a more complete file listing, use the command `ls -a <CR>` (LiSt -All). The listing will look like the following:

```
./          .defaults .msgsrc    admin/     list.todo
../         .history  .tcshrc   biol-cyber/ mail/
aliases    .login    .uwmrc     drafts/    personal/
.cshrc     .logout   .xtools   lisp/      ytex.local
```

This listing includes the normally invisible *dot files*, which contain commands used to run the machine, establish the environment, and perform “administrative” functions. The `./` and `../` entries are always present, and refer to the current working directory and its parent directory. The `.login` and `.logout` files and files to run the shells (`.cshrc` and `.tcshrc`) are described in Section 2.5. Some of the dot files contain special programs and commands to run the X window system (`.uwmrc`, `.xtools`). The files and directories created by the user are shown at the end of the `ls -a` listing.

If you wish to obtain more information about your user created files and directories, use the command `ls -l <CR>` (LiSt -Long). The listing will be similar to the following;

```

total 32
drwxrwxr-x  2 liz      512 Feb 29 18:43 admin
drwxrwxr-x  2 liz      532 Feb 17 12:57 biol-cyber
drwxrwxr-x  5 liz      512 Jan 21 15:33 drafts
drwxrwxr-x  2 liz      512 Feb 24 10:28 lisp
-rw-rw-r--  1 liz      277 Dec 14 13:40 list.todo
drwxrwxr-x  3 liz      512 Mar  2 10:42 mail
drwxrwxr-x  2 liz      512 Feb 24 11:47 personal
-rw-rw-r--  1 liz     4681 Dec  7 16:57 ytex.local

```

The first line of this listing shows the total number of files in your personal directory structure. This will not equal the actual number of entries in the listing, since each directory may contain multiple files. Each entry has its own line. A `d` at the beginning of a line indicates that the entry is a directory. The `rw` fields are *access mask* codes; they indicate who has permission to do what with the directories and files: possible options are read (`r`), write (`w`), and execute (`x`). The first field denotes permissions for the owner of the entry, the second for groups, and the third for all users. A missing letter in one of these fields indicates that a permission is withheld. Access masks are in described more fully in Section 5.8. The number following the permission fields indicates the number of subdirectories the entry contains (if it is a directory). This number is 2 if there are no subdirectories. Individual files are designated by 1. To determine the actual number of subdirectories in an entry, subtract two from the number shown. The next field is the username of the owner of the entry. The field after the username shows the size of the entry (directory sizes are small; the directory size is not equal to the sum of the sizes of the files it contains). Next is the date and time the entry was last modified. The final field is the name of the entry.

It is also possible to get a listing arranged by time, with the most recently modified entry listed first. To do so, use the command `ls -t <CR>` (LiSt -Time).

These listing commands may be used in combination. For example, the command `ls -Fla` will give a detailed (long) listing that includes dot files.

## 5.7 Types of Files

File names may be chosen to convey certain information to the user. Parts of file names are separated by dots (e.g., `new.tex`). The first part of the file name may be compounded using hyphens (e.g., `first-new-letter.tex`). Suffixes to the file name may be used to indicate the contents of the file. Some examples are shown on the next page:

```

#smith.ltr#      cfg-parse.lsp.~2~  smith.log  smith.ltr~~
cfg-parse.lsp    vision-draft.ytex  smith.ltr  ytex.local
cfg-parse.lsp.~1~  smith.dvi        smith.ltr~

```

Files containing text processing commands should end in the suffix `.tex`, `.ytex`, or `.latex`. This will allow the system to select the correct file when a text processing program is run. Alternatively, you may select file suffixes that are mnemonic, for example, `.ltr` for files that contain letters. When the text processing programs run, they will produce files with the suffix `.dvi`. These files will contain special commands to be interpreted by the printer. Such programs will also produce files ending in `.log` which contain a transcript of the running of the text processor program (equivalent to the `.LST` file on TOPS-20). Files that contain Lisp code have the suffix `.lsp`. This suffix will indicate to GNU EMACS that Lisp Mode should be in effect, enabling proper indentation as well as code execution from within the editor.

UNIX uses special characters to designate backup files. On most UNIX systems, only one backup version of each file is saved. The backup version is indicated by a single tilde following the file name. Each time you make changes to a file, the new version is saved in `filename` and the previous version is saved in `filename~`. If you edit the backup version, the result will be a filename followed by two tildes (e.g., `smith.ltr~~`) indicating a backup of a backup. It is possible to configure your editor to save multiple versions of a file. Files saved when this feature is in effect have a suffix that contains a number between two tildes (e.g., `cfg-parse.lsp.~2~`). File entries at the beginning of the listing surrounded by # signs (e.g., `#smith.ltr#`) are backup files produced by the autosave feature, which automatically saves work in progress after a certain number of keystrokes. Once you have made all changes and saved the final version of a file, autosave files may be deleted. When editing, processing, or printing a file, the latest version of the file is assumed unless otherwise specified.


## 5.8 Setting Access Masks - Permission Options

As noted in the section above on file listing, each directory and file entry has a range of *access masks* associated with it, as indicated by the following schematic:

```

drwxrwxr-x  2 liz          512 Feb 24 11:47 lisp
-rw-rw-r--  1 liz          4681 Dec  7 16:57 ytex.local

```



```

  o  g  a
  w  r  l
  n  o  l
  e  u
  r  p

```



Capabilities include the permission to read a file (**r**); write, or modify, a file (**w**); and execute a directory (**x**). Execution applies only to directories, and refers to permission to search the given directory (for example, to perform an `ls` listing). These permissions may extend to only the owner of a file or directory, or they may be granted to a group of users or to all users of the system. The owner of a file or directory is generally the user that created it, but this parameter may be changed with the command `chown` (CHange OWNership). Only the *superuser* may change ownership. Groups may be configured to include any set of users that will have access to a file or directory. All users belong to some group, and files are marked accordingly. The command to change a group is `chgrp` (CHangeGRouP). Type `chgrp [group-name] [file-name] <CR>`. Group affiliation can only be changed by the owner of the file or the superuser.

Each file owner may change the access masks associated with his files. The command to do this is `chmod` (CHange MODE). This command may be used with either character codes or an octal representation of the codes. Command usage is `chmod [permission] [filename] <CR>`. Character codes have three fields, as listed in the table below:

	who	change	capability
	-----	-----	-----
<b>u</b>	<b>owner</b>	<b>- remove</b>	<b>r read</b>
<b>g</b>	<b>group</b>	<b>+ add</b>	<b>w write</b>
<b>a</b>	<b>all users</b>	<b>= assign</b>	<b>x execute</b>

The `[permission]` field in the `chmod` command should be replaced by a three character code containing one option from each of the above columns in the form: `[who][change][capability]`. For example, `a-x` means remove execute permission from all users; `g+w` means add write permission to the group.

The octal option allows you to set the same permissions using numerical codes. The `[permission]` field of the `chmod` command should be replaced by a number representing the desired access configuration, as indicated in the chart below:

	read	write	execute
<b>owner</b>	<b>400</b>	<b>200</b>	<b>100</b>
<b>group</b>	<b>40</b>	<b>20</b>	<b>10</b>
<b>all users</b>	<b>4</b>	<b>2</b>	<b>1</b>

To determine the correct code, add the numbers of all the permissions you wish to assign; for example, the code `666` assigns read and write permission to the owner, the group and all users, but withholds execution permission.

## 6. MANIPULATING FILES

### 6.1 Creating Files

There are a variety of ways to create files within UNIX. The most basic method is the `cat` command. To use this command, type `cat > new-file-name <CR>` at the shell prompt. After entering this command, your cursor will appear on the following line. At this point, begin typing in any text that you wish to put into the file. This text can be arbitrarily long, and may include text processing commands, programming code, etc. To save the text and exit `cat`, type `ctrl-d` on a line by itself (this is the EOF, or End Of File character). You will see when you do a listing that the file now exists. Note that the `>` symbol in the `cat` command indicates that whatever you type should be redirected to the file you specify.

The second way to create a file is with the `touch` command. To use this command, type `touch new-file-name <CR>` at the shell prompt. You will not be asked to enter text at this point, but will instead be returned to the prompt. This is an example of UNIX's silent command operation in action. When you list your files, you will see that the file has been created; there will be a 0 in the size field of a `ls -l` listing, indicating that the file is empty. The `touch` command may also be used to update the modification time of a file without actually altering it.

You may also create files using the GNU EMACS editor as you would on a TOPS-20 system. There are two ways to create a file using GNU EMACS. The first is to type `emacs filename <CR>` at the shell prompt. This will put you into the GNU EMACS program, and you will be within the specified file. Here you can enter any text you wish, using the movement and manipulation commands described in Section 10.6. Alternatively, you can enter the GNU EMACS program by simply typing `emacs <CR>` at the shell prompt without a filename. If you are using an X window system, you may invoke the GNU EMACS program by clicking your mouse on the "emacs" option in the root menu. When you do this, or when you type the `emacs <CR>` command from the main window, a "rubberband" window will appear on your screen which you may resize and move by dragging the mouse. The GNU EMACS job will run in this new window.

If you enter GNU EMACS without a filename, you will see a screen displaying basic editor information. You may then use the `ctrl-x ctrl-f` sequence to find the file you want. When you use the find sequence, a partial pathname will appear at the bottom of your screen or editing window indicating the current directory path. If you are seeking a file in this directory, simply add the filename to the end of the path and press `<CR>`. If you wish to locate a file in a different directory, use the `<del>` key to erase the given pathname and replace it with the one you want. If you try to find a file that does not exist, a new file will be created.

### 6.2 Saving Files

To save a file, type `ctrl-x ctrl-s`. If you are not using a window system, type `ctrl-z` to stop the job and exit the editor. If you are using X windows, moving the mouse out of the window will

deactivate the job. Resume the editing job either by using the `fg` command (see Section 3.3) or by moving the mouse back into the editing window. To kill the job (as well as the editing window) and exit the editor, type `ctrl-x ctrl-c`.

### 6.3 Viewing Files

There are a variety of ways to view files within UNIX. The first is to enter GNU EMACS and use the `ctrl-x ctrl-f` sequence to find the desired file. To view a file without entering GNU EMACS, you can use the `cat` command. This is the same command used in the previous section to create new files; if the file already exists, however, the `cat` command will allow you to look at it. Type `cat filename <CR>` at the shell prompt, and the file will be displayed on your screen.

It may happen that your file is too long to fit on one screen; the `cat` command will cause the file to scroll by rapidly without stopping. This can be controlled by using the commands `ctrl-s` to stop scrolling and `ctrl-q` to resume it, but an easier way to view long files is with the `more` command. Type `more filename <CR>` at the shell prompt. This will cause the file to be printed on your screen one page at a time, with the message: `--More (n%)--` printed at the bottom. At this point you have three options: press the `<space bar>` to reveal the next page of text; press `<CR>` to reveal the next line of text; or type `q` to leave `more` file and return to the shell prompt. There are also commands called `head` and `tail` which will allow you to view the beginning and the end of a file, respectively. Type `head -[n] filename <CR>` or `tail -[n] filename <CR>`, replacing `[n]` with the number of lines you wish to see. For example, `tail -20 filename <CR>` will show you the last twenty lines of the file.

The final method of viewing a file is the `pr` (PRint) command. This command will display the file on your screen, with headers showing the date, time, filename and page number inserted at the top of each page. Like `cat`, the `pr` command causes the file to scroll by rapidly on your screen. The `pr` command is more useful for redirecting or piping output to other files (see Section 8). The `pr` command may also be used with options. For example, the command `pr -3 filename <CR>` will cause the file to be printed in three columns. Note that the `pr` command is not useful for `.dvi` previewing, since the page breaks will not coincide with those produced by text formatting programs such as `yTeX`. For more information about `pr`, consult the standard UNIX documentation or type `man pr <CR>`.

### 6.4 Removing Files

The command to remove files within UNIX is `rm` (ReMove). Unlike TOPS-20 systems, there is no "soft delete" feature that allows files to be marked for deletion and expunged at a later time. Once the `rm` command is executed, the file is gone. To use the removal command, type `rm filename <CR>`. The `rm` command can be augmented with the `-i` option for safety purposes. When you enter `rm -i filename <CR>`, the system will ask you whether you really want to remove the file; type `y <CR>` if you do, or just `<CR>` if you do not. The `rm` command may be aliased to `rm -i` in your `.cshrc` or `.aliases` file so that you are always queried before a file is removed. The `rm` command

can be used with the wildcard symbol `*` to remove all files that contain a given string. When you use `rm -i` for a wildcard deletion, you will be queried about each matching file in turn.

In order to conserve memory space on the system, it is good practice to clean up your directories periodically to remove old versions and files you no longer need. At present, there is no provision for “migrating,” or taking files offline that have not been accessed for a certain length of time. Until such a feature is implemented, it is the responsibility of the users to keep the file structure as clean as possible.

In some cases, it may be possible to recover files that have been removed by retrieving them from back-up tapes. The SUN system is generally backed up at least weekly. There are presently no commands available for remote retrieval of archived files (like the `RETRIEVE` and `GROVEL` commands on TOPS-20); such operations must be done directly from the tape drive console. Check with a system administrator or a technical support person if you must retrieve files from tapes.

## 6.5 Concatenating Files

The `cat` command and the redirect symbol (`>`) which were described briefly in Section 6.1 above can also be used to combine files (somewhat like the `APPEND` command on TOPS-20). The command name `cat` can be thought of as an abbreviation for “conCATenate”. To combine two files, type the command `cat file1 file2 > new-file <CR>` at the shell prompt. This tells the computer to take the contents of `file1` and `file2` and redirect them to `new-file`. If you omit the `>` symbol and the new filename, `file1` and `file2` will be printed one after the other on your screen. The files will be appended in the order they are given (in the above example, `new-file` will consist of the contents of `file1` followed by the contents of `file2`). It is possible to concatenate any number of files with one `cat` command; type `cat f-1 f-2 f-3...f-n > f-new <CR>`. To ensure that the operation was successful, check your file listing using `ls -l` and verify that the sizes of the component files add up to the size of the new file.

## 6.6 Moving Files

There is a command within UNIX that allows you to move files, in effect renaming them. The command is `mv` (MoVe). Typing `mv file1 file2 <CR>` will tell the system to put the contents of `file1` into `file2`. WARNING: If a file named `file2` already exists, it will be written over by the contents of `file1`. When you check your file listing, you will see that `file2` is listed, but not `file1`, which will have been renamed.

## 6.7 Copying Files

It is possible to copy files using the `cat` command and the redirect symbol; the command `cat file1 > file2 <CR>` will redirect the contents of `file1` to `file2`, effectively creating a copy. However, the UNIX system provides a special local copying function that can be used instead, the `cp` (CoPy) command. Type `cp file1 file2 <CR>`. This will cause the contents of `file1` to be

copied to `file2` (remaining in `file1` as well). This command can be used to create extra versions of a file to compensate for the single backup version. You must use a “from” filename and a “to” filename with the `cp` command (unlike on TOPS-20, where the `COPY` command will create a new version number of the file if a second file name is omitted). The `cp` command may be used to copy any files that are part of the SUN file structure. Use a full pathname to copy to or from a directory other than your current working directory

### 6.8 Remote Copying from Other UNIX Systems

UNIX systems have a useful feature that allows you to easily copy files to and from other machines running UNIX. This is the `rcp` (Remote CoPy) command. Files may be copied to or from your machine if you know the name of the other machine and the name of the file on that machine. At the shell prompt, type `rcp other-machine-name:filename new-filename <CR>` to copy from another machine to your machine; or `rcp filename other-machine-name:new-filename <CR>` to copy from your machine to another. It is not necessary to include the name of your machine, since this is assumed by default. If you are copying a file from your working directory, the entire pathname is not necessary; however, you should know the entire pathname of a file on the other machine in order to copy it. Care should be taken that you do not give the “to” file a name that may already exist on the other machine, since any file so named will be written over by the new file.

### 6.9 Linking Files

In addition to copying, it is also possible to use the `ln` (LiNk) command. Linking allows you to refer to files by different names in different directories without actually having separate copies of the file. In this way, you can access a file in different directories, but do not need to change multiple files when an alteration is necessary, since changing one of the linked files is sufficient. To use this command, type `ln file-name alternate-name <CR>`. After the linking operation is completed, either `file-name` or `alternate-name` may be used to refer to the file.

## 7. TRANSFERRING FILES

### 7.1 File Transfer Program Usage

There are programs available on UNIX, known as *file transfer programs*, that may be used to transfer files over the network to other machines (which may or may not be running UNIX). When a file transfer program is used, a copy of the file is transferred; the file will also remain on the computer from which it originated.

The most basic such program is called `ftp` (File Transfer Program). This program may be used to transfer files from one computer to another; these computers do not need to be at the same site or on the same local network. To use this program, type the command `ftp <CR>` to invoke the program. To initiate a connection, type `open other-computer-name <CR>` at the `ftp>` prompt. Once the link is initiated, supply the program with the information that it requests. This information will include a "from" file and a "to" file. On certain systems, it will be necessary to have a login name and a password on the other machine in order to complete the connection. Type `quit <CR>` to exit and kill the `ftp` program.

### 7.2 OZ <---> SUN Transfers

There is a special version of the `ftp` program currently available for use at the AI Lab to transfer files from the OZ computer to the SUN system. This program currently routes transfers through a LCS computer named MC. The routing procedure is likely to change as the new AI Lab system is consolidated. Operation of this program is shown below.

```
@ftp
Chaosnet/TCP gateway FTP, Preliminary bare-bones version 0.13
Use ^G to abort. Report bugs to BUG-FTP@MIT-OZ.
FTP>connect fruit-and-fibre.ai.mit.edu
<< fruit-and-fibre.ai.mit.edu FTP server (Version 4.7 Sun Sep 14 12:44:57 PDT 19!
986) ready.
Assuming ASCII transfers, do SET MODE to change transfer mode.
MC|FRUIT-AND-FIBRE.AI.MIT.EDU>login sun-username
Password for logging in as liz to FRUIT-AND-FIBRE.AI.MIT.EDU: sun-password
<< User liz logged in.
MC|FRUIT-AND-FIBRE.AI.MIT.EDU>send OZ-FILE.TXT.3
To foreign file: /wh/username/sun-file.txt
<< Opening data connection for /wh/username/sun-file.txt (10.3.0.44,3267).
<< Transfer complete.
MC|FRUIT-AND-FIBRE.AI.MIT.EDU>get /wh/username/sun-file.txt
To local file: OZ-FILE.TXT
```

```
<< Opening data connection for OZ-FILE.TXT (10.3.2.44,3267).  
<< Transfer complete.  
MC|FRUIT-AND-FIBRE.AI.MIT.EDU>exit
```

Note that the command to initiate the link is `connect`, not `open` as it usually is for `ftp`. To ensure a proper link, use the full machine name of an individual SUN workstation or multi-user SUN (not a file server). Unlike some other `ftp` implementations, you will not be given a login prompt, but must login by typing `login username <CR>` at the first `MC>` prompt using your SUN username. You will also be prompted for your SUN password. The file sent from OZ will be put into your home directory on the SUN system, not your current working directory (note that your home directory resides on a file server, not on the machine that is used to do the transfer). You will be notified when the transfer is complete. To leave the `ftp` program, type `exit <CR>`.

### 7.3 Transfers from Reagan

The Lisp Machine file server `Reagan` can be considered as a peripheral addition to the SUN file structure. In order to access files on this machine, it will be necessary to *mount* `Reagan` from your workstation (`Reagan` should not be mounted from a file server). To do this, type the command `mount /b <CR>` at the shell prompt. `/b/` is the UNIX path designation for `Reagan`; this should precede any file name on `Reagan` (e.g., `/b/liz/drafts/file.txt`) Once `Reagan` is mounted, you can use the basic UNIX commands such as `ls` and `cp` to operate on `Reagan` files; they will be treated as part of the SUN file structure. You may also edit files on `Reagan` from a SUN; use the find sequence `ctrl-x ctrl-f` with the complete `/b/` pathname. GNU EMACS will work the same way it does normally, although editing and saving will be slower since the changes are being made to a remote machine. When you have finished using files on `Reagan`, type the command `umount /b <CR>` (UnMOUNT) to maximize efficiency.

## 8. REDIRECTION AND PIPING

### 8.1 Redirection

You have seen UNIX's *redirection* feature in action in some of the previous sections. This is a versatile mechanism that has a variety of uses. The redirect symbol `>` causes whatever is entered before the symbol to be redirected to the location specified after the symbol. The opposite symbol `<` may also be used; redirection will occur in the direction pointed to by the arrowhead. If either the field before or the field after the symbol is left blank, redirection will occur to the terminal or from the keyboard as a default. This feature can be used to create new files as described in Section 6.1. Typing `cat > new-file <CR>` will cause whatever is entered on the keyboard to be redirected to `new-file`. The mechanism may also be used to copy files (Section 6.6). Typing `cat file1 > file2 <CR>` will redirect the contents of `file1` to `file2`. Omitting a field after the symbol will redirect the output to your screen, allowing you to view it. Redirection is also used to concatenate files (Section 6.4); the command `cat file1 file2 file3 > new-file <CR>` will cause the content of `file1`, `file2` and `file3` to be redirected, in that order, to `new-file`.

It is also possible to use this mechanism to redirect the output of commands. For example, typing `ls > file1 <CR>` will redirect the output of the `ls` command to `file1`; `file1` will then contain a listing of your files. You can use the `pr` command in conjunction with the redirect feature to make a new file of text that includes the page headers added by `pr`. For example, typing `pr file1 > file-new <CR>` will cause the text currently in `file1` to be put into `file-new` with the new headers inserted.

You can also use the reverse redirection symbol with the `mail` command (see Section 9.4) to send a message to multiple users. First create the message in a file. Then enter the command `mail username-1 username-2...username-n < message-file <CR>`.

A variant of the redirect feature allows you to redirect new material, outputs, etc. to the bottom of an existing file. The symbol to use is `>>`. For example, typing `ls >> file2` will put a listing of your files at the bottom of `file2` after whatever it currently contains, in contrast to the standard redirection symbol `>` which will cause `file2` to be overwritten by the new material.

### 8.2 Piping

*Piping* is a mechanism of redirecting input and output that allows you to use output from one operation as input to another (somewhat like passing a value to a function). This may be used to pass the results of one command to a second command as input. The piping symbol is the vertical bar (`|`). For example, you may pipe the results of a command that produces a long output through the `more` command so that it will appear one page at a time; the command `ls -la | more` will pass your file listing to `more`, which is useful if you have more than a page's worth of files to list. This feature is used within the `Mail` program (see Section 9.3) so that incoming messages are displayed page by page. In a piping operation such as this, a command like `more`, which receives and modifies



output, is referred to as a *filter*. Pipes may also be used to send output to alternative devices, such as printers. It is possible to use numerous pipes in succession in a single command line. Consult the standard UNIX documentation for more information about pipe usage.

## 9. USING ELECTRONIC MAIL

### 9.1 Mail Basics

The standard electronic mail programs on UNIX are called `mail` and `Mail`. These programs are similar, but `Mail` has more features (it is similar to the `MM` program on TOPS-20). On many of the AI Lab UNIX machines, the command `mail` has been aliased to run the `Mail` program. In addition, there is a program called `rmail` which runs within GNU EMACS and is similar to the `BABYL` program on the TOPS-20. Consult the *Idiot's Guide to OZ* for more information about using `BABYL` and the *GNU Emacs Manual*, by Richard Stallman, © Free Software Foundation (1985, 1986) for information about GNU EMACS `rmail`. UNIX also has its own program called `rmail`. This may be accessed by typing `rmail <CR>` at the shell prompt or by clicking on the "rmail" option in the root mouse menu of an X window system. This program, which is distinct from the GNU EMACS `rmail`, will not be discussed in this guide; type `man rmail <CR>` for more details.

### 9.2 The UNIX Mail Program

The commands `Mail` or `mail` are entered at the shell prompt to invoke the electronic mail program. Within the program, you will have a different prompt and a distinct set of options. The command `mail` is also used with a username at the shell prompt to send messages; this is described in Section 9.4. To avoid confusion in this manual, `Mail` will be used to designate the electronic mail program and `mail` will be used to refer to the mail sending command.

When incoming mail is received, the message `You have mail` will appear on your screen. This message will also appear when you log in if you have received mail since your last session. If you are using X windows, you can set up a mailbox icon that lights up or beeps when mail is received using the command `xbiff & <CR>`.

### 9.3 Reading Your Mail

When you initially type `Mail`, you will enter the `Mail` program, and a numbered list of messages will appear on your screen as shown below. If you have no waiting messages, you will see the notice: `No mail for Username`.

```
Mail version 5.2 6/21/85. Type ? for help.
"/usr/spool/mail/username": 4 messages 3 new 1 unseen
U 1 poggio@oz.ai.mit.edu Tues May 17 16:10 11/401 "Reminder"
>N 2 liz@wheaties.ai.mit.edu Wed May 18 11:40 11/433 "Stereo"
N 3 liz@wheaties.ai.mit.edu Wed May 18 11:41 12/448 "Seminar"
N 4 tlp@oz.ai.mit.edu Wed May 18 13:43 11/433 "Meeting"
&
```

The list shows the current messages, including sender, date and subject. The designation **U** at the beginning of a line indicates that the message was previously received, but has not yet been seen. A **N** indicates that the message is new. The **>** will point to the first message in the read queue. After the message list, you will see the **&** symbol, which is the Mail program prompt. At this point, you have several options to choose from. Typing **?** will list the possibilities.

To view a message on your screen, enter **t [number] <CR>** (Type), **p [number] <CR>** (Print), or simply **[number] <CR>** after the **&** prompt. Replace **[number]** with a message number from the list. When you do this, the specified message will appear. If the message is too long to fit on the screen, you will see the flag **--More(n%)--** at the bottom. Typing **<space bar>** at this point will cause the next page of the message to appear and **<CR>** will display the next line of the message. When you have viewed the entire message, you will be put back at the **&** prompt, or you can type **q** at the pause to return to the prompt without viewing the full message.

At this point, decide what to do with the message you have just seen. Type **d <CR>** (Delete) to remove the message. Enter **s filename <CR>** (Save) to save the message to a specified file; if the file already exists, the new message will be appended at the end, not overwritten. The option **w filename <CR>** (Write) will put the message in the file without the mailer heading. Typing **s <CR>** or **w <CR>** with no filename will save or write the message in a file in your home directory called **mbox**. You may also use the **d <CR>**, **s <CR>** and **w <CR>** commands with message numbers, allowing you to delete, save and write messages without first looking at them. Use the option **+ <CR>** to show the following message and **- <CR>** to show the previous one. You may also simply type **<CR>** to go on to the next message without doing anything with the current one; messages skipped in this manner will be saved in **mbox** automatically when you leave the mail program.

To send a return message to the sender and the cc list, type **r <CR>** (Reply). To send a return message to the sender only, type **replysender <CR>**. After you have entered one of these commands, a heading will appear on your screen; after the heading, type in your reply message, ending it with a **ctrl-d** on a separate line.

To exit the Mail program, type **q <CR>** at the **&** prompt. Any messages that have not been explicitly deleted or saved elsewhere will be put into your **mbox** file. This file may be edited or deleted like any other file. All unseen messages will be stored in a file called **/usr/spool/mail/username**. If mail is received while you are within the Mail program, you will get the message **New mail has arrived** when you exit. It will be necessary to re-enter the mail program to see the newly arrived messages.

The Mail program allows certain options. The **-f** option will allow you to read mail from a specified file instead of from **/usr/spool/mail/username**. For example, **Mail -f mbox <CR>** will read messages from your **mbox** file. The **-r** option will allow you to read your mail in reverse, with the most recently received message presented first.

## 9.4 Sending Mail

To send mail within UNIX, it is not necessary to enter the Mail program. Send mail from the shell prompt by typing `mail [username] <CR>`, replacing `[username]` with the name of the desired recipient. You may also send mail to multiple users, separating the usernames by commas (i.e., `mail diane, smd, dms, liz <CR>`), or to mailing list entities that represent multiple users (i.e., `mail sun-users <CR>`). If you wish to send mail to someone on an AI Lab computer other than the one you are using, you must include the name of the machine (i.e., `mail username@reagan`).

To send mail to users outside the AI Lab, you must include the machine address. The standard machine address on the Internet is of the form: `username@computername.institution.classification`. Classifications include `.edu` for educational institutions and `.com` for commercial entities. To send mail to a user at a machine that is at MIT but not part of the AI Lab network, include the institution field `.mit` (i.e., `mail sds@cogito.mit`). A full machine address must be used to send mail to users beyond the MIT campus (i.e. `mail orville@cvs.rochester.edu`).

When you enter the `mail` command, a name and date heading is prepared automatically. You will see a prompt for `Subject:`, but will not be prompted for your message; simply enter a `<CR>` to move to the line after the subject line and begin typing. To complete and send the message, enter a `<CR>` to move to the next line and type a `ctrl-d` on the new line; the `EOT` (End Of File) designation will appear at this point. Unlike the TOPS-20 mail systems, the UNIX mailer will not tell you that your message has been queued. You will also not be warned immediately if you attempt to send mail to an unknown or malfunctioning host; instead, you will receive a message a short time later from "Mailer" letting you know that the message did not go through.

It is also possible to create a message beforehand in a file and then send the file. Once the message file is created, type `mail username` as described above and enter a `<CR>` to move to the message area. At this point, `~rfilename`. There should be no space between the `~r` and the filename. The system will echo the name of the file to be sent. You must still type `ctrl-d` on a line by itself in order to complete and send the message. Alternatively, you can use the reverse redirection mechanism for this purpose; type `mail user1 user2 user3 < msg-file <CR>`.

If you mistakenly type `ctrl-z` at the end of the message (as you would on TOPS-20), your mail job will be stopped. Type `fg <CR>` to resume the job where you left off.

## 9.5 Other Sending Methods: talk and write

It is possible to send and receive immediate messages to and from other users without using the Mail program or the `mail` command. The options described below will cause messages to appear directly on the recipient's screen; the user will not have to enter the Mail program to read these messages.

There are two similar programs called `talk` and `write` which allow users to carry on two-way "conversations". To invoke these programs, type `talk [username] <CR>` or `write [username]`

<CR>, replacing [username] with the name of the user with whom you wish to converse. If the user is logged on at more than one site, you should also specify which terminal you wish you send messages to (i.e., **talk** [username] [tty20] <CR>). The specified user will receive a notice on his screen letting him know that you wish to converse. The recipient must reply with **talk** [requester's-name] <CR> or **write** [requester's-name] <CR> in order to complete the connection. Once the link is established, messages may be sent back and forth (this may be a bit slow since characters are sent as they are typed). End individual messages with (o) (Over) and the final message with (oo) (Over and Out). To end the connection, each participant must enter a **ctrl-d** on a line by itself.

You may interrupt a **talk** or **write** session in order to perform other tasks. Any line within the **talk** or **write** session that is preceded by an exclamation point (!) will be interpreted as a command to the shell, and will not be sent as part of the message.

It is possible for a user to set a parameter that denies permission to others trying to writing to his screen. The parameter setting command is **mesg**. Use **mesg n** <CR> to revoke write permission, and **mesg y** <CR> to reinstate it. Using the **mesg n** setting will disable **talk** and **write** unless the sender uses the permission override option when invoking these programs (this should only be done by the superuser).

There is also a command that may be used to send brief, urgent "Broadcast Messages" to all users (except those who have their **mesg** parameter set to n); this is similar to the **SHOUT** command on TOPS-20. The command to use is **wall** [filename] <CR> (Write ALL). The [filename] option may be used to send a message that has been previously created in a file, or the option may be omitted and the message entered directly from the terminal.

## 10. EDITING WITH GNU EMACS

### 10.1 Introduction

**GNU EMACS** is an *editor*, a program that is used to enter, move, and manipulate text. It has many features that allow you to do advanced word processing and program editing. In this section, you will learn about the various commands needed to move and rearrange text, how to use **M-x** commands to perform special functions, and how to design keyboard macros to simplify your editing work.

If you are using a non-dedicated terminal to access a SUN, you may have trouble running **GNU EMACS** because the display environment will be incompatible. It may be possible to remedy this by typing the command `unsetenv DISPLAY <CR>` (**UNSET ENVIRONMENT**) at the shell prompt before invoking **GNU EMACS**.

The **GNU EMACS** program consists of many commands, some of which perform quite complex functions. Fortunately, you will only need a small subset of these commands to get started, and you can learn more as you gain experience. For a more detailed treatment of **GNU EMACS**, consult the **GNU Emacs Manual**, by Richard Stallman, © Free Software Foundation (1985, 1986).

It is important to note that on **UNIX** systems, **GNU EMACS** is not the principal editor. The default editors are programs called **ed** or **vi**. These editors will not be discussed here because they are more limited and have less features than **GNU EMACS**. However, it may happen that you accidentally find yourself within one of these editors; this will occur if you type `edit <CR>` or `edit filename <CR>` at the shell prompt as may be done on **TOPS-20** systems. The **ed** editor can be recognized by its `?` prompt. To exit this program, type `q <CR>`. The **vi** editor requires that all commands be preceded by a colon (`:`). To write (save) the file, type `:w <CR>` (this may be done from anywhere within the file). To write and exit, type `:wq <CR>`. To exit without saving type `:q! <CR>`. To learn more about the **ed** and **vi** editors, consult the standard **UNIX** documentation.

### 10.2 On-Line Help

The **GNU EMACS** editor includes extensive on-line documentation. The help character within **GNU EMACS** is `ctrl-h`. When you type `ctrl-h`, you will see a message at the bottom of your screen or editing window asking you to type `?` for further information. Doing so will cause a list of options to appear at the bottom of the screen. If you are unfamiliar with these options, type `ctrl-h` a second time for a display telling you how they can be used. Among the most useful are **A** (**Apropos**), which gives general information about a function or operation; **C**, which briefly describes a command; **K**, which fully describes a command; and **T**, which invokes the **GNU EMACS** tutorial (this is similar to the **TEACH EMACS** tutorial on **TOPS-20**, and will give you fairly explicit instructions as you go along).

You may also use the help character with a specific key sequence to get an explanation of what that sequence does. To exit the help module, type `q` (if at the full screen command description) or `q <CR>` (if at the command line on the bottom of the screen).

### 10.3 Using Buffers

**EMACS** is designed to work with *buffers*. Buffers are unsaved files that contain work in progress. Normally when you are editing, you will be working with only one buffer, but you may create and work with as many buffers as you choose. The contents of a buffer will be lost when you logout (or if the system crashes or is rebooted) unless you explicitly save the buffer as a file using `ctrl-x ctrl-s`. If you are using **GNU EMACS** with X windows, the command `ctrl-x ctrl-c` will query you about each currently active buffer and save the ones you request before killing the **GNU EMACS** program and the window in which it is running. Although it is possible to keep an arbitrary number of buffers active at any time, it is wise to frequently save your buffers as files and remove extra buffers in order to maximize efficiency and minimize damage if the buffers are lost.

Buffers should not be confused with *minibuffers*, which are auxiliary buffers used for entering commands within **GNU EMACS**. If you unintentionally enter a second minibuffer (you will see a message such as `Command attempted to use minibuffer while already in minibuffer`), type `ctrl-g` to recover. The `ctrl-g` sequence may be used at any time to abort the most recently entered command and return to the body of the current buffer.

### 10.4 Using Windows Within EMACS

If you wish to work with two different files simultaneously, it is possible to create windows. The windows referred to here are windows within **GNU EMACS**, and should not be confused with the X window system; in this section, the “screen” refers either to the full screen display if you are not using X windows, or to the editing window if you are. Using **GNU EMACS** windows will allow you to display two or more files simultaneously on your screen, one above the other, and enable you to move between them to edit files and move text from one to another. To create a second window, type the command `ctrl-c 2`. This will open a new window at the bottom half of the screen. You may also use the `ctrl-c 2` command within one of the new windows to divide that window in half, and so on. The cursor will appear in the original window. The command `ctrl-c o` (Other) will move your cursor into the next (lower) window; in the bottom window, this command will move the cursor back to the top. You can edit or look at another file in a new window by using the find sequence `ctrl-x ctrl-f` described in the next section. To resume full screen editing, move your cursor to each window in turn and use the command `ctrl-s` to save the buffer in that window as a file. When all desired buffers are saved, move the cursor to the window you wish to continue working in and type `ctrl-c 1` for one window.

### 10.5 Entering GNU EMACS to Edit or Create a File

In order to invoke the **GNU EMACS** editor to edit an old file or create a new one, type `emacs filename <CR>` or `emacs <CR>` at the shell prompt. If you are using a single screen, the **GNU EMACS** setup will appear immediately. If you are using the X window system, typing the `emacs filename <CR>` command in the main window or clicking on the “emacs” option in the root mouse menu will

create a “rubberband” window which you may resize and move by dragging the mouse; the GNU EMACS editor will appear in this new window. If you invoked GNU EMACS with a filename, you will now be within the file you requested; if the file did not previously exist, you will see the designation (New File) at the bottom of your screen or window. If you entered GNU EMACS without specifying a filename, you will see a display screen giving some information and instructions about the editor. When you are within GNU EMACS, the following will appear at the bottom of your screen or window:

```
---**EMACS: *scratch*                (Fundamental)-----All-----
```

The designation `*scratch*` refers to a new file. If you entered GNU EMACS with a filename, this filename will replace `*scratch*`. The word in parenthesis refers to the current editing mode. (Fundamental) is the no-frills default mode. The next field tells where you are in the file. All means that you are at the beginning of the file; otherwise, you will see a percentage in this field indicating how far you are into the file.

If you entered GNU EMACS with a filename, you are now ready to edit. If you entered GNU EMACS without a filename, type `ctrl-x ctrl-f` to find a file; you will see at the bottom of your screen or window: Find file: `~/`. The `~/` is a partial pathname. If you had previously been working on another file, the directory pathname of that file will appear here. Add the pathname of the desired file using the `<del>` key to erase and alter the current path if necessary. If the file you are seeking does not exist, it will be created.

## 10.6 Basic Editing Commmands

### 10.6.1 Moving Within the File

Once you have entered the GNU EMACS editor, there is a set of commands you can use to move around within the file. The most frequently used cursor movement commands are listed below. More complex movement commands exist, and are explained fully in the GNU Emacs Manual.

```
ctrl-v - Move down one screen of text
M-v - Move up one screen of text
M-< - Move to the beginning of the file
M-> - Move to the end of the file
ctrl-p - Move to the Previous line of text
ctrl-n - Move to the Next line of text
ctrl-f - Move Forward one letter
ctrl-b - Move Backward one letter (same as <back space> key)
M-f - Move Forward one word
M-b - Move Backward one word
ctrl-a - Move to beginning of the current line
ctrl-e - Move to End of the current line
```



- M-[ - Move to beginning of the current paragraph
- M-] - Move to end of the current paragraph

NOTE: within GNU EMACS, the meta key on the SUN workstations is only the <esc> key, even though the <left> and <right> keys may also serve as meta keys outside of GNU EMACS. To use M-< and M->, the <shift> key must also be used to get the upper case character.

### 10.6.2 Inserting and Transposing Text

To enter text into a new file or to add to an old one, use the movement commands described above to move the cursor to where you wish to put the new text and begin typing. Use the `ctrl-o` command to open up a new line at the cursor. There is no special command needed to enter insert mode. If you make a mistake, you may use the following commands to reverse the order of letters and words:

- `ctrl-t` - Transpose two letters
- M-t - Transpose two words

### 10.6.3 Deleting and Undeleting Text

You can use the <del> key to delete the characters preceding the cursor. The commands listed below are for more complex deletions:

- `ctrl-d` - Delete the character under the cursor
- `ctrl-k` - Delete (Kill) from the cursor to the end of the line
- M-d - Delete one word forward
- M-<del> - Delete one word back
- `ctrl-x <del>` - Delete one sentence back

If you make a mistaken deletion, use the command `ctrl-y` to “Yank”, or reinstate, the text.

## 10.7 Searching for Strings of Text

The GNU EMACS editor has commands to locate specific strings within the text file. A string is any piece of text: it can include letters, digits and punctuation, and is of arbitrary length.

- `ctrl-s` - Search forward in the file
- `ctrl-r` - Search backward in the file

When you enter one of these commands, you will see a line at the bottom of the screen that will say `I-search:` or `I-search backward:` Enter the desired string after the colon and press <CR>. Pressing `ctrl-s <CR>` or `ctrl-r <CR>` repeatedly will find subsequent or previous occurrences of the same string. If the string is not found, you will see the message: `Failing I-search`, and your cursor will be moved to the string that is most similar to the one you requested. Case will be ignored for searches, even though UNIX is usually sensitive to case. You may also enter modes with

the command lines `Regexp I-search:` or `Wrapped I-search:`. These modes are used in a manner similar to the basic searches described above. To end a search, press the `<esc>` key.

## 10.8 Moving and Copying Blocks of Text

You can move and copy blocks, or regions, of text by marking the text you wish to move, deleting it from the current location if desired, and reinstating it at the new location. The procedure is shown below. There are also commands you can use to mark paragraphs, pages and buffers for moving and copying. Consult the **GNU Emacs Manual** for more details.

`ctrl-<space bar>` - Mark the beginning of the desired region

Use the cursor movement commands to move to the end of the desired region

`ctrl-w` - Delete (Wipe) the text within the marked region; this will be saved in a memory buffer.

`M-w` - Alternatively, put the marked text in the memory buffer to be moved, but keep a copy in the original location as well.

Use the movement commands to put the cursor where you want the text moved to.

`ctrl-y` - Reinststate (Yank) the text at the new location.

You can use the `ctrl-y` command repeatedly to reinststate the text in the buffer at multiple locations.

## 10.9 Some Special Character Commands

This section describes commands you can use to perform special operations within **GNU EMACS**. These are known as *character commands* because the operations are performed by typing special characters. The following are only a few of the available character commands; see the **GNU Emacs Manual** for more.

The command below may be used to repeat another command a specified number of times. After typing `ctrl-u`, enter the desired number of repetitions, followed by the command to be executed.

`ctrl-u [number] [command]` -- Execute the command the specified number of times

The following commands will change the appearance of your text file:

`M-l` -- Convert the current word to Lower case letters

`M-q` -- Reformat the current paragraph

`M-s` -- Center the text preceding the cursor

`M-u` -- Convert the current word to Upper case letters

To use the following command, move to the end of the word you want underlined and enter the command. Control characters will be inserted before and after the underlined word to let the computer know that it should be underlined when printed. On the screen, this will look like: `BwordE`; the actual underscore characters will not appear on your terminal.

**M- \_ -- Underline the word preceding the cursor**

The following command will check the spelling of a word within the file. Move the cursor to the word you wish to check and enter the command. If the word is found by the speller, the designation **correct** will appear at the bottom of your screen. If not, the incorrect spelling will appear at the bottom of the screen. You may use the **<del>** key to change the spelling, or enter **<CR>** to keep the word as it is. To exit spelling mode, type **ctrl-g**.

**M- \$ -- Check the spelling of the previous word**

### 10.10 Extended (M-x) Commands

One of the special features of GNU EMACS is its capacity to allow you to use specialized commands to perform complex functions that cannot be done using the character commands alone. These are called *extended* (or **M-x**) *commands*. Some of the basic **M-x** commands are described below. To execute these commands, type **M-x** anywhere within the file. Your cursor will then appear on a command line at the bottom of the screen (the minibuffer); at this point you can enter the appropriate command word(s). To abort an **M-x** command, type **ctrl-g** (twice if necessary). The command **M-x apropos <CR>** will give you more information about the various extended commands. Consult the GNU Emacs Manual for a full list of **M-x** commands and a more detailed description of how to use them.

A GNU EMACS parameter can be set that causes the text to wrap around to the next line as you type. Entering the command below alternately turns auto fill mode on and off. If it is on, the status line at the bottom of your GNU EMACS screen will say (Fundamental Fill).

**M-x auto fill <CR> -- Toggle auto fill mode**

Another parameter can be set that automatically saves your work after a certain number of keystrokes. Auto-saved files will be prefixed and suffixed with # signs in your file listing.

**M-x auto-save <CR> -- Toggle auto-save mode**

The command below can be used to view a listing of your files from within GNU EMACS. When you enter the command, a partial pathname will appear at the bottom of your screen; enter the path of the directory that you wish to list.

**M-x dired <CR> -- List the files in a directory**

The following command alternately turns Lisp mode on and off. When on, using the <tab> key produces indentation similar to pretty-printing, and matching parentheses will be indicated. If Lisp mode is on, the status line at the bottom of the GNU EMACS screen will read (Lisp).

**M-x lisp mode <CR> -- Toggle Lisp mode**

The command below will allow you to run Lisp from within GNU EMACS; Lucid Common Lisp is the default implementation. Before entering the command, divide your screen into two windows as described in Section 10.4. Lisp code may be entered and edited in one window and expressions evaluated in the other. Changes made to function definitions in the editing window are immediately available for use in the evaluation window. Move between the windows using **ctrl-x o**.

**M-x run lisp <CR> -- Run Lisp from within {\tt GNU EMACS}**

The commands below will locate occurrences of a string and replace them with a substitute string. After typing either of the commands above, press the <CR> to prompt for the string to replace. Enter <CR> again, and type the word to substitute. If you use **replace string**, all instances of the string will be replaced immediately. If you use **query replace**, you will be presented in turn with each occurrence of the original string. Typing <space bar> will cause the current instance of the string to be replaced with the substitute string; typing <del> will leave it as it is. More options for **query replace** may be found in the GNU Emacs Manual. To leave replace mode, type <esc>.

**M-x replace string <CR> -- Replace all occurrences of a string within the file**

**M-x query replace <CR> -- Replace specific occurrences of a string within a file**

The following command invokes the **ispell** program, which will search through the file, find misspelled words, and ask you whether you wish to replace them. Enter this command at the beginning of the file to be checked. The **ispell** program works in a manner similar to the **M-§** command for individual words. When you use this command, misspelled words will appear at the bottom of your screen. You may type a replacement word at this point, which will cause GNU EMACS to enter query-replace mode and ask if you want to change each occurrence of that spelling. Alternatively you may enter <CR> to skip over the word. If you skip the word, the program will assume that the word is spelled correctly, and it will not ask you about future occurrences of the same word. Type **ctrl-g** to quit the spelling program.

**M-x spell buffer <CR> -- Check and correct the spelling of the file**

The `revert-buffer` command may be used to recover changes to the buffer directly from the disk. This may be necessary if, for example, two users make changes to the same file at the same time. The `recover file` command is used to get the latest version of the file from the disk.

```
M-x revert buffer -- Read contents from disk
M-x recover file -- Recover latest version from disk
```

The command below will undo your most recent change to the buffer, for example, reinstating the most recently deleted material at the location from which it was removed.

```
M-x undo <CR> -- Undo the most recent change
```

## 10.11 Using Keyboard Macros

A *keyboard macro* is a special function you can define to abbreviate a sequence of commands. This will allow you to perform multi-command operations with a single character command. The procedure below can be used to create simple macros. Such macros can also be named or stored as permanent user-defined commands. See the *GNU Emacs Manual* for details on more advanced keyboard macro usage.

```
ctrl-x ( - Begin defining a keyboard macro
Type the commands you want to become part of the current macro
ctrl-x ) - End the definition of the current keyboard macro
ctrl-x e - Execute the most recently defined macro (this may be used repeatedly)
ctrl-u [number] ctrl-x e - Execute the macro the specified number of times
ctrl-g - Abort the current macro definition
<esc> - Kill the most recently defined macro
```

## 10.12 Saving the File and Exiting GNU EMACS

When you are finished editing a file, save it by typing `ctrl-x ctrl-s`. A new version of the file will be created that contains the changes you have just made, and the notice: `Wrote filename` will appear at the bottom of your screen or editing window. If the file remains the same, you will see: `(No changes need to be saved)`. It is recommended that you save your files frequently. If the system crashes or is rebooted, any editing work done before the last save is safe; you may lose changes made to the buffer after that save.

You may also save the changes you have made to another file you specify. Type `ctrl-x ctrl-w` from within *GNU EMACS*, and enter the desired file name by completing the partial pathname at the bottom of the screen.

If you are not using a window system, type `ctrl-z` to stop the job and exit the editor. If you are using X windows, moving the mouse out of the window will deactivate the job. Resume the editing job either by using the `fg` command (see Section 3.3) or by moving the mouse back into the editing window. To kill the job (as well as the editing window) and exit the editor, type `ctrl-x ctrl-c`.

When you are done with all editing work, kill any existing GNU EMACS jobs by using the commands `kill %[job-number] <CR>` with a number from the `jobs` listing, or `kill [PID] <CR>` with a PID from the `ps` listing. Editing jobs can be recognized by the `emacs` designation in either listing.

## 11. WORKING WITH FILES

UNIX provides a number of features that allow for more advanced work with files. These include the ability to compare, contrast, sort, and count elements in files.

### 11.1 Locating Strings within Files

The `grep` command, which may be used to find strings within files, can be quite useful given the potentially confusing nested directory structure. Usage of this command is `grep [string] filename <CR>`. Replace `[string]` with the string you wish to locate; if the string includes blank spaces, enclose the string within double quotes ("`string 1`"). You may use the wildcard symbol `*` to search all files within a specified directory; for example `grep [string] /wh/liz/admin/* <CR>` will find the string in all files within the directory `/wh/liz/admin`. Within a string, different characters can be used to represent special cases. The symbols `*` and `$` are used to designate the beginning and end of a line, respectively. The wildcard symbol `*` can be used to indicate a recursive pattern. Square brackets `[ ]` may be used to search for classes of matching characters. Consult the standard UNIX documentation for more details on `grep` usage, or type `man grep <CR>`.

### 11.2 Counting Elements in a File

The `wc` (Word Count) command will count elements in a file. You may invoke the `wc` command with one of three options: `wc -l filename <CR>` returns the number of lines in a file; `wc -w filename <CR>` returns a count of the words; and `wc -c filename <CR>` returns the number of characters. Using the command `wc filename <CR>` without one of the options above will return a count of lines, words, and characters, in that order.

### 11.3 File Comparison

Two commands available for comparing files are `comm` and `diff`. The `comm` (COMpare) command compares two files and returns a three column chart. Type `comm file1 file2 <CR>`. Column one will show lines that only exist in `file1`, column two lists lines that only exist in `file2`, and column three lists lines common to both files.

The `diff` (DIFFerence) command compares two files and returns the lines that differ. Type `diff file1 file2 <CR>`. Lines from the first file are marked with the symbol `<`; lines from the second file are designated by `>`. You must look through the line to find the actual difference. The `diff` command has many available options, and may also be used on directories. See the full UNIX documentation or consult `man diff <CR>` for more details.

### 11.4 Pruning, Sorting and Splitting Files

The `uniq` (UNIQue) command can be used to prune files of redundant material. Enter `uniq file1 file2 <CR>`. The program will compare adjacent lines in `file1`, and remove any instances of repeated lines. The cleaned up file will be returned as `file2`.

The UNIX `sort` command can be thought of as a merge operator. To use this command, type `sort -o file-new file-1 file-2...file-n <CR>`. The program will go through the given files, compare the contents, and return a file containing the combined contents of all the input files without duplications. The `-o` option allows you to give the name of the file in which to put the merged contents (`file-new` in the above example). Omitting this option will direct the output to the terminal screen. This command can also be used with different sort fields. See the full UNIX documentation for more information, or type `man sort <CR>`.

The `split` command provides a method for dividing large files into smaller pieces. Command usage is `split [-number] file-in file-out <CR>`. Replace `[-number]` with the desired number of lines per piece; the default size is 1000 lines. `file-in` is the file to be split. The desired root name of the output file is `file-out`; if no name is given, this file will be called `x`. Appended to the output file name will be a sequence of letters indicating which part of the original file it contains. For example, if the command `split -1000 file1 new-file <CR>` is given, with `file1` being a 4000 line file, the result will be four 1000-line output files named `new-fileaa`, `new-filebb`, `new-filecc`, and `new-filedd`.

### 11.5 The UNIX Spell Checker

UNIX provides a spell checking program that may be used with raw text files or files prepared using `ed` and `troff`, the standard UNIX editor and text processor. This program is not familiar with `YTeX`, and will flag all `YTeX` commands within a file as misspellings. To use this program, type `spell filename <CR>`. Words that are misspelled or that cannot be located in the dictionary will be displayed on the screen. You may direct the misspelled word list to the bottom of the original file by typing `spell filename >> filename <CR>`.



## 12. TEXT FORMATTING WITH $\gamma$ TeX

### 12.1 Introduction

$\gamma$ TeX is a program which takes a file that you have prepared using the GNU EMACS editor and translates it into a set of instructions which will cause the printer to output the file in the style you have specified. This process is known as *text formatting*, or *typesetting*. Although it is possible to print a raw GNU EMACS text file directly, formatted files have a much more pleasing appearance and allow a wider range of style options.

$\gamma$ TeX is not the only text formatting program available to you on the UNIX systems. Another widely used formatting program is LaTeX; this program provides somewhat more options than  $\gamma$ TeX, including automatic table of contents creation, and it allows the use of a program called BibTeX, which automatically compiles and formats bibliographical information. There is also an extension available known as SLiTeX for producing transparencies. This guide will focus on  $\gamma$ TeX, because it is somewhat easier to use; many features and commands in  $\gamma$ TeX also apply to LaTeX. This guide will describe the basic features and operations of  $\gamma$ TeX and the commands you will need to format simple documents. For an detailed treatment of  $\gamma$ TeX, consult *How to Use  $\gamma$ TeX*, by Daniel Brotsky, A.I. Laboratory Working Paper 273, © Massachusetts Institute of Technology (1986). For a complete description of equation and document typesetting, see *The TeXbook*, by Donald Knuth, © American Mathematical Society (1986). Although this book is based on the original TeX program, it is quite helpful. If you prefer to use LaTeX, refer to *LaTeX Document Preparation System: User's Guide and Reference Manual*, by Leslie Lamport, © Addison-Wesley (1986). In addition, there are standard UNIX text formatting programs known as *nroff* and *troff*; these programs do not offer as many options or features as the TeX programs. If you wish to learn more about them, consult the standard UNIX documentation.

$\gamma$ TeX may seem quite confusing at first, especially if you are accustomed to working with a dedicated word processing system. Remember, however, that you only need to know a small subset of commands to get started. A good way to learn about  $\gamma$ TeX is to compare an unprocessed *.ytex* file with the final formatted document and notice what commands produce what output.

### 12.2 Using $\gamma$ TeX: The Basic Idea

$\gamma$ TeX commands, also called *backslash commands*, are commands inserted throughout the text file to tell the printer what the final output should look like. These commands always begin with the character `\`. Files that contain commands for  $\gamma$ TeX will usually end in the suffix *.ytex* or *.tex*. Once you have prepared your *.ytex* file, you can run the  $\gamma$ TeX program on it to produce a new file that contains instructions that can be interpreted by the printer.

In the course of preparing your *.ytex* file, you may wish to insert comments into the file as notes or reminders to yourself. This can be done anywhere within the file. To enter a comment, type the character `%` followed by the desired comment text.

## 12.3 $\gamma$ TeX Opening Commands

Certain  $\gamma$ TeX commands must be included at the beginning of every `.ytex` file in order to set up the general format of the document.

### 12.3.1 Setting Sizes

At the beginning of the file, enter commands to set the size of the type and of the margins that will appear throughout the formatted document:

```
\typesize=11pt
\hsize=6.0in
\vsiz=9.0in
```

The first command above indicates the size of the typeface you wish to use. 10pt is standard typewriter size; 11pt and 12pt are slightly larger.

The `\hsize` (Horizontal SIZE) command sets the width of the text to be printed. The `\vsiz` (Vertical SIZE) command sets the length of the text on the page. If you are using 8.5 x 11 inch paper, an `\hsize` of six inches (6.0in) and a `\vsiz` of nine inches (9.0in) will produce appropriate margins; these may be changed to accommodate different paper sizes or special layouts.

### 12.3.2 Inputting Auxiliary Files

If you frequently produce complex documents which have a similar format, you may wish to establish a file that contains the basic set up commands instead of re-entering them each time you create a `.ytex` file. Such a file, which may be named `/wh/username/ytex.local` (since it contains local parameters), can contain commands for sizing, page numbering, section numbering, etc.

When you run the  $\gamma$ TeX program on a specific `.ytex` text file, you will want the program to read the commands in the `/wh/username/ytex.local` file. To ensure that this occurs, enter the `\input` command after the size commands in the text file:

```
\input /wh/username/ytex.local
```

Although the full path name is not necessary if your `.ytex` text files are in the same directory as the `ytex.local` file, using the full path will ensure that the proper file is accessed, and will allow the program to be run from any machine and by other users.

The `input` command may also be used to input files of equation commands, references, or user-defined macros (see below). Ask an experienced  $\gamma$ TeX user how to do this.

### 12.3.3 Defining Macros

To simplify your editing work, you may wish to define *macros*, which are special user-defined functions. For example, you can define a command that will spell out a long string when you enter

an abbreviation. These commands are in the following form, and should be entered after the input commands but before the actual text of your document:

```
\def\AIL{ Artificial Intelligence Laboratory}
```

When the above command is entered, typing \AIL within your text file will cause **Artificial Intelligence Laboratory** to appear in the formatted output.

There are more complex types of macros that can be defined, such as letter and document formats. Refer to a **The TeXbook** or consult an expert for more information.

## 12.4 Stylistic Commands

### 12.4.1 Layout Commands

Normally in a formatted .y<sub>T</sub>ex file, text will be arranged so that it is right- and left-justified and all the lines will be filled. If you type:

```
This is a line
of text. It will be
filled and justified.
```

the following output will result:

```
This is a line of text. It will be filled and justified.
```

If you do not want your text arranged in full lines, use the following set of commands:

```
\beginnofill                                %Do not fill the following text
Text to be
left unfilled
\endnofill                                    %End non-fill mode
```

When you skip a line between two portions of text, the y<sub>T</sub>EX program will assume that you are starting a new paragraph. It will insert a blank line into the finished output and indent the following line five spaces. If you do not want your text to be indented, enter the following command on the line preceding the text:

```
\noindent                                    %Do not indent the next line
```

If you wish to center a line of text, use the following command:

```
\centerline{text to center}           %Center a line of text
```

The text to be centered should be entered within the brackets.

There are two ways to create underlined text. The command:

```
\underbar{text to underline}         %Underline text
```

will cause the text within the brackets to be underlined.

The command:

```
\line{\hrulefill}                   %Draw a line
```

will draw a line across the page from margin to margin. This can be used under a line of text or by itself.

$\TeX$  has commands that automatically number chapter and section headings in outline form, and print the headings in bold face type and graduated sizes. Typing the commands:

```
\chapter{This is Chapter One}        %Begin a chapter
\section{This is the First Section}   %Begin a section
```

will produce the formatted output:

## **1. This is Chapter One**

### **1.1. This is the First Section**

Other commands are available in  $\TeX$  to produce inverted paragraphs, bulleted paragraphs and more. Refer to **The TeXbook** for details.

#### **12.4.2 Spacing Commands**

When you are editing a text file, inserting blank lines into the `.ytex` file will not cause equivalent spacing to appear in the finished output. The methods for producing blank lines in the formatted version are shown below:

<code>\smallskip</code>	<code>%Small Skip</code> - about one half line skipped
<code>\medskip</code>	<code>%Medium Skip</code> - about one line skipped
<code>\bigskip</code>	<code>%Large Skip</code> - about two lines skipped

Exact spacing will depend on the typesize and font you are using. You may have to experiment to achieve the spacing you desire.

To skip larger chunks of space, you can use the `\vskip` command. This command works with measurements in inches or millimeters:

<code>\vskip 0.5truein</code>	<code>%Skip one half inch</code>
<code>\vskip 4.0mm</code>	<code>%Skip four millimeters</code>

To double space your entire file or a part of it, use the following commands:

<code>\doublespace</code>	<code>%Begin double spacing</code>
Text to be double spaced	
<code>\singlespace</code>	<code>%End double spacing</code>

Normally, the  $\TeX$  program will break your document into pages properly. However, if you wish to override the default pagination, use the command:

<code>\vfil\eject</code>	<code>%Start a new page</code>
--------------------------	--------------------------------

### 12.4.3 Font Changing Commands

One of the nice features of  $\TeX$  is that it allows you to use a variety of different fonts to improve the appearance of your document.

For example, if you have set your typesize to 11pt, you may wish to have certain lines, such as titles, printed in a `large` or `larger` size type. To do this use the commands:

<code>{\bigsize [text]}</code>	<code>%Large size text</code>
<code>{\biggsize [text]}</code>	<code>%Larger text</code>

These commands print text in increasingly larger type fonts. Put the text to be enlarged after the command within the brackets (omitting the `[ ]`). Again, experimentation may be necessary to achieve the desired results.

If you wish to emphasize a string of text, you can put it in *italics*, **bold face**, *slanted type* or **typewriter font**. Use the following commands, replacing [text] with the text to be emphasized (omitting the [ ]):

<code>{\it [text]}</code>	<code>%Italics</code>
<code>{\bf [text]}</code>	<code>%Bold face</code>
<code>{\tt [text]}</code>	<code>%Typewriter font</code>
<code>{\sl [text]}</code>	<code>%Slanted type</code>

#### 12.4.4 Special Characters

As you will see when you begin to format equations, many of the specialized  $\TeX$  commands contain normally used symbols. For example, the equation delimiter is  $\$$ . If you wish to actually print out a  $\$$ , you must precede it with a backslash ( $\backslash\$$ ). This also true for the symbols  $\#$ ,  $\&$  and others.

### 12.5 Formatting Equations

The  $\TeX$  program will enable you to typeset complex equations. As the sample equations on the next page demonstrate, available features include Greek letters; varying sizes of parentheses, brackets, and braces; subscripts and superscripts; and symbols for derivative, sum, square root, and vector, among others.

#### 12.5.1 Horizontal Mode

There are two modes of equation formatting in  $\TeX$ . The first is called *horizontal mode*, also known as *text style*. In this mode, numbers, symbols, and short equations are inserted within a line of normal text. When entering the text, the equation should be preceded and followed by single dollar signs. Letters within the dollar signs will be italicized. For example, if you type:

If  $m=2$ , then the solution is  $J_2(f)$ .

the formatted output will be:

If  $m = 2$ , then the solution is  $J_2(f)$ .

Horizontal mode can also be used to number refernces, as the following example shows. If you type:

This fact was also pointed out by Winograd<sup>[15],[23]</sup>.

the processed output will be:

This fact was also pointed out by Winograd<sup>[15],[23]</sup>.

### 12.5.2 Vertical Mode

The second available equation mode is called *vertical mode*, or *display style*, and is used to produce more complex equations that are set off from the body of the text, as shown by the sample equations below:

$$\Omega(\mu, \nu) = \frac{s_3 - ps_1 - qs_2}{\sqrt{1 + p + q^2}} \quad (1)$$

$$J_m(f) = \int \int_{-\infty}^{+\infty} \sum_{\nu=0}^m \binom{m}{\nu} \left( \frac{\partial^m f}{\partial x^\nu \partial y^{m-\nu}} \right)^2 dx dy \quad (2)$$

$$\left( -\frac{4f}{4 - (f^2 + g^2)}, -\frac{4g}{4 - (f^2 + g^2)}, 1 \right) \cdot \vec{s} \quad (3)$$

Equations in vertical mode are delimited by double dollar signs in the `.ytex` file. These equations may extend over several typed lines. As an example of how to use this mode, the input below was used to produce Equation (1) above:

```


$$\Omega(\mu, \nu) = \frac{s_3 - ps_1 - qs_2}{\sqrt{1 + p + q^2}}$$


```

### 12.5.3 Some Basic Equation Commands

There are too many equation typesetting commands to list in this manual, and most of them are only used for specialized purposes. Some of the more basic commands are shown below. All of these may be used in either horizontal or vertical mode. See *The TeXbook* for a complete listing of the equation commands and how to use them.

INPUT	FORMATTED OUTPUT	
<code>\delta</code>	$\delta$	%Lower case Greek letter
<code>\Delta</code>	$\Delta$	%Upper case Greek letter
<code>\int</code>	$\int$	%Integral sign
<code>\infty</code>	$\infty$	%Infinity sign
<code>x^2</code>	$x^2$	%Superscript
<code>y_3</code>	$y_3$	%Subscript

<code>\sqrt2</code>	$\sqrt{2}$	%Square root
<code>\vec z</code>	$\vec{z}$	%Vector
<code>x\le y\ne z</code>	$x \leq y \neq z$	%Less than or equal to; not equal

## 12.6 Creating Tables

$\TeX$  allows you to construct fairly intricate tables. A normal table will fit entirely on one page. To create this type of table, enter the command `\begin{table}`. A table that extends over page breaks is called an open table. Begin this type of table by typing `\begin{opentable}`. Directly after either of these opening commands enter indicators of the number of columns in brackets ([ ]). Indicators to use are `l`, `c`, or `r`; these indicate that the columns should be left-justified, centered, or right-justified, respectively. The number of indicator letters in the brackets will control the number of columns in the table. For example, `[lll]` will create three left-justified columns. When typing the table text, use the `&` symbol to indicate that you wish to begin the next column. A `\cr` is used at the end of each completed line of text; if used by itself, `\cr` it will cause a line to be skipped. Use the command `\end{table}` or `\end{opentable}` to resume normal text format. The following commands will create a simple table. See *The TeXbook* for more detail.

```

\line{\hrulefill}
\begin{table} [lcr]
COURSE TITLE &INSTRUCTOR &NUMBER\cr
\cr
Introduction to Artificial Intelligence &B. Horn &6.123\cr
Visual Information Processing &T. Poggio &9.380\cr
Graduate Topics in Linguistics &N. Chomsky &24.999\cr
\end{table}
\line{\hrulefill}

```

The resulting table will look like:

---

COURSE TITLE	INSTRUCTOR	NUMBER
Introduction to Artificial Intelligence	B. Horn	6.123
Visual Information Processing	T. Poggio	9.380
Seminar in Robot Hand Control	J. Hollerbach	9.393
Graduate Topics in Linguistics	N. Chomsky	24.999

---



## 12.7 Closing Commands

When you have finished entering text and are ready to complete your `.ytex` file, enter the following commands:

```
\filpage          %Fill the current page
\end              %End the .ytex file
```

If you wish to print out only part of the formatted `.ytex` file, place the `\end` command where you wish to stop printing.

## 12.8 Running $\Upsilon$ TeX

In order to translate the commands you have entered in the `.ytex` file into commands that can be understood by the printer, it is necessary to run the  $\Upsilon$ TeX program on the file. To do so, type `ytex /wh/username/filename.ytex <CR>`. If you are running the  $\Upsilon$ TeX program from within the directory that contains the `.ytex` file, you do not need to use a full pathname. It is not essential that the file have the `.ytex` suffix, although some suffix is necessary; the `.ytex` is useful for mnemonic purposes. The above command will initiate a  $\Upsilon$ TeX job. As  $\Upsilon$ TeX runs, you will see the current status of the program on your screen. The program will let you know what file it is processing, what input files it is using, and where it currently is in the file. Chapters and sections will be listed by name as they are processed, and page numbers will appear in square brackets, as shown below:

```
This is TeX, Version 2.0 for Berkeley UNIX (preloaded format =
ytex-cm 87.10.25)
(/wh/liz/filename.tex {YTEX version 2.0} {No fixes} {typesize=11.0pt}
(/wh/liz/ytex.local) {Main title} [0] [1] [2] {Chapter: \n This is
Chapter One} [3] {Section: \n This is the First Section} [4] [5] [6]
[7]
```

```
Output written on filename.dvi (7 pages, 4136 bytes).
Transcript written on filename.log.
```

## 12.9 The `.dvi` and `.log` Files

As you can see from the above fragment, the  $\Upsilon$ TeX program produces an output file called `filename.dvi`. This is the file that contains the instructions for the printer. All print files output by  $\Upsilon$ TeX will end with the `.dvi` suffix. The `.dvi` file consists of unreadable control characters to be interpreted by the printer, not the output as it will appear when printed.

The  $\Upsilon\text{T}\text{E}\text{X}$  program also produces a file that contains a transcript of the running of the program, including error messages. This file will be named `filename.log`. All transcript files will end with the `.log` suffix.

Each time you rerun the  $\Upsilon\text{T}\text{E}\text{X}$  program on the same `.ytex` file, the corresponding `.dvi` and `.log` files will be updated. After you have eliminated all errors and printed the final document, the `.dvi` and `.log` files may be deleted, since the text is saved in the `.ytex` file and can be reformatted if necessary.

### 12.10 Debugging the `.ytex` File

The  $\Upsilon\text{T}\text{E}\text{X}$  fragment shown above is what will appear if your `.ytex` file contains no errors. If an error is encountered in the input file, the  $\Upsilon\text{T}\text{E}\text{X}$  program will stop and tell you what is wrong, as shown below:

```
This is TeX, Version 2.0 for Berkeley UNIX (preloaded format =
ytex-cm 87.10.25)
(/wh/liz/filename.tex {YTEX version 2.0} {No fixes} {typesize=11.0pt}
(/wh/liz/ytex.local) {Main title} [0] [1]
! Undefined control sequence.
1.55 \bigskip
?
```

Many of the  $\Upsilon\text{T}\text{E}\text{X}$  error messages are cryptic, but with practice you will learn to interpret the most common ones. In this case `!Undefined control sequence` means that  $\Upsilon\text{T}\text{E}\text{X}$  did not understand a command. The designation `1.55` tells you that the error occurred on line number 55 of the file. In this case the unknown sequence `\bigskip` was entered instead of the legal command `\bigskip`. At this point, the easiest thing to do is to edit the `.ytex` file to fix the error. To exit the  $\Upsilon\text{T}\text{E}\text{X}$  program, type `x` after the question mark. This will not kill the program, but will stop it and allow you to resume after corrections are made. The following notice will appear:

```
No pages of output.
Transcript written on filename.log
```

You can now enter `GNU EMACS` to edit the `.ytex` file. If you are using X windows, move the mouse back into the editing window; if the editing job is stopped, type `fg %[job-number] <CR>`; otherwise invoke `GNU EMACS` as described in Section 10.5.

To get to the line that contained the error, move the cursor to the beginning of the file and type `ctrl-u`. Your cursor will appear in a minibuffer at the bottom of the screen. Here, type the number of the line on which the error occurred as shown in the  $\Upsilon\text{T}\text{E}\text{X}$  error message, and press `ctrl-n`. The

cursor will be moved to the line containing the error. You can now fix the error, save the file, and run the  $\text{\TeX}$  program again. You may have to do this a number of times before the file is completely error-free.

If you get an error message you do not understand, press  $\langle\text{CR}\rangle$  at the question mark following the message. Most errors are not fatal, and the  $\text{\TeX}$  program will usually skip over them and continue processing the `.ytex` file. Most often  $\text{\TeX}$  will produce a `.dvi` file even if the input file contained errors. When you print the `.dvi` file and look at the output, you should be able to determine what the problem was from the appearance of the printed document.

### 12.11 Previewing the `.dvi` File

If you are using a SUN workstation, or any other system that provides a *bit-map* display, it is possible to see what your `.dvi` file looks like before you send it to the printer. On a SUN workstation running X windows, type `texx filename`. This will cause a "rubberband" window to appear on your screen which you may resize and move by dragging the mouse. The final form of the document will appear in this window, generally in a format showing two adjacent. There are two viewing modes available: Normal Mode shows the basic document layout, Large Mode can be used to view a part of the document in detail. Large Mode is accessed by holding down a mouse button over the text you wish to view. Pages can be accessed by typing `[page-number] g` within the previewer. To leave the `.dvi` previewer and return to the shell, type `q`. For more information, type `man texx`  $\langle\text{CR}\rangle$ .

### 12.12 A Sample Document

The following example shows a `.ytex` file used to produce a letter, as well as the final processed document. Examine this example to see which  $\text{\TeX}$  commands produce which features in the final output.

Input `.ytex` file:

```

\type size=11pt           %Set typesize
\input /wh/liz/ytex.local   %Input local  $\text{\TeX}$  command file

\def\AIL{ Artificial Intelligence Laboratory} %User - defined macro

\nopagenumbers           %Do not number pages of letter
\noheaders

|
\bigskip                 %Start six spaces down on page
\bigskip
\bigskip
\beginnofill            %Put address heading in block style

```

15 April 1988

\medskip — %Skip some space

Professor T. MacDuff  
 Department of Computer Science  
 University of Rochester  
 Rochester, NY 14627

\bigskip %Skip more space

Dear Professor MacDuff:

\medskip

\endnofill %End non-filled mode

\noindent %Do not indent paragraph

I would be happy to have you visit our laboratory the next time you are in Boston. I will be out of town until {\it August 16}, but any time after that would be fine. Let me know when you have finalized your plans.

\medskip %Skip space

\beginnofill %Enter no fill mode again

Yours sincerely,

\vskip .75in %Skip 3/4 inch

Professor T. MacBeth

MIT \ALL %Use of abbreviation macro

\bigskip

TM:lh

\endnofill %End no fill mode

\filpage %Fill page

\end %End .ytex file

15 April 1988

Professor T. MacDuff  
Department of Computer Science  
University of Rochester  
Rochester, NY 14627

Dear Professor MacDuff:

I would be happy to have you visit our laboratory the next time you are in Boston. I will be out of town until *August 16*, but any time after that would be fine. Let me know when you have finalized your plans.

Yours sincerely,

Professor T. MacBeth  
MIT Artificial Intelligence Laboratory

TM:lh

## 13. PRINTING FILES

### 13.1 Printers Available at the A.I. Laboratory

There are a number of printers at the lab that can be accessed from the UNIX systems and used to print either raw text files or .dvi formatted files. There are three fast, high capacity QMS 2400 printers that are capable of printing letter quality output on a variety of paper sizes. A fourth printer is a QMS PS-800 LaserWriter. The output quality of this printer is somewhat better, but it is slower and limited in what it can do. An Apple LaserWriter is also available. These are PostScript printers. The various printers, their names, and their locations are listed below:

PRINTER	NAME	LOCATION
7th Floor Laser Printer (2400)	pravda	Xerox Room, 744
7th Floor LaserWriter (PS-800)	le-monde	Across from Room 707
7th Floor LaserWriter (Apple)	the-washington-post	Across from room 739
8th Floor Laser Printer (2400)	daily-planet	Xerox Room, 800d
9th Floor Laser Printer (2400)	national-enquirer	Near Machine Shop

### 13.2 Defining a Printer

In order to use the printing commands, it is first necessary to define, or set, a printer. To do so, type `setenv PRINTER [printer-name] <CR>` (SET ENVIRONMENT), for example, `setenv PRINTER le-monde <CR>`. Once a printer is defined, all the printing commands described below will apply to that printer until the printer is redefined by using the `setenv PRINTER` command with a different printer name. It is also possible to define a default printer within your `.login` file.

### 13.3 Checking the Printer Status and Queue

To check the current status of all the printers in building NE43, use the command `lpc status <CR>`. This command will let you know which printers are enabled and ready to be used. The command `lpc status [printer-name] <CR>` will give this information about a specific printer.

The command `lpq <CR>` (Laser Printer Queue) may be used to view the print queue of the currently defined printer, including whether the printer is working, what job is currently printing, and what other jobs are waiting to print. A typical report will look like the following:

7th Floor QMS PS-2400 is ready and printing

Time	Owner	Job Files	Size
*16:44	liz	388 guide.dvi	1448
16:48	ariel	389 mail.txt	6789
16:55	little	390 Screen Hardcopy	8930

The most recent job printed was:

16:42 barb            project.dvi

The \* will appear next to the job that is currently printing. If there are no jobs in progress or in the queue, you will see the following message:

7th Floor QMS PS-2400 status: idle

The queue is empty. The most recent job printed was:

16:44 liz            guide.dvi

### 13.4 Printing a File

There are two basic printing options available depending on which type of file you wish to print. ASCII printing is used to print files that contain raw text, that is, text that has not been processed by a text formatting program, or programming code. The command to print raw files is `lpr filename <CR>` (Laser PRint).

You can also print .dvi files that result from the running of a text formatting program such as  $\text{\TeX}$ . The command to print processed files is `lpr -d filename <CR>` (Laser PRint -Dvi). Note that this differs from TOPS-20, in which one command is used for both types of printing.

### 13.5 Stopping a Print Job

If for some reason you decide you do not wish to print a file that has been sent to the printer, you can remove a job from the print queue, or stop it if it has already started to print. When you do a `lpq` listing, note that there is a number for each job listed before the filename in the Job column. To dequeue a job (remove a job from the queue), type the command `lprm [job number] <CR>` (Laser Printer ReMove). Alternatively, the command `lprm [username] <CR>` will dequeue all jobs owned by the specified user.

### 13.6 Printing Errors

Sometimes when you attempt to check the status or send a job to a printer, there will be a pause and you will see the message: `Host is not up`. The *host* is a special computer that routes

jobs from the different computers to the various printers. The host for most of the printers is named **Prep**. If a host computer is down, it will not be possible to use the printers served by that host. Such outages are usually of short duration; wait a few minutes and try to print the file again.

Once you have successfully sent your job to the printer, other types of messages may appear on your screen if the printer gets an error while printing your file. These messages are of the following form:

```
[Message from daemon@prep.ai.mit.edu 16-Jul-87 16:59:09]
7th Floor LaserWriter needs paper.
```

These messages usually refer to a paper jam, an empty paper tray, or a similar physical problem. When this occurs, you will have to go to the printer to determine what is wrong and fix it. If the jobs in the queue seem to be taking an inordinately long time to print, it is a good idea to check the printer; only the user whose job is currently printing will receive the error messages.

### 13.7 Unwedging a Printer

If the printer has no clear mechanical problem and still will not print, the printer or its host is probably *wedged*. When this occurs, there are two sets of commands you can use to attempt to fix it. Type `lpc abort <CR>` to stop any current processes, immediately followed by `lpc start <CR>` to restart the printer. The problem may also be with the queue. To unwedge the queuing mechanism, type `lpc disable <CR>`, immediately followed by `lpc enable <CR>`. Never give the `abort` or `disable` commands without subsequently issuing the corresponding `start` or `enable` commands. If the printer still does not work, consult an expert or log a service call for repair.



## 14. OTHER PROGRAMS AND PACKAGES

There are a number of special packages and programs developed for use on UNIX systems. Only a few are listed here. Consult the standard documentation to find out more about standard UNIX packages, or a systems expert to find out about implementations available specifically on the AI Lab computers.

### 14.1 C

As mentioned previously, UNIX is implemented in the C programming language, and there are a number of C implementations available for programming use. To learn more about C programming, consult **The C Programming Language**.

### 14.1 Lisp

On the SUNs, you can run an implementation of Common Lisp called Lucid. This is a well developed implementation, and features the standard Common Lisp instruction set, as well as a number of extended commands and an efficient debugging environment. This is the default Lisp, and may be invoked by typing `lisp <CR>` at the shell prompt; this program is also invoked when you type `M-x run-lisp <CR>` from within GNU EMACS. To leave the Lisp program, type `(quit) ,CR>` at the `>` prompt. For more information, consult the **SUN (Lucid) Common Lisp User's Guide**, © Sun Microsystems, Inc. and Lucid, Inc. (1986). The current release is 2.0.

### 14.2 Wreq

There is a program called `wreq` (Write REquisition) available on the SUN system that allows you to submit orders to the AI Lab fiscal office electronically. It is recommended that all orders be submitted in this manner, since the program maintains a data base that facilitates keeping track of items ordered and money spent. Use this program by typing `wreq <CR>` at the shell prompt. This will give you a requisition template that may be filled in using GNU EMACS-like movement and editing commands. It may be necessary to have a system administrator change certain parameters to give you permission to run `wreq`. For more information and complete instructions, consult the AI Lab manual **The New Wreq**.

### 14.3 Games

There are many game programs available, stored in the directory `/usr/games` on the SUN system. Available games include Adventure, Chess, Cribbage, and Trek. Check the UNIX documentation or consult an expert for more details.

## APPENDIX A: LIST OF RESOURCES

- Brotsky, Daniel, **How to Use YTeX**, A.I. Laboratory Working Paper 273, Massachusetts Institute of Technology, Cambridge, MA, 1985, 1986. (Available from the A.I. Laboratory publications office).
- Christian, Kaare, **UNIX Command Reference Guide**, John Wiley and Sons, New York, NY, 1988. (Reference copy located in the A.I. Laboratory publications office).
- Highleyman, Liz A., **The Idiot's Guide to OZ**, A.I. Laboratory Working Paper 298, Massachusetts Institute of Technology, Cambridge, MA, 1987, 1988. (Available from the A.I. Laboratory publications office).
- Jones, Scott A., **Printer Cheat Sheet**, available for all A.I. Laboratory printers, Massachusetts Institute of Technology, 1986. (Posted copy located on or near each printer; also available from the A.I. Laboratory publications office).
- Knuth, Donald E., **The TeXbook**, Addison-Wesley Publishers, Reading, MA, 1984, 1986. (Available through the MIT Coop).
- Lamport, Leslie, **LaTeX Document Preparation System: User's Guide and Reference Manual**, Addison-Wesley Publishers, Reading, MA, 1986. (Available through the MIT Coop).
- Samuel, Arthur L., **First Grade TeX: A Beginner's Manual**, Report STAN-CS-83-985 (Version 1), Stanford University Department of Computer Science, Stanford, CA, 1983. (Available by mail from Stanford University).
- Stallman, Richard M., **GNU Emacs Manual**, Free Software Foundation, Cambridge, MA, 1985, 1986. (Available from the A.I. Laboratory publications office).
- Todino, Grace and John Strang, **Learning the UNIX Operating System: A Nutshell Handbook**, O'Reilly and Associates, Inc., Newton, MA, 1986, 1987. (Reference copy located in the A.I. Laboratory publications office).
- SUN (Lucid) **Common Lisp User's Guide**, Sun Microsystems, Inc. and Lucid, Inc., Mountain View, CA, 1986.
- SUN Microsystems **Documentation**, multi-volume set covering UNIX and SUN specifics, Sun Microsystems, Inc. Mountain View, CA, 1986. (Available from SUN Microsystems)
- Using TeX on MIT-OZ**, A.I. Laboratory, Massachusetts Institute of Technology, Cambridge, MA, 1985. (Available from the A.I. Laboratory publications office).
- X11 Manual**, O'Reilly and Associates, Inc., Newton, MA.