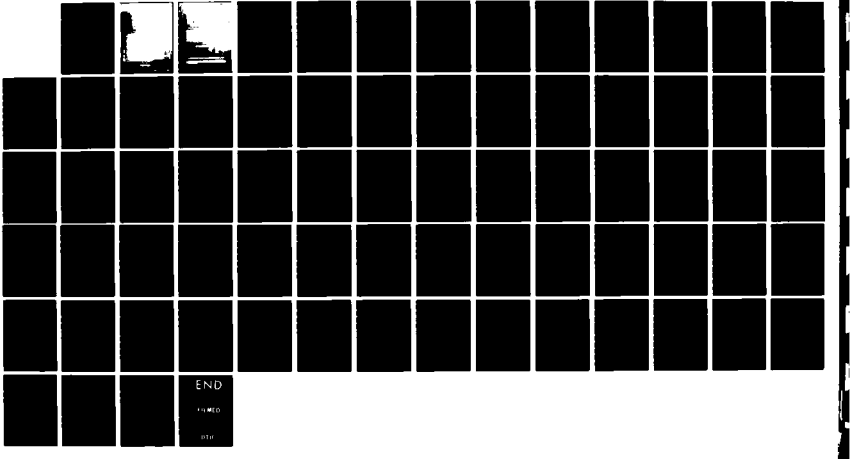
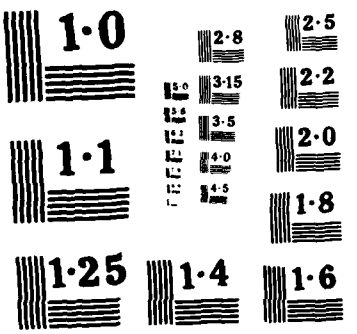
AD-A158 237 IDIOT'S GUIDE TO OZ: A MANUAL FOR THE COMPLETE BEGINNER 1/1
INTRODUCING EMACS..(U) MASSACHUSETTS INST OF TECH
CAMBRIDGE W GILSON 1984 AFOSR-TR-85-0583

UNCLASSIFIED F49620-83-C-0135 F/G 9/2 NL

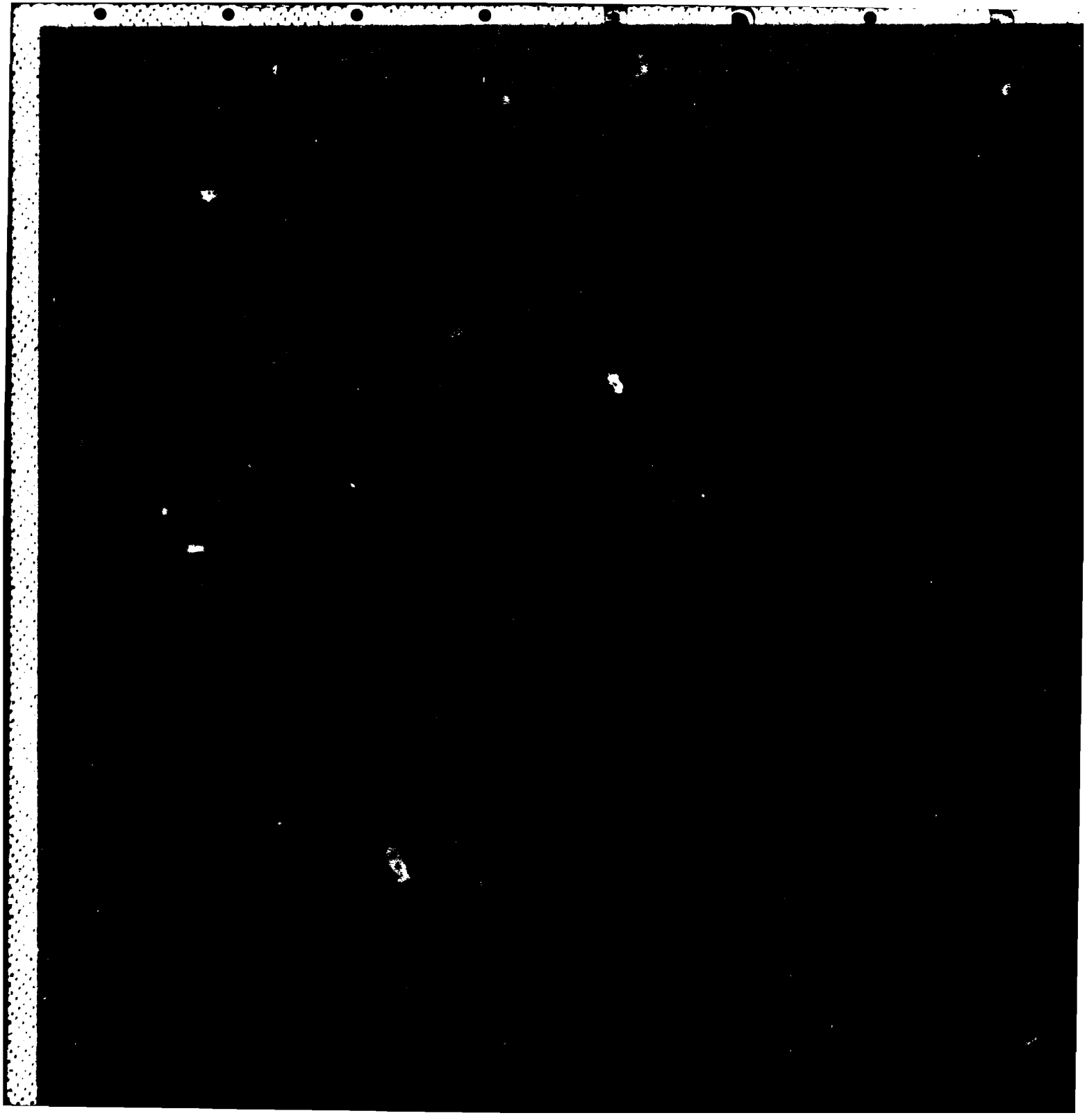




NATIONAL BUREAU OF STANDARDS
MICROCOPY RESOLUTION TEST CHART

AD-A158 237

IC
CTE



UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a REPORT SECURITY CLASSIFICATION: Unclassified	1b RESTRICTIVE MARKINGS: N/A
2a SECURITY CLASSIFICATION AUTHORITY: N/A	3 DISTRIBUTION/AVAILABILITY OF REPORT: Approved for public release; distribution unlimited.
2b DECLASSIFICATION/DOWNGRADING SCHEDULE: N/A	
4 PERFORMING ORGANIZATION REPORT NUMBER(S): N/A	5 MONITORING ORGANIZATION REPORT NUMBER(S): AFOSR-TR- 83-0583
6a NAME OF PERFORMING ORGANIZATION: Massachusetts Institute of Technology; 6b OFFICE SYMBOL (If applicable): N/A	7a NAME OF MONITORING ORGANIZATION: AFOSR/NL
6c ADDRESS (City, State and ZIP Code): E10-120, 79 Amherst Street, Cambridge, MA 02139	7b ADDRESS (City, State and ZIP Code): Bldg 410, Bolling AFB, DC 20332
8a NAME OF FUNDING/SPONSORING ORGANIZATION: Air Force Office of Sponsored Research; 8b OFFICE SYMBOL (If applicable): NL	9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER: F49620-83-C-0135
8c ADDRESS (City, State and ZIP Code): Bolling Air Force Base, Washington, DC 20332	10 SOURCE OF FUNDING NOS: PROGRAM ELEMENT NO. 61102F; PROJECT NO. 2313; TASK NO A5; WORK UNIT NO
11 TITLE (Include Security Classification): "A Manual for the Complete Beginner, Introducing the Word Processing Program and the Text Formatting Program	
12 PERSONAL AUTHOR(S): Gilson, William	
13a TYPE OF REPORT: Reprint; 13b TIME COVERED FROM _____ TO N/A	14 DATE OF REPORT (Yr., Mo., Day): 1984; 15 PAGE COUNT: 51
16 SUPPLEMENTARY NOTATION: Idiot's Guide to Oz	
17 COSATI CODES: FIELD / GROUP / SUB GR	18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number): Computer guide; Tops 20; Beginner manual

19 ABSTRACT (Continue on reverse if necessary and identify by block number)

SEE OTHER SIDE.

Accession For
NTIS GRA&I
DTIC TAB
Unannounced
Justification
By
Distribution/
Availability Codes
Dist | Avail and/or Special
A-1

INSPECTED 3

20 DISTRIBUTION/AVAILABILITY OF ABSTRACT: UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT ☐ DTIC USERS ☐	21 ABSTRACT SECURITY CLASSIFICATION: Unclassified
22a NAME OF RESPONSIBLE INDIVIDUAL: Dr. John Tangney	22b TELEPHONE NUMBER (Include Area Code): (202) 767-5021; 22c OFFICE SYMBOL: NL

DD FORM 1473, 83 APR EDITION OF 1 JAN 73 IS OBSOLETE.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

PREFACE

This manual is for complete beginners. It assumes the reader knows nothing about computers. It will teach you to log onto the computer called OZ* and do basic word processing: creating files, editing text, getting text printed. You will also learn how to send and receive electronic mail. To avoid confusion, there is a lot this manual doesn't discuss; but there is a list on pages 39–40 of other manuals available, with some comments about their usefulness for beginners.

To start, you will need a terminal that is connected to OZ. Because the MIT campus has several kinds of terminals, and more than one way of connecting to OZ, you should begin by asking someone nearby how to "wake up" your particular terminal so that you can use OZ. This waking-up process will involve no more than a few key-strokes; but write the sequence down so you don't forget it.

As a regular OZ user you will have your own *ACCOUNT*. That is, you will have a *LOGIN NAME*, complete with secret password, and you will have a certain amount of memory space allocated to your use. This manual assumes you do not yet have your own account, and it provides one for you to use temporarily.

In the word-processing sections of this manual you will be shown how to begin to use two programs: EMACS and TeX. The EMACS editor enables you to move letters, words, sentences—i.e., "text" around on your terminal screen; it can be thought of as a way of typing that is vastly superior to using a typewriter. The other program, TeX, allows you to get what you typed into the computer printed out on paper. TeX is a "text formatter."

IMPORTANT: As of October 1984 TeX at MIT is in a confusing state. There are at least three TeX programs: (1) the old TeX, known as TeX80 or Tbase; (2) the new TeX, known as TeX82; and (3) LaTeX, a modified form of TeX82. Most people still use TeX80, but since the recent introduction of TeX82 and LateX, TeX80 has entered a state of obsolesence: everyone is being encouraged to switch to the new programs. This presents a difficult situation for a beginner. The IDIOT'S GUIDE shows you how to use the old TeX, or TeX80. The old TeX still works; if you read through the first section of this manual and do the excercises, you will still learn how to begin to do word processing on OZ. However, it is obviously not a good idea to put a lot of time into learning a program that is being replaced by something different. The next edition of the IDIOT'S GUIDE will explain how to use the new programs, but until that appears, it is hoped that this manual will remain useful. The sections explaining EMACS and mail programs are still reasonably up to date, and there is a lot of information in these pages which persons new to computers might have trouble finding elsewhere. Anyone wishing directions toward learning the new TeX programs should turn to Section 20, page 39.

AIR FORCE [illegible]
[illegible]
Chief, Technical [illegible] Division

*OZ, also known as MIT-OZ, is a DEC PDP-20 computer located on the ninth floor of Building NE43.

CONTENTS

Section One:
BEGINNING WORD PROCESSING

Page

1. Introduction 1
2. Getting Started: Logging In 2
2.1 Conventions to Remember 3
3. The Directory 3
4. Copy a File 4
5. Text Editing 5
5.1 Keys and Commands Used So Far 10
6. Printing It Out 11
6.1 TeX and Dover Commands Used So Far 14
7. Errors 15
8. Cleaning Up Your Directory 16
8.1 Commands for Directory Cleanup 18
9. Killing Jobs and Logging Out 18
9.1 Job-Killing and Logout Commands 19
10. Creating a File from Scratch 19
11. Mathematics 22
12. Input and Output Samples . . . . . *(COLORED PAGES)* 24

Section Two:
MAIL

13. Sending and Receiving Mail 25
14. Using MM for Mail 25
14.1 Send Yourself a Letter 25
14.2 Using MM for Receiving Mail 27
14.3 MM Commands Used So Far 28
15. Using BABYL: the Basic Idea 28
15.1 BABYL: Sending a Letter 28
15.2 BABYL: Reading Your Mail 31
15.3 Basic BABYL Commands 32

Section Three:
ODDS AND ENDS

16. INQUIRE, WHOIS, and FINGER 33

16.1 INQUIRE, WHOIS, and FINGER Commands 33

17. The Files in the NCG Directory 34

18. Help on the System 35

19. List of Keys and Commands 36

20. List of Manuals 39

21. Glossary 41

22. Index 43

Section One:
BEGINNING WORD PROCESSING

1.0 Introduction

We'll assume you're sitting in front of a terminal. First, you will need to be familiar with two of the keys on the keyboard:

ESC (upper left)

and

CTRL (lower left)

These keys are used in combination with other keys. For example, ESC D means to press the ESC key, then D.

The common abbreviations for ESC and CTRL are $ and ↑, respectively. Throughout this manual both the letters on the keys and the abbreviations will be used:

ESC = $

and

CTRL = ↑

IMPORTANT

There is an idiosyncrasy of the keyboard that you must bear in mind—when you press ESC ($) in combination with another key, press ESC ($) once, release it, then press the next key. When you press CTRL (↑) you must hold it down while you press the next key.

2.0 Getting Started: Logging In

To begin:

We'll assume you are sitting at a terminal and have "woken it up" (see Preface). The screen, if the computer is working properly, will say something like

```
Kkjob Job 42, Users NCG, Account, TTY6,
 at 10-May-83 13:01, Used 0:00:22 in 0:50:10
```

Or the screen may be full of information, including a list of the login names of all the users currently on the system.

Hold down the CTRL (↑) key and press C

[NOTE: Throughout the rest of this manual instructions as to what commands to type to the computer will be given in uppercase letters. This is merely because it is easier to read them on the page. But unless otherwise specified you can type either upper or lowercase (i.e., *either* CTRL c (↑c) *or* CTRL C (↑C) — *the computer won't care which you use.*]

After typing CTRL C (↑C)*, wait for the screen to change. It will say something like:

```
MIT-OZ Artificial Intelligence Laboratory, TOPS-20
There are 34+6 jobs and the load av. is 1.75
```

Then the @ ("atsign") will appear at the left margin, thus:

@

The atsign indicates "top level"; from this level you start "jobs" or "programs." (Another commonly used term for top level is EXEC.) But first you have to log in.

To log in, type your "login" name: in this case, NCG. Next type a space, and then your password: NCG. The password does not appear on the screen, so that anyone watching what you are doing, either over your shoulder or at another terminal, will not be able to log in and change or read your files. Now type a carriage return—hit the RETURN button at the right of the keyboard. [The abbreviation for RETURN is <CR>. This symbol will be used throughout the rest of this manual.]

NCG is the name of an *ACCOUNT* which has been especially set up for people using OZ for the first time. As a regular user of OZ you will have your own account, with your own login

*This may not work on all terminals. There are actually several ways to make this first login step, depending on the terminal you're using. If CTRL C (↑) doesn't work, you should ask someone to show you the correct sequence for your particular terminal.

name and password. Even if you already have your own account, you should log in as NCG anyway.

Something like this should appear on the screen:

```
JOB 46 on TTY 26 7-Jun-84 10:52
Last login: 6-Jun-84 9:12
[Continuing]
[No messages]
[There are messages]
End of Login.cmd.4

@
```

What appears on the screen may be more complicated than this, and it may not all fit on the screen at once. If this is so, at the bottom of the screen it will say --Pause-- . Hit the space bar to see the next screenful. Do this until the @ sign appears at the left of the screen.

2.1 Conventions to Remember

<CR> (carriage return) means RETURN

$ means ESC

↑ means CTRL

If at the bottom of the screen it says --Pause-- or --More--, press the space bar to see the next screenful.

3.0 The Directory

In logging in, you have identified yourself to OZ as a user named NCG. As such, you have a certain amount of memory space allotted to you, and you have a *DIRECTORY* in which your *FILES* are listed. To look at the NCG directory type

```
DI <CR>
```

Something like this will appear on your screen:

```
    PS:<NCG>
EMACS.VARS.1
LEARN.TXT.3,4
LOGIN.CMD.2
LOGOUT.CMD.1
IDIOT'S-GUIDE.ERRATA.1,2
MM.INIT.2
   LST.1; OFFLINE
NCG.BABYL.2
TEXT82.TEXT.1
WELCOME.DOC.1

Total of 10 files.
```

The directory will not look exactly like this, since it changes each time you store or remove something from OZ's memory. Each file name has three parts, separated by dots, and the numbers after the second dot are the *version numbers* of the file.

4.0 Copy a File

In using this manual, you'll need to make changes in a file which already exists in OZ's memory. In order for the file to remain unchanged so that other people can use it, your first job will be to *COPY* it. This will preserve the original version, and give you your own version to work on. In the process of copying it, you should also give it a new name.

The file is now called `LEARN.TXT`. In copying and renaming it, use your initials plus `.TXT`. For example, you might call it `WBG.TXT`.

With the @ at the left of the screen, type

```
COPY LEARN.TXT [YOUR INITIALS HERE].TXT <CR>
```

For example, if your initials are WBG, type

```
COPY LEARN.TXT WBG.TXT <CR>
```

the screen will say

```
LEARN.TXT.25 WBG.TXT.1 [OK]
```

In the following pages this file will be referred to as `WBG.TXT`; but our assumption will be that you have renamed it with your own initials.

[NOTE: In the event that you already have your own account and are logged in as someone other than `NCG`, you can copy the `LEARN.TXT` file into your own directory by typing `COPY <NCG>LEARN.TXT <YOUR LOGIN NAME>LEARN.TXT <CR>`.]

5.0 Text Editing

Make sure you are at *top level* (the *EXEC*)—that is, you have the @ at the left of the screen. Top level can be thought of as the starting place from which you call up other programs, such as those which edit text or send mail. If for some reason you are not at top level, type CTRL C CTRL c (↑C ↑C). That should get you back there.

Let's assume that that what you want to end up with, the text you want printed on paper, will look like this:

LEARNING EMACS

A File to Practice With

Did you ever chance to hear the midnight flight of birds passing through the air and darkness overhead, in countless armies, changing their early or late summer habitat? It is something not to be forgotten. A friend called me up just after 12 last night to mark the peculiar noise of unusually immense flocks migrating north (rather late his year.) In the silence, shadow & delicious odor of the hour, (the natural perfume belonging to the night alone,) I thought it rare music. You could *hear* the characteristic motion—once or twice "the rush of mighty wings," but oftener a velvety rustle, long drawn out—sometimes quite near—with continual calls and chirps, and some song-notes. It all lasted from 12 till after 3. Once in a while the species was plainly distinguishable; I could make out the bobolink, tanager, Wilson's thrush, white-crowned sparrow, and occasionally from high in the air came the notes of the plover.

— *Walt Whitman*

The file you already copied and renamed with your initials will—once you have made certain corrections in it—produce this *OUTPUT*.

To look at this file, and to work on it, you need to call up the EMACS program. To do this, type

```
EMACS <CR>
```

If the screen answers you with an Eh? or other question, be sure you typed the command exactly. If you didn't, just type it again.

At the top of the screen it will say:

```
EMACS Editor, version 163 -- type ↑↼ (the help character) for help.
```

You have called up the EMACS text editing program. Another way of saying it is that you have an EMACS "job."

NOTE

> While you are using EMACS, it is possible that something will go wrong and things will appear on the screen which are meaningless to you. When this happens, first type CTRL L (↑L); this command clears the screen of any system messages or extraneous characters which, for whatever reason, might have accumulated. If this does not help, type CTRL X CTRL S (↑X ↑S) to *save* any text changes you've made so far; then go back to top level by pressing CTRL Z CTRL Z (↑Z ↑Z). Then type LOGOUT <CR>, which will log you out. Now log in again and start the whole process over. If this doesn't work, try CTRL C CTRL C (↑C ↑C). If even this fails, ask someone nearby for help.

Now, to continue with EMACS.

Type:

CTRL X CTRL V (↑X ↑V) (You can do this by holding the CTRL button down while you press X V).

At the bottom of the screen it will say:

```
Visit File(Default PS:<NCG>GAZONK.DEL.0):
```

The *CURSOR*—either a little blinking line or a little blinking rectangle—will be just to the right of the colon. Type in the name of the file you wish to visit. For example,

```
WBG.TXT <CR>
```

On the screen will appear:

```
\input tbase     %-*-Tex-*-
\input paper
\hehe_ten

\ctrline{\bf\twelvepoint LEARNING EMACS}

\vskip 10mm

\ctrline{\bf A File to Practice With}

\vskip 5mm

\single_space

Did you ever chance to heer the midnight flight birds passing through
the air and darkness overhead, in countleast armies, changing their
early or late summer habitat?  It is something not to be 44xforgotten.
A friendcalledme up just after 12 last night to mark the peculiar
noise of unusually ikmmense flocks migrating north (rather late this
year.) In the silence, shadow & delicious odor of the hour, (the
natural belinging to the knight alone,) I thought it rare
garbage.  You could {\it hear} the characteristic motion---once
or twice ''the rush of mighty wings,'' but oftener a velvety rustle,
long drawn out---sometimes quite near---with continual calls and
chips, and some song-notes.  It all lasted from 12 till after 3.  Once
in a while the species ws plainly distinguishable; I could make out
the bobolink, tanager, Wilson's crazy thrush, white--crowned sparrow,
and occasionally from high down in the air came the stones notes of
the plover.

\hbox to gize{\hfill {\it ---Walt Whitman}}

\end
```

You may not be able to see all of the above at once, because it may not fit on the screen. To make the rest of the text appear, use the commands CTRL V (↑V) and ESC V ($V):

CTRL V (↑V) - moves *FORWARD* one screenful

ESC V ($V) - moves *BACKWARD* one screenful

Now you are ready to edit text. The file which you are looking at contains mistakes. In order to get the output as it appears on page 5, you are going to have to correct the mistakes. Here is what the file should look like when you are finished:

```
\input tbase    %-*-Tex-*-
\input paper
\hehe_ten

\ctrline{\bf\twelvepoint LEARNING EMACS}

\vskip 10mm

\ctrline{\bf A File to Practice With}

\vskip 5mm

\single_space

Did you ever chance to hear the midnight flight of birds passing
through the air and darkness overhead, in countless armies, changing
their early or late summer habitat?  It is something not to be
forgotten.  A friend called me up just after 12 last night to mark the
peculiar noise of unusually immense flocks migrating north (rather
late this year.) In the silence, shadow & delicious odor of the hour,
(the natural perfume belonging to the night alone,) I thought it rare
music.  You could {\it hear} the characteristic motion---once or
twice ''the rush of mighty wings,'' but oftener a velvety rustle, long
drawn out---sometimes quite near---with continual calls and
chirps, and some song-notes.  It all lasted from 12 till after 3.
Once in a while the species was plainly distinguishable; I could make
out the bobolink, tanager, Wilson's thrush, white-crowned sparrow,
and occasionally from high in the air came the notes of the plover.

\hbox to size{\hfill {\it ---Walt Whitman}}

\end
```

EDITING TEXT:

There are no mistakes in the first nine lines of the file. These lines are *COMMANDS* which tell the computer how to format the text for printing. For now, ignore them.

The cursor is probably somewhere in the left upper corner of the screen. You can think of it as something like the tip of a pencil, and it is what you move around in order to make changes in text. EMACS has a variety of ways in which the cursor can be moved, but for now use these:

DEL (RUBOUT) - *DELETES* the character preceding the cursor
CTRL D (↑D) - *DELETES* the character above the cursor
CTRL F (↑F) - moves the cursor *FORWARD* one space
CTRL B (↑B) - moves the cursor *BACKWARD* one space

CTRL N (↑N) – moves the cursor to the *NEXT* line
CTRL P (↑P) – moves the cursor to the *PREVIOUS* line
CTRL V (↑V) – moves *FORWARD* one screenful
ESC V ($V) – moves *BACKWARD* one screenful

To begin editing, correct the spelling of the word "heer" in the first line:

Use CTRL N (↑N): hold the CTRL (↑) button down and advance the cursor down line by line, pressing N for each line. Then, using CTRL F (↑F) and CTRL B (↑B) put it under the second • and press CTRL D (↑D). The • will vanish. Now, type an a and it should appear on the screen.

You should be able to fix all the errors with these few commands.

The line which begins \hbox... is a *command line* which tells the computer to push the phrase -- Walt Whitman over against the right margin.

Having made the corrections, you may want to insert some words of your own. To do this, put the cursor on the line just below the words *Walt Whitman*. Type in whatever you want.

Remember that the command line \end must remain at the very end if you are going to get a printout of the file.

For now, assume you have done everything correctly and you have finished editing your file.

Type CTRL X CTRL S (↑X ↑S).

This saves your file in its present form, storing it in OZ's memory. Actually, you can save your file at any point while you are working on it; in fact, it is a good idea to save it frequently while you're working, so that in the event there is a serious system malfunction, or "crash," you won't loose the work you've done so far.

At the bottom of the screen it will say:

```
EMACS (TeX Fill) WBG.TXT (1) --BOT--
```

Again, this may not be exactly what it says—the number in parenthesis after TXT may be different, as this number indicates the number of times you have saved the file; but this is nothing to worry about.

5.1 Keys and Commands Used So Far

KEYS:

CTRL (↑) – (lower left) hold it down while you press other keys

ESC ($) – (upper left) press it and release it

<CR> – (large key at right) carriage RETURN

TOP LEVEL (@) COMMANDS:

CTRL C (↑C) – try this if trouble develops and you want to get back to top level. Try it twice if once doesn't work

KK <CR> – to log out

EMACS <CR> – calls up the EMACS program

CTRL L (↑ L) – clears screen and redisplays it.
[NOTE: this command is not limited to top level; you can use it any time.]

EMACS COMMANDS:

CTRL X CTRL S (↑X ↑S) – *saves* what you've done so far

CTRL Z CTRL Z (↑Z ↑Z) – gets you out of EMACS and back to top level (@); similar in some ways to CTRL C (↑C)

CTRL V (↑V) – moves text *FORWARD* one screenful

ESC V ($V) – moves text *BACKWARD* one screenful

DEL (RUBOUT) – *DELETES* the character preceding the cursor

CTRL D (↑D) – *DELETES* the character above the cursor

CTRL F (↑F) – moves the cursor *FORWARD* one space

CTRL B (↑B) – moves the cursor *BACKWARD* one space

CTRL N (↑N) – moves the cursor to the *NEXT* line

CTRL P (↑P) – moves the cursor to the *PREVIOUS* line

CTRL V (↑V) – moves *FORWARD* one screenful

ESC V ($V) – moves *BACKWARD* one screenful

Other *EMACS* commands not mentioned so far, but good to know:

ESC F ($F) – moves the cursor FORWARD one word

ESC B ($B) – moves the cursor BACKWARD one word

ESC < ($<) – moves to the top of your text file

ESC > ($>) – moves to the bottom of your text file

CTRL K (↑K) – erases one line. Pressing it *twice* erases one line and the space the line occupied.

6.0 Printing It Out

Having saved the file, type CTRL Z CTRL Z (↑Z ↑Z). An @ should appear.

Something important to understand is that in order to get the text you have been editing into the computer and back out in the form of printed "output," you will be using two "jobs" or "programs." One is EMACS, the program you just finished using. EMACS enables you to insert letters and words and numbers into the computer's memory, and it makes it very easy to change them around. But in order to get these words and numbers onto paper you have to use TeX.

TeX is a program which reads the file you have prepared with EMACS, then translates it into a set of instructions which will cause the Dover, or printer, on the ninth floor of Building NE43 to print out your text the way you want it. In order to do this correctly, TeX must be given certain instructions, or *commands*.

[NOTE: Don't be confused by two uses of the word *COMMAND*: the first is a *COMMAND* you make directly to the computer by pressing certain keys, such as CTRL C (↑C); the other is a *TeX COMMAND*, or *COMMAND LINE*, which you type into your text file, so that when the TeX program reads your file it will print your text on paper the way you want it.]

A command to TeX always begins with a *backslash* (\). The series of lines at the beginning of the file WBG.TXT are made up of such commands.

TeX is a very clever program, and if, in the course of translating your file for the printer, it has to stop because you have typed in a faulty command, it will send you an *error message.* It isn't always possible to tell exactly what is wrong from TeX's error messages, but with practice you can usually figure it out.

Type TEX WBG.TXT <CR>

There will be a delay, then something like this will appear:

```
MIT TOPS20 TEX 6.6 of March 28, 1982
Outputs PRESS files

*(wbg.txt.1
  TEX:tbase.TEX 1
    (TEX:first.TEX 1)
    (TEX:lib.TEX 1 2 3 4 5)
    (TEX:math.TEX 1)
    (TEX:date.TEX 1)2 3 4 5 6 7)
 (TEX:paper.TEX 1 2 3 4)
 (TEX:fntmac.TEX 1 2 3 4 5 6)
 (TEX:hehe.TEX 1
    (TEX:hex10m.TEX)
    (TEX:18s.TEX ))
 (TEX:cmm10m.TEX 1)
 (vismac.TEX 1) [1]

End of SAIL execution
 @
```

It may take a while for all of this to appear on your screen; sometimes OZ is very slow.

If this is what appears on your screen, it means TeX has checked your file, found it to be okay, and created a *press file*. If anything other than `End of SAIL execution` appears on your screen it means TeX has discovered an error. In this case turn to page 15, "Errors."

But if TeX has okayed your file and made a press file, you have only to send it to the Dover to be printed.

Type:

```
DOVER WBG.PRESS <CR>
```

If all parts of the system are working smoothly—and very often they aren't—the screen will say something like:

```
[Opening CHAOS connection to MC ... OK]
Sending ps:<NCG>wbg.press.1 to MC via CHAOS net.
Length is 2 disk blocks
Disc blocks sent:... -Done.
       1  2 - Done
28672 bits in 2.397 secs; 11.961 K bits/sec.
Spooling file is $NCG 1.
Queue entry is -QUEUE 24.
[CHAOS connection to MC closed]

 @
```

Your file has been sent to the Dover for printing. Often there is a queue of files waiting to be printed by the Dover, and that is what the line on the screen beginning `Queue entry is...` means. It tells you where your file is in the queue. A way to find out more closely what's going on with the queue is to type, at top level, `DVRQ <CR>`. If the Dover is working you will get a message on your screen like this:

```
Dover spooler TARAKA DVRSPL is in operation.
Last file sent to Dover at 10:50am  4/07.
The current time is 10:50am 4/07.
$IWASA    1          15K 10:47am 4/07  IWASA   -QUEUE 20
$GROG     2           5K 10:45am 4/07  GROG    -QUEUE 21
$POGGIO   3           7K 10:40am 4/07  POGGIO  -QUEUE 22
$JOSEF    4          15K 10:32am 4/07  JOSEF   -QUEUE 23
$NCG      5           3K 10:30am 4/07  NCG     -QUEUE 24

Number of files = 5; Total size = 45K
```

If things haven't gone this smoothly, don't be alarmed. OZ and the Dover are amazingly clever, but they're also complicated, and they rarely work for very long without something going haywire.

OZ can stop working without giving you any notice—which is why it is good, when in EMACS, to save your file [`CTRL X CTRL S` (↑X ↑S)] frequently. If suddenly the screen seems "stuck" and nothing you do on the keyboard makes anything happen, there's nothing to do but wait. To make sure it's not something you've done, you might find another terminal and see if the same thing is happening there. Eventually OZ will begin working again, but it might be minutes or days before it does.

The Dover is frequently "down" (jargon for "not working"), and this can result in a variety of messages appearing on your screen when you try to send your file. The simplest is

```
Host is not up.
```

There is nothing to do but wait until the host is "up." You can from time to time go to your terminal, log in and type `DVRQ <CR>`. Sometimes the message will say the Dover is broken and will give you a rough idea when it will be repaired. When it is again up, there will usually be a message to that effect.

Sometimes a message will tell you that attempted transmission over the "Chaosnet" has failed. The Chaosnet is a network of wires linking many MIT computers. You may be offered a list of choices, such as:

```
Sorry, MC says the Dover is broken.

Options are:
  A - ARPAnet to MC
  C - CHAOSnet to MC
  D - Direct to Dover
  F - to a disk file
  Q - to Quit

Type a character:
```

If this happens, try D. Your file may get through to the Dover. Or, the message may be repeated. You then might try Q, which should bring you back to top level, at which point you can try to send the file again. Failing all this, you might conclude that something's wrong with the Dover and wait for a while and try it later.

Another problem might be that your file goes to the Dover, appears on the queue, and then the queue stops advancing: something has gone wrong, but there is no message on your screen telling you what it is. Again, you have to wait.

What you should bear in mind here is that the Dover is often broken. Once you've made a press file, it does not mean you can get it printed right away. Usually you can, but if you are going to be working against a deadline you should try to figure into your plans the very real possibility that a day or more may go by without getting output from the Dover.

If you've run a file through TeX but can't get it printed by the Dover, you simply must wait for the Dover to resume working. Before you log out, turn to page 16, "Cleaning Up Your Directory."

6.1 TeX and Dover Commands Used So Far

TEX [NAME OF FILE] <CR> – tells TeX to check your file and make a *press file*

DOVER [NAME OF FILE] <CR> – sends your *PRESS FILE* to the Dover

DVRQ <CR> – shows you the Dover queue and tells you if the Dover is working

7.0 Errors

It's very likely that things won't go so smoothly when you have TeX look over your file. Dealing with TeX's error messages can be frustrating, but it is something you get better at with time; for now let's look at an example of one situation.

If in making the corrections in the `WBG.TXT` file you didn't fix the line beginning `\hbox...` you may get an error message like this:

```
MIT TOPS20 TEX 6.6 of March 28, 1982
Outputs PRESS files

*(wbg.txt 1
   (TEX:tbase.TEX 1
     (TEX:first.TEX 1)
     (TEX:lib.TEX 1 2 3 4 5)
     (TEX:math.TEX 1)
     (TEX:date.TEX 1) 2 3 4 5 6 7)
   (TEX:paper.TEX 1 2 3 4)
   (TEX:fntmac.TEX 1 2 3 4 5 6)
   (TEX:hehe10.TEX 1
     (TEX:hex10m.TEX 1)
     (TEX:hex18s.TEX 1))
   (TEX:cmm10m.TEX 1)
   (vismac.TEX 1)
! Illegal unit of measure (pt inserted).
<to be read again>
                    i
<to be read again> i
                     z
p.1,l.32 \hbox to giz
                       e\hfill \it --- Walt Whitman

↑
```

The most important thing about this error message is that it is telling you where you made your mistake. The location of the mistake appears near the bottom of the error message: `p.1, l.32`. This means *line 32* of the `WBG.TXT` file. `p.1` means page one, but error messages always say page one—so the line number means *number of lines from the very beginning of the file.* In this case the file is small, and it is easy to find the place where the error occurred. With a large file you can go back into EMACS, put the cursor at the very beginning of the file, and type `CTRL U (↑U) [THE LINE NUMBER] CTRL N (↑N)`, then `CTRL P (↑P)`. For example, to go to line 32 type `CTRL U (↑U) 32 CTRL N (↑N) CTRL P (↑P)`.

This error message says that on line 32 something happened that TeX couldn't handle. Notice the last two lines of the error message. The word "`gize`" is broken, the letters `giz` on one line and the letter `e` followed by `\hfill \it --- Walt Whitman`. The break takes place in the word where TeX ran into trouble: `gize` should be `size`. The easiest way to deal

with this for now is to get out of the TeX job, get back into EMACS, fix the file, then run it through TeX again.

At the left of the screen, at the end of your error message you will see a small up arrow (↑). Type X. The x will appear next to the ↑. A message on the screen will tell you to type X again:

```
(↑x
Type x again to exit:
No output file.)
```

So, type X again. The result should be:

```
No output file.
End of SAIL execution.
```

This will put you back at top level, giving you the @ sign. Type CON EMACS <CR> (short for CONTINUE EMACS). Using the edit commands discussed earlier, change the word gize to size. Then proceed as before to have TeX check out your file. If you forget how to do this, turn back to page 11.

You must keep repeating this process until you get TeX to okay your file.

8.0 Cleaning Up Your Directory

Assuming you have corrected the errors in the WBG.TXT file and sent your press file to the Dover, you are almost ready to log out. It is a good idea, before logging out, to delete any extra copies of files you don't need. It isn't fair to other users of the computer to take up scarce memory space with more copies of your file than you need.

At top level, type DI <CR> (short for DIRECTORY).

On the screen will appear a list of files in the NCG directory. It will look like this:

```
    PS:<NCG>
EMACS.VARS.1
LEARN.TXT.3
LOGIN.CMD.2
LOGOUT.CMD.1
IDIOT'S-GUIDE.ERRATA.1
MM.INIT.2
   LST.1; OFFLINE
NCG.BABYL.2
TEXT82.TEXT.1
WBG.TXT.1,2,3,4
   .ERR.1
   .PRESS.1
WELCOME.DOC.1

Total of 10 files.
```

The directory will of course not look exactly like this, since it is a list of files and thus changes almost every time you log in. But in the list you will see the WBG.TXT file followed by the number of each version now stored in the computer. Unless you have a special reason, there is no need to keep more than two copies of a file. More than this takes up valuable space in memory. So, you should DELETE and then EXPUNGE all but two copies.

The reason you have to DELETE and then EXPUNGE is that you must go through two operations before removing a file, thereby giving you a chance to reconsider and lessening the risk that you'll erase something you didn't mean to.

Suppose the listing for the WBG.TXT file in the directory looks like this:

```
WBG.TXT.9,10,11,12
```

You need only save the last two versions, 11 and 12. Thus you can get rid of 9 and 10. So, type

```
DELETE WBG.TXT.9 <CR>
```

On the screen will appear

```
WBG.TXT.9 [OK]
```

This means that the WBG.TXT.9 file has been DELETED. It is still not irrevocably gone. To get it back you can type

```
UNDELETE WBG.TXT.9 <CR>
```

Assuming you have deleted WBG.TXT.9 and you are sure you want to erase it for good, it's time to EXPUNGE it.

Type EXPUNGE <CR>

This command expunges *everything* you have deleted—so it is always wise to be sure you want to do an expunge. Your screen should say

```
PS:<NCG> [1 page freed]
```

You have "freed up" one page of space in memory.

Now that you know how to delete and expunge, you should continue to get rid of all but the last two WBG.TXT files, the WBG.PRESS file, and the WBG.ERR file. The ERR file contains any error messages TeX wrote while processing your file. Press files take up a lot of space, and since they can be created again from the text file, it is a good idea to delete them once you've gotten the output.

Files should be deleted one at a time, but you can expunge all the deleted files at once.

[NOTE: There is a faster way to delete the text files. While in your EMACS job, type ESC X ($X), then type REAP <CR>. A message on the screen will ask if you want to delete all but two files. Type Y for yes. You must still go to top level and expunge these deleted files.]

8.1 Commands Used for Directory Cleanup

DI <CR> – Shows you your directory

DELETE [NAME OF FILE] <CR> – DELETES the file specified

UNDELETE [NAME OF FILE] <CR> – UNDELETES, i.e, gets back a deleted file

ESC X ($X), then type REAP <CR> – *(works only in EMACS)* Deletes all but two copies of file. You must also go to top level and EXPUNGE

EXPUNGE <cr> – EXPUNGES all deleted files

9.0 Killing Jobs and Logging Out

Now that you have cleaned up your directory and eliminated unnecessary files, you should kill the various jobs you have running.

Type INF FO <CR>.

This is short for INFORMATION ABOUT FORK-STATUS. "Fork" is another term for "job." The screen may display your jobs thus:

```
   EMACS (1): Kept, HALT at 51364, 0:00:41.2
=>TEX (2): Kept, HALT at 51364, 0:00:44.4
```

TeX is of course the job you used to make your press file, and EMACS is your word processing program. The little arrow (=>) to the left of TeX indicates the job you were using most recently.) You can kill various jobs one at a time or all at once. Either way, you use the RESET command. "Reset" means "kill." To kill the TeX job type

```
RESET TEX <CR>
```

or, to kill the EMACS job type

```
RESET EMACS <CR>
```

To kill all your jobs at once, type

```
RESET * <CR>
```

The * is a "wildcard" and is useful in other ways, as you'll discover.

Having killed your jobs and cleaned up your directory, you're ready to log out. Type either `LOGOUT <CR>` or `KK <CR>`.

You are now LOGGED OUT.

9.1 Job-Killing and Logout Commands

`INF FO <CR>` – shows the jobs you have running
`RESET [NAME OF JOB], or *` – kills one or all jobs
`KK <CR>` – logs you out

10.0 Creating a File from Scratch

Now that you're familiar with editing a simple file in EMACS, you'll want to know how to create a file of your own. This is also a good time to learn a few more EMACS commands.

Assuming you're logged in, get an EMACS job (`EMACS <CR>`). Now, when

```
EMACS Editor, version 163 -- type ↑+ (the help character) for help.
```

appears at the top of the screen, type—as you did when you wanted to get the `WBG.TXT` file — `CTRL X CTRL V` (↑X ↑V). And when

```
Visit File(Default PS:<NCG>GAZONK.DEL.0):
```

appears at the bottom of the screen, type the name of the file you wish to create. You can name it whatever you want, but keep the title fairly short. Also, it should end with `.TXT`. So, let's say you decide to call the new file `SCRATCH.TXT`. That is what you type:

```
SCRATCH.TXT <CR>
```

The bottom of the screen will say:

```
EMACS (Fill Text) SCRATCH.TXT (0)
(New File)
```

Thus far the file exists only in EMACS. Note the version number (0) at the end of the line. When you save the file [CTRL X CTRL S (↑X ↑X)] for the first time the (0) will change to (1) and you will have created a file. It will thereafter be listed in your directory.

You'll remember the list of commands which appears at the beginning of the WBG.TXT file, and how these commands were necessary to get the file printed out. In beginning a new text file, you have to type in similar commands.

You should begin with the minimum needed to get output from the Dover. As you become more experienced with EMACS and TeX you'll want to experiment with some of the many possible commands and instructions, all of which allow you to change the file according to your own wishes, and—usually—make your editing work easier.

Take it line by line. First, type:

```
\input tbase      %-*-Tex-*-
```

This tells the computer to *input* the file "Tbase" (i.e., get the "Tbase" file and stand ready to use it). "Tbase" is a large file of instructions which TeX will use in formatting your file. The spacing between `tbase` and `%` isn't critical, but for now use five spaces. The line `%-*-Tex-*-` sets up your file so that every time it is called up by EMACS it will automatically check, whenever you type a parenthesis or brackets, to make sure that you've balanced a left bracket with a right. This can be very important for TeX commands, as you'll see.

[NOTE: Since you are now typing TeX *command lines* (i.e., you are not issuing commands directly to OZ) upper and lowercase letters are not interchangeable. You should type exactly what appears on the page.]

The next line, `\input paper`, inputs another necessary file.

The line `\hehe_ten` tells TeX what size and what type families you want to use: in this case, helvetica, the type used in this manual, in ten point size. Here again, once you get some proficiency, you can decide for yourself how you want to change these commands.

That's all you need for now as far as inputting files. Now skip a space and type

```
\ctrline{The Scratch File}
```

This tells TeX to center whatever you type between the brackets.

To put some white space on the page between the title and whatever follows, type

```
\vskip 10mm
```

This says to jump vertically 10 millimeters before printing anything.

Now let's put a subtitle over against the left margin:

```
\fdisplay{1.0 Introduction}
```

And some more white space:

```
\vskip 5mm
```

To get double spacing in the text:

```
\double_space
```

(If your terminal doesn't have an underscore (_), use a backarrow (←).

And to get *indented* paragraphs:

```
\pp
```

(\pp is short for "printer's paragraph.")

It's a good idea to leave a space between each of these commands.

You can now insert your text. Remember to end the whole file with \end.

Here is what the whole thing should look like so far:

```
\input tbase    %-*-Tex-*-
\input paper
\hehe_ten

\ctrline{The Scratch File}

\vskip 10mm

\fdisplay{1.0 Introduction}

\vskip 5mm

\double_space

\pp

[text goes here]

\end
```

As you can no doubt tell, this represents the barest beginnings of what you can type into a file in the way of commands to format printed text. However, this should be enough to get you going. Next you should begin to do two things:

1) Examine a sample of input and compare it with the resulting output. Samples of both are included with this manual. This is one of the best ways to learn what TeX can do.

2) Look at the various manuals available. A list of manuals with some comments appears on page 39. The more you learn about EMACS and TeX, the more helpful these manuals will become.

11.0 Mathematics

You may have noticed by now that TeX is not the easiest thing in the world to get the hang of. There are simpler and more "friendly" text formatting programs available on MIT computers, and many people use them. But TeX has one major advantage over any others: its ability to handle mathematical equations.

Although TeX can do virtually anything you want in the way of equations, you have to learn how to tell it what to do. What you type on the screen does not look at all like what will come out on the page. Here, for example, is an equation:

$$\theta_2 = tan^{-1}\left[\frac{x_{p1}^2/x_{p2}^2 - 1}{1 - r^2 p}\right]^{\frac{1}{2}}$$

And here is what was typed to get that equation:

```
$$\theta^A{2} = tan^K{-1}\left[{{x^K{2}^A{p1}}/{x^K{2}^A{p2}}-1}
\over{1-r^K{2}p}\right]^K{{1}\over{2}}$$
```

The best way to learn to do this kind of thing is to compare the input with the output of a paper which contains equations. Such a paper is included as part of this manual.

There are some basic principles, however, that you should learn right from the start:

- There are two kinds of Math Mode: *text style* and *display style*.
- *Text style* is when the formula is included as part of a sentence. For this, the equation is set between single dollar signs. For example, `${1}\over{2}$` results in $\frac{1}{2}$. [NOTE: The $ symbol is not the same as pressing the ESC key. To use math mode you must type an actual $ and make it appear on the screen.]
- *Display style* is when the equation is set off from the text. Such an equation begins and ends with double dollar signs. For example,

  ```
  $${{A}\over{B}}\div{{C}\over{C}} =?$$
  ```

  results in

$$\frac{A}{B} \div \frac{C}{C} =?$$

Your basic tools for learning mathematical typing in TeX will be (1) The TeX manual contained in the book *TeX and Metafont* (for information on this book see page 39, "List of Manuals"), and (2) input and output of papers containing mathematics.

Here are a few examples of simple, commonly used math typing upon which you can build.

`$x^K{2}$` gives x^2 [Type: `$ x CTRL Q CTRL K (↑Q ↑K) {2}$`]

`$x^A{2}$` gives x_2 [Type: `$ x CTRL Q CTRL A (↑Q ↑A) {2}$`]

`$x^K{y}^A{z}$` gives x_z^y

`$\sqrt{x-1}$` gives $\sqrt{x-1}$

`$${x+b}\over{\theta - 2}$$` (display mode) gives

$$\frac{x+b}{\theta-2}$$

`$$B^A{x}^K{z}\left({\sigma + 1}\over {{1 - p}^K{2 -1}}\right)^K{2 + x}$$` gives

$$B_z^x\left(\frac{\sigma+1}{1-p^{2-1}}\right)^{2+z}$$

```
$${\hbox{Number of Closed Strings:}}= 2 + 2^K{n - 2}+\sum
^A{j = 3}^K{N}2^K{n - j}\eqno\hbox{(3).}$$
```

gives

$$\text{Number of Closed Strings:} = 2 + 2^{N-2} + \sum_{J=3}^{N} 2^{N-J} \qquad (3).$$

[NOTE: The symbols ^A and ^K which appear above are not the same as the CTRL (↑) commands used previously in this manual to tell you what keys to press. ^A is a symbol, or character, which you make appear on the screen by typing (in EMACS) CTRL Q CTRL A (↑Q ↑A). On your particular screen the up-arrow may appear either as a ↑ or a ^. The ^A(or ↑A) which appears on the screen is a command to TeX which says, "Take whatever is within the brackets immediately following and make it a subscript." The same is true for ^K(or ↑K), which you get by typing CTRL Q CTRL K (↑Q ↑K). The ^K tells TeX to make a superscript.]

12.0 Input and Output Samples

This section contains examples of input and Dover-output: that is, what was typed into the computer and what was finally printed out. This material is by no means comprehensive as far as showing the kinds of things that can be done with TeX. But by browsing through the following pages a TeX novice will hopefully begin to get a feel for the relationship between what appears on one's terminal screen and what gets printed on paper.

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
ARTIFICIAL INTELLIGENCE LABORATORY

A.I. Memo No. XYZ June 1984

A Sample Artificial Intelligence Lab Memo*

Willard Burt Groliard

Abstract. In the following pages you will find some examples of the kinds of things which can be done with Tex: section headings, figure captions, footnotes, math formulas, etc. This is in no sense an exhaustive or even-near complete demonstration of what is possible. This material has been assembled from a variety of sources, and much of it won't make much sense if you try and read it for content.

Acknowledgement. This report describes research done at the Department of Psychology and the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology. Support for this work is provided by WING and GROOVE under a combined grant for studies in Word Processing, grant 43-2198-CS. Comments and criticisms by Ian Carlet, Joe Prue and Elizabeth Browning were greatly appreciated. Technical support was kindly provided by Sir Thomas Browne.

© 1984 Massachusets Institute of Technology

*This is an example of how you can put a footnote on a title, if you want to.

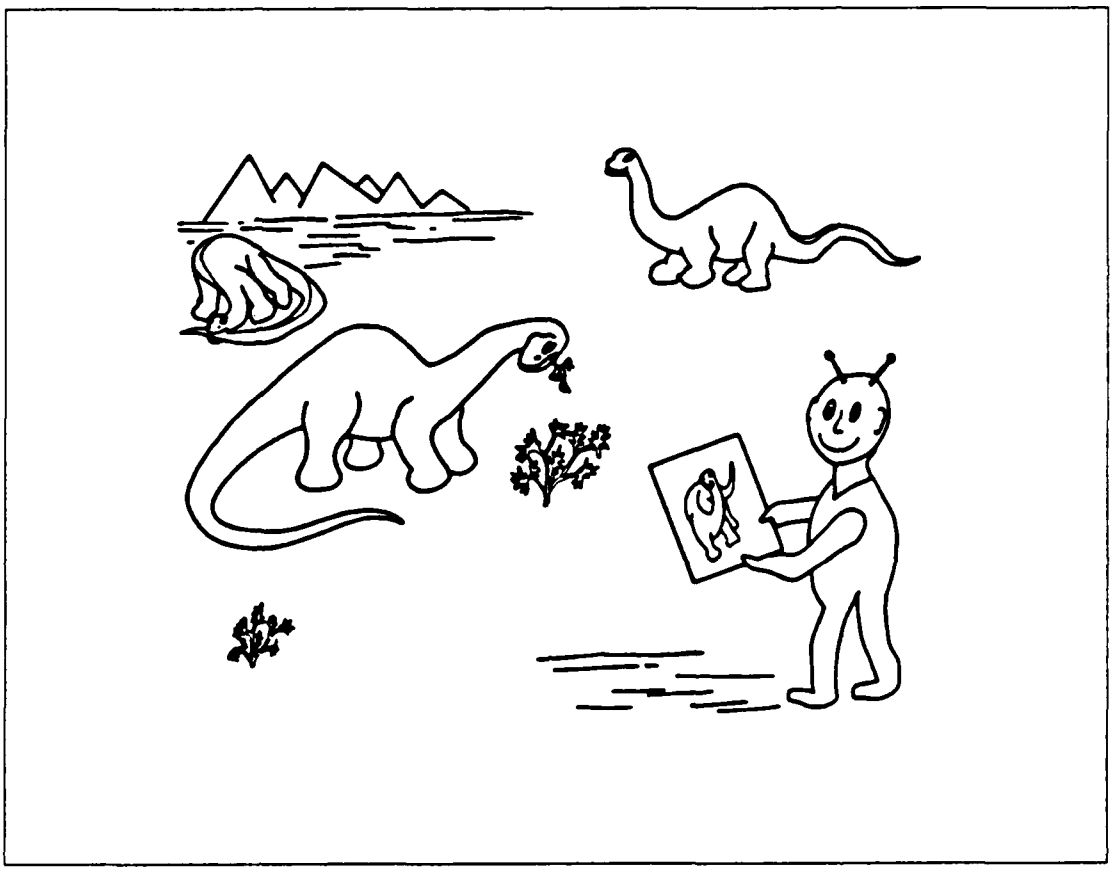


Figure 1 "These can't be dinosaurs. None of them match this picture!"

1.0 The Name of the Game

Perceiving systems are subject to a massive bombardment of signals from the external world. From this deluge of data, useful bits and pieces of information are abstracted from which intelligent decisions can be made. These information abstraction processes cannot be completely arbitrary. They clearly depend upon the goals of the system, its environment, and often upon certain expectations.

In simple environments such as many industrial settings and laboratories, the goal of the perceiver is usually quite limited and well-defined: find the "red" cube on the table, or the open-end wrench on the conveyor belt. Because the "object" of interest is known and expected in advance, simple "template" matching often suffices to solve these tasks. Examples of template-matching can also be found in natural environments: the blowfly feeds when its receptors identify the ring structure of a sugar, and rejects the hydrocarbon chains of alcohols (except Inositol, which is an unnatural ring alcohol [Hodgson, 1961]!). Or the hungry fledgling gull that responds immediately to the looming red spot on its parents' beak; the mating call of the cricket (or bee), which is so precisely engineered that a simple pattern of pulses can be tailored to reflect even subtle species differences. Such examples are countless (Tinbergen, 1951; Wilson, 1971). In each case, an important primitive goal such as feeding or the reproduction of the species, is achieved sucessfully in a very direct and reflexive manner only because the environment is limited or well controlled.

Yet how can such a simple template-matching strategy serve a more sophisticated being, who lives in a complex, changing environment? Here, surprises may often be the rule. When we look out a window, walk into an unfamiliar building, or simply view a novel

picture or postcard, we have no difficulty in grasping the meaning or context of the scene. The greater our perceptual repertoire, the larger is the spectrum of the unexpected and the variety of "things" that must be recognized and dealt with, often out of the immediate context or frame of mind. Simple template matching to prestored models then becomes impossible, for there are just too many possibilities. Even for one simple item—let's say a dinosaur—the possible views and configurations is usually an infinity in itself (Fig. 1).[1] Without some method of initializing the perceptual system, it must founder as a perceptron will (Minsky and Papert, 1965). What is needed at the outset are some low level representations or assertions that are powerful enough to capture the essence of the "event" or "thing", yet are readily and routinely computable from the raw sense data. These primitive, low level assertions will constitute the answers to our 20 Questions. What inquiries then should we ask? Under what conditions can we expect such a set of questions to provide a useful set of answers?

2.0 From Templates to Questions

In the case of lower animals which react to certain stimuli in essentially a reflexive manner, the system is preprogrammed to recognize a simple pattern. The presence of this pattern is almost guaranteed to represent an "event" or "thing" of importance to the animal.[2] The pattern is thus an attribute uniquely associated with the event of interest, given the expected context. The red spot on the beak of the gull suffices for the fledgling gull because from its nest it will almost never encounter other instances of looming red spots—such as traffic lights or red balloons. In this case a simple template-matching strategy works well because of the controlled context. A simple question suffices to make reliable assertions about a complex event, namely that a parent has arrived, presumably with food.

The situation becomes considerably more complicated, however, for a general purpose perceptual system that must respond intelligently to a wide range of events in a variety of contexts. We cannot hope to find attributes or features unique for each event of interest and for each possible contextual situation. How then can we even hope to find simple questions that will have the same power as the red spot on the gull's beak? The proposed solution is to choose the questions carefully so they inquire about the more general properties of all things regardless of context.

Consider the classical children's game of 20 Questions, where the goal is to identify an object. The first questions usually attempt to identify the general class of the object. Is it animal, vegetable or mineral? Subsequent questions attempt to determine the size, shape or mass, or the sounds "it" might make, how "it" moves, or perhaps its function. The final questions then become very specific and detailed. If we are clever and shrewd in our choices, we rapidly converge to the object. Why can't a perceptual system be designed along similar lines? Imagine that for our first set of questions we identify a dozen or two—let's say twenty—very general but independent attributes of "things". We simply ascertain whether each attribute is present or not. Then 2^{20} or roughly a million different types of events could be crudely categorized (Webster's Dictionary only lists 60,000 words total.)

---

[1] This is a footnote.

[2] This is another footnote.

In sum, we now have four major criteria for our choice of questions:

(i) *Computational Validity* – The representation of the attribute must be easy and reliable to compute.

(ii) *Conveyance* – The attribute should encode a general property of object (such as size, mass, etc).

(iii) *Viewer Independence* – Representations of attributes should be insensitive to the particular relations between the perceiver and the "object", i.e., to object distance, scale or disposition.

(iv) *Orthogonality* – Different attributes or questions should be capturing independent qualities of the "events" or "things".

• • • •

Legal Codon Triples					
Codon	0 –	0+	1 –	1+	2
0 –	0– 2	1 –	0– 2	1 –	0– 2
0+	1+	0+	1+	0+	1+
1 –	1+	0+	1+	0+	1+
1+	0– 2	1 –	0– 2	1 –	0– 2
2	0– 2	1 –	0– 2	1 –	0– 2

Table 2-3

• • • •

$$\{(0^+\,0^+)(0^+\,1^+)(1^-\,0^+)(1^-\,1^+)\} \mapsto 0^-$$
$$\{(0^-\,0^-)\} \mapsto 0^+$$
$$\{(0^-\,0^-)(0^-\,1^-)\} \mapsto 1^-$$
$$\{(1^+\,0^-)(2\,0^-)\} \mapsto 1^+$$
$$\{(1^+\,1^-)(1^+\,2)(2\,1^-)(2\,2)\} \mapsto 2$$

NOTE

Here is an example of how you can put a small paragraph in the center of the page for emphasis. You can change the size of the paragraph, put the type in **boldface** or *italic* or ***bold italic***...You can also change the size of the type...

Box 1: *Another box* – In this case you can put a word or phrase out at the left and a chunk of text boxed up to the right, and you can adjust the size of the whole thing. You'll see, however, when you look at the input, that what you type on the computer screen looks not at all like what you hope to get on the page. That is one of the main troubles with Tex—it is not "interactive." Hopefully someday it will be.

• • • •

2.0 This is a section heading

2.1 This is a subsection heading

Here are some math examples.

Proof. Let the two lines of sight from each stereo view lie in the XZ plane and intersect at 0, as shown in Fig. 3. Any point $P(x, y, z)$ can then be specified by its distance from 0 and two angles σ, τ. Because the views are orthographic, τ appears in the image plane, as does the elevation of P, namely y_p and its azimuth x_p. The problem then reduces to recovering σ or P_{xz}, the projection of $P(x, y, z)$ onto the XZ plane.

As seen from above, the projection of $P(x, y, z)$ onto the XZ plane is shown in Fig. 4. For notational convenience P_1 has replaced P_{xy} and $\theta = \frac{\pi}{2} - \sigma$. Our unknowns are thus θ_i and z_{pi}, because x_{pi} appears in the image plane.

From the fact that the length OP_i is constant over all views, we obtain

$$\overline{OP}_1^2 = \overline{OP}_2^2 = x_{p1}^2 + z_{p1}^2 = x_{p2}^2 + z_{p2}^2 \tag{1}$$

with unknowns z_{p1}, z_{p2}.

From the fact that each view is stereoscopic, we obtain the distance-disparity relation

$$\frac{\delta x_{p1}}{\delta x_{p2}} = \frac{z_{p1}}{z_{p2}} = r_p \tag{2}$$

where δx_{pi} is the measured disparity, thereby making r_p a known constant. This relation follows from the fact that the horizontal disparity of P relative to O is given by

$$\delta x_{pi} = z_{pi}(I/D^2) \tag{3}$$

where I is the interpupil distance and D is the line of sight distance to O, and given that the distance OP is much smaller than D. Taking the ratio of (3) for $i = 1, 2$ eliminates the (I/D^2) dependency.

• • • •

$$F * I(\underline{n}) = \int F(\underline{n} - \underline{\varsigma}) I(\underline{\varsigma}) d\underline{\varsigma}.$$

• • • •

$$0 = \int_{-\infty}^{\infty} f''\left(\frac{x - zeta}{\sigma}\right) I\varsigma) d\varsigma. \tag{1}$$

• • • •

$$\frac{dx}{d\sigma} = \frac{\int_{\infty}^{\infty} \left(\frac{x-zeta}{\sigma}\right) f'''\left(\frac{x-\varsigma}{\sigma}\right) I(\varsigma) d\varsigma}{\int_{-\infty}^{\infty} f'''\left(\frac{x\varsigma}{\sigma}\right) I(\varsigma) d\varsigma} \tag{2}$$

• • • •

$$\begin{pmatrix} T(x_o - \varsigma_1) & \dots & T(x_o - \varsigma_n) \\ T_x(x_o - \varsigma_1) & \dots & T_x(x_o - \varsigma_n) \\ T_{xx}(x_o - \varsigma_1) & \dots & T_{xx}(x_o - \varsigma_n) \\ T\sigma(n_o - \varsigma_1) & \dots & \sigma(n_o - \varsigma_n) \end{pmatrix} \begin{pmatrix} A_1 \\ \cdot \\ \cdot \\ A_n \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ -\ell^2 \\ 1 \end{pmatrix} \tag{20}$$

• • • •

Appendix I: Example Games

Here's an example of one way to align text into columns.

GAME 1

Habitat: (previously determined to be temperate environment, green rolling hills. Elevation of "thing" is on the ground.)

	ANIMAL	PLANT	MINERAL
Q1: Is it moving? *Implication*: It's an animal in motion.	*translates*	sway	no
Q2: How many supports? *Implication*: Confirms animal — has four "legs".	*2,4 or >4*	1	0
Q3: What acoustic frequencies are emitted? *Implication:* Disconfirms animal. "Thing" makes low frequency, broad-band sounds, moves and has 4 legs. Must be big. Elephant or cow?	narrowband	*broad*	*broad*
Q4: Acoustic Source *Implication:* Confirms "animal" or isolated object.	*point*	extended	extended

An example of a numbered list:

Appendix II

Documentation of devices that can provide answers to each of the twenty questions.

1. **Acoustic Frequency.** Comb filtering has been used for several years to separate sound sources (Shields, 1970; Flanagan, 1972; Zwicker et al., 1979). Unless many broad-band sources are active simultaneously at S.P.L.'s comparable to the narrowband sources, this question can be answered with available technology (Klatt, 1977). As initially formulated (Richards, 1980), the question simply addresses whether the source is broad-band or not (such as wind through the trees, rushing water, or an animal cry). Much more useful but also much more difficult, would be to extract the physical properties of the source — i.e., its acoustic "color": Is it metallic, wood striking wood, or a footfall?
2. **Acoustic Modulation.** Tracking a sound source to determine its modulation characteristics (Atal, 1972) also requires localization (as may Question #1). For narrow-band, harmonic sources with different spectral signatures, such localization is possible provided there are only a few competing sources (Altes, 1978). Again, as in Question #1 work should be undertaken to understand how the "textural" properties of the source can be extracted from the modulations. For example, is the source "harsh" or grating, or like clacking sticks, or "suave " and "smooth", or "roaring" like a brook or lion.
3. **Frequency Change.** Here again, as in Questions #1 and #2 localization is helpful but not as necessary because only ANIMALS are generally capable of producing sounds of variable frequency. Simple 1/3 octave filtering should allow the detection of frequency change (Flanagan, 1972; Klatt, 1977.)
4. **Motion.** The motion of an "object" can be both visual and auditory. Clearly the detection of auditory movement requires localization (Altes, 1978; Searle et al., 1980), and may be difficult. Visual motion detection has progressed enormously over the past ten years, and can be detected with simple systems provided the background is stationary (Horn and Schunck, 1981; Thompson, 1981; Ullman, 1981; Hildreth, 1982). More work is still required, however, to use motion to segregate a visual scene, especially if sway or scintillation is to be disambiguated from translation or rotation.
5. **Support.** Although a powerful question, to estimate the numbers of "legs" supporting a region is quite complicated. First, the ground plane must be determined (see Question #20), secondly the candidate "support" must be recognized (e.g., leg or trunk) and finally a region should be identified as being supported although it may have a different color or texture. In the case of stationary supports, the local parallelism of the vertical occluding edges of the support may serve as a basis for determining the supporting member (Stevens, 1980). What to do in the case of animal motion, however? Also, shrubs clearly may have many "supports". The computational validity of this attribute is questionable, therefore, although a strong assertion would be quite useful.

• • • •

One way in which references can be formatted:

REFERENCES

Agin, G., Representational description of curved objects. *Stanford AI Project Memo AIM-173, Stanford University, 1972.*

Altes, R.A. Angle estimation and binaural processing in animal echolocation. *J. Acoust. Soc. Amer.*, 1978, *63*, 155-183.

Atal, B.S. Automatic speaker recognition based on pitch contours. *J. Acoust. Soc. Amer.*, 1972, *52*,1687-1697.

Bajcsy, R. and Badler, N., *Representation of Three-Dimensional Objects.* New York: Springer-Verlag, 1982.

Ballard, D.H. and Brown, C.M. *Computer Vision.* New Jersey: Prentice-Hall, 1982.

Barbe, D.F., Smart sensors. *Proc. Soc. Photo-Opt. Instrum. Eng.*, 1979, *178.*

Benedict, R.P., *Fundamentals of Temperature, Pressure, and Flow Measurements,* New York: Wiley, 1969.

Binford, T., Inferring surfaces from images. *Art. Intell.*, 1981, *17*, 205-244.

Boden, M., *Artificial Intelligence and Natural Man.* New York: Basic Books, 1975??

Chance, J.E. and LeMaster, E.W., Suits reflectance models for wheat and cotton: theoretical and experimental tests. Applied Optics, 1977, *16*, 407-412.

Cook, R.L. and Torrance, K.E., A reflectance model for computer graphics. *ACM Trans. on Graphics*, 1982, *1*, 7-24.

Cox, C.P. and Baron, M., A variability study of firmness in cheese using the ball-compressor test. *Journal of Diary Research*, 1955, *22*, 386-390.

Davis, L. and Rosenfeld, A., Computing processes for low level usage: a survey. *Art. Intell.*, 1981, *17*, 245-263.

Dirlam, D.K., Most efficient chunk sizes. *Cogn. Psych.*, 1972, *3*, 355-359.

Flanagan, J.L., *Speech Analysis: Synthesis and Perception.* Berlin: Springer-Verlag, 1972.

Francis, F.J. and Clydedale, F.M., *Food Colorimetry: Theory and Applications.* Westport, CT: AVI Publishing, 1975.

Grimson, W.E.L., *From Images to Surfaces.* Cambridge, MA: MIT Press, 1981.

Harmon, L.D., Automated tactile sensing. *Int. J. Robotics Res.*, 1982, *1 (2)*, 3-32.

Harmon, L., The recognition of faces. *Sci. Amer.*, 1973, *229*, 70-82.

Herzfeld, C.M., *Temperature, Its Measurement and Control in Science and Industry, vol. 3.* New York: Reinhold, NY, 1962.

Hildreth, E., The integration of motion information along contours. *IEEE Proceedings of a Conference of Computer Vison Representation and Control, Sept., 83-91, 1982.*

Hillis, W.D., A high resolution image touch sensor, *Int. J. Robotics Res.*, 1982, *1 (2)*, 33-44.

```
\input tbase    %-*-Tex-*-
\input paper
\hehe_ten
\input vismac                   % this is a file containing handy
                                % macros -- for figures and captions,
                                % etc.  Its full OZ address is
                                % ps:<whit>vismac.tex

\def\titlerays#1{\hbox{\raise5.2pt\hbox{$#1$}}}
\def\str{\ctrline{*\ \ \ \ *\ \ \ \ *\ \ \ \ *}}

%Time derivative in dot notation.
\def\dot#1{\spose{\raise 2.667pt\hbox{\char'56}}{#1}}

%Double time derivative in dot notation.
\def\ddot#1{\spose{\raise 2.667pt\hbox{\char'56 \char'56}}{#1}}

%Set up a vector font
\def\vec#1{\hbox{\bf #1}}

%Vector represented by underline
%for those characters not available in bold
\def\vec#1{\underline{#1}}

\author_title_format{\bf{WBG}}{\bf{SAMPLE}}

\defstr\print_page_one_headings{T}

\title_page

\vskip 3mm

\cdisplay{\tenpoint
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
ARTIFICIAL INTELLIGENCE LABORATORY}

\vskip 0.4 in

\ctrline{A.I. Memo No. XYZ \hfill June 1984}

\vskip 0.32 in

\ctrline{\fourteenpoint \bold_face A Sample Artificial Intelligence
Lab Memo\titlerays{*}}

\vskip 0.35 in

\ctrline{\twelvepoint\bf
Willard Burt Groliard}

\vskip 30mm

\single_space

\fp
{\bf Abstract.}\ In the following pages you will find some examples of
the kinds of things which can be done with Tex: section headings,
figure captions, footnotes, math formulas, etc.  This is in no sense
an exhaustive or even-near complete demonstration of what is possible.
This material has been assembled from a variety of sources, and much
of it won't make much sense if you try and read it for content.

\vskip 30mm

\fp {\bf Acknowledgement.}\ This report describes research done at the
Department of Psychology and the Artificial Intelligence Laboratory of
the Massachusetts Institute of Technology.  Support for this work is
provided by WING and GROOVE under a combined grant for studies in Word
Processing, grant 43\~ 2198\~ CS.  Comments and criticisms by Ian
```

Carlet, Joe Prue and Elizabeth Browning were greatly appreciated. Technical support was kindly provided by Sir Thomas Browne.

```
\bottom_note{$^{*}$This is an example of how you can put a footnote on
a title, if you want to.}

\vskip 15mm

\fp $\copyright$ 1984 Massachusets Institute of Technology

\bp

\cdisplay{\bf 1.0 The Name of the Game}

\penalty 1000

\vskip 4mm

\penalty 1000

\pp

Perceiving systems are subject to a massive bombardment of signals
from the external world.  From this deluge of data, useful bits and
pieces of information are abstracted from which intelligent decisions
can be made.  These information abstraction processes cannot be
completely arbitrary.  They clearly depend upon the goals of the
system, its environment, and often upon certain expectations.

\fp
\boxfig{5.75}{4.5}{1}{''These can't be dinosaurs.  None of them match
this picture!''}

\pp
In simple environments such as many industrial settings and
laboratories, the goal of the perceiver is usually quite limited and
well-defined: find the ''red'' cube on the table, or the open-end
wrench on the conveyor belt.  Because the "object" of interest is known
and expected in advance, simple ''template'' matching often suffices
to solve these tasks.  Examples of template-matching can also be found
in natural environments: the blowfly feeds when its receptors identify
the ring structure of a sugar, and rejects the hydrocarbon chains of
alcohols (except Inositol, which is an unnatural ring alcohol
[Hodgson, 1961]!).  Or the hungry fledgling gull that responds
immediately to the looming red spot on its parents' beak; the mating
call of the cricket (or bee), which is so precisely engineered that a
simple pattern of pulses can be tailored to reflect even subtle
species differences.  Such examples are countless (Tinbergen, 1951;
Wilson, 1971).  In each case, an important primitive goal such as
feeding or the reproduction of the species, is achieved sucessfully in
a very direct and reflexive manner only because the environment is
limited or well controlled.

Yet how can such a simple template-matching strategy serve a more
sophisticated being, who lives in a complex, changing environment?
Here, surprises may often be the rule.  When we look out a window,
walk into an unfamiliar building, or simply view a novel picture or
postcard, we have no difficulty in grasping the meaning or context of
the scene.  The greater our perceptual repertoire, the larger is the
spectrum of the unexpected and the variety of ''things'' that must be
recognized and dealt with, often out of the immediate context or frame
of mind.  Simple template matching to prestored models then becomes
impossible, for there are just too many possibilities.  Even for one
simple item---let's say a dinosaur---the possible views and
configurations is usually an infinity in itself (Fig. 1).\foot{This is
a footnote.} Without some method of initializing the perceptual
system, it must founder as a perceptron will (Minsky and Papert,
1965).  What is needed at the outset are some low level
representations or assertions that are powerful enough to capture the
```

essence of the ''event'' or ''thing'', yet are readily and routinely computable from the raw sense data. These primitive, low level assertions will constitute the answers to our 20 Questions. What inquiries then should we ask? Under what conditions can we expect such a set of questions to provide a useful set of answers?

\vskip 8mm

\cdisplay{\bf 2.0 From Templates to Questions}

\penalty 1000

\vskip 4mm

\penalty 1000

\pp In the case of lower animals which react to certain stimuli in essentially a reflexive manner, the system is preprogrammed to *recognize a simple pattern. The presence of this pattern* is almost guaranteed to represent an ''event'' or ''thing'' of importance to the animal.\foot{This is another footnote.} The pattern is thus an attribute uniquely associated with the event of interest, given the expected context. The red spot on the beak of the gull suffices for the fledgling gull because from its nest it will almost never encounter other instances of looming red spots---such as traffic lights or red balloons. In this case a simple template-matching strategy works well because of the controlled context. A simple question suffices to make reliable assertions about a complex event, namely that a parent has arrived, presumably with food.

The situation becomes considerably more complicated, however, for a general purpose perceptual system that must respond intelligently to a wide range of events in a variety of contexts. We cannot hope to find attributes or features unique for each event of interest and for each possible contextual situation. How then can we even hope to find simple questions that will have the same power as the red spot on the gull's beak? The proposed solution is to choose the questions carefully so they inquire about the more general properties of all things regardless of context.

Consider the classical children's game of 20 Questions, where the goal is to identify an object. The first questions usually attempt to identify the general class of the object. Is it animal, vegetable or mineral? Subsequent questions attempt to determine the size, shape or mass, or the sounds ''it'' might make, how ''it'' moves, or perhaps its function. The final questions then become very specific and detailed. If we are clever and shrewd in our choices, we rapidly converge to the object. Why can't a perceptual system be designed along similar lines? Imagine that for our first set of questions we identify a dozen or two---let's say twenty---very general but independent attributes of ''things''. We simply ascertain whether each attribute is present or not. Then 2^{20} or roughly a million different types of events could be crudely categorized (Webster's Dictionary only lists 60,000 words total.)

\bp

In sum, we now have four major criteria for our choice of questions:

{
\crlist
\aitem
{\it Computational Validity} -- The representation of the attribute must be easy and reliable to compute.

\aitem
{\it Conveyance} -- The attribute should encode a general property of object (such as size, mass, etc).

```
\aitem
{\it Viewer Independence} -- Representations of attributes should be
insensitive to the particular relations between the perceiver and the
''object'', i.e., to object distance, scale or disposition.

\aitem
{\it Orthogonality} -- Different attributes or questions should be
capturing independent qualities of the ''events'' or ''things''.

}

\vskip 10mm

\str

\vskip 10mm


$$\vbox{\tabskip 0pt
\def\|{\vrule height 9.25pt depth 3pt}
\def\.{\hskip-10pt plus 10000000000pt}
\hrule
\hbox to 250pt{\|\.Legal Codon Triples\.\|}
\hrule
\halign to 250pt{#\tabskip 0pt plus 100pt
^V#\hfill^V#^V#^V\ctr{#}^V#^V\ctr{#}^V#^V\ctr{#}^V#^V\ctr{#}
^V#^V\ctr{#}^V#\tabskip 0pt\cr
\|^V\hfill\.Codon\.^V\|^V\hskip -8pt\|^V0 --^V\|^V0+^V\|^V1 --^V\|
^V1+^V\|^V2^V\|\cr\noalign{\hrule}
\noalign{\hrule}
\noalign{\hrule}
\|^V0 --^V\|^V\hskip -8pt\|^V0--  2^V\|^V 1 -- ^V\|^V0--
2^V\|^V 1 -- ^V\|^V0--  2^V\|\cr\noalign{\hrule}
\|^V0+^V\|^V\hskip -8pt\|^V 1+ ^V\|^V 0+ ^V\|^V 1+ ^V\|^V 0+
^V\|^V 1+ ^V\|\cr\noalign{\hrule}
\|^V1 --^V\|^V\hskip -8pt\|^V 1+ ^V\|^V 0+ ^V\|^V 1+ ^V\|^V 0+
^V\|^V 1+ ^V\|\cr\noalign{\hrule}
\|^V1+^V\|^V\hskip -8pt\|^V0--  2^V\|^V 1 -- ^V\|^V0--
2^V\|^V 1 -- ^V\|^V0--  2^V\|\cr\noalign{\hrule}
\|^V2^V\|^V\hskip -8pt\|^V0--  2^V\|^V 1 -- ^V\|^V0--
2^V\|^V 1 -- ^V\|^V0--  2^V\|\cr\noalign{\hrule}}
\vskip 3pt
\hbox{\hfill Table 2-3}}$$

\vskip 10mm

\str

\vskip 10mm


$$\eqalign{\left\{ (0^+ \, 0^+ )\  (0^+ \, 1^+)\
(1^- \, 0^+ )\  (1^- \, 1^+)\right\}^V\ \mapsto\  0^- \cr
\left\{ (0^- \, 0^- )\right\}^V\ \mapsto\  0^+ \cr
\left\{ (0^- \, 0^- )\  (0^- \, 1^- )\right\}^V\ \mapsto\  1^- \cr
\left\{ (1^+ \, 0^- )\  (2 \, 0^- )\right\}^V\ \mapsto\  1^+ \cr
\left\{ (1^+ \, 1^- )\  (1^+ \, 2)\  (2 \, 1^- )\  (2 \, 2)\right\}^V\ \mapsto\  2\cr}$$

\vskip 10mm

\ctrline{\bf NOTE}
\pen
{\increment_both_margins{68}
Here is an example of how you can put a small paragraph in the
center of the page for emphasis. You can change the size of the
paragraph, put the type in {\bf boldface} or {\it italic} or {\bi bold
italic}...You can also {\eightpoint change the size of the type}...}

\bp
```

```
\vskip 10mm

\hbox{
\vbox to .8in{
{\hbox to .6in{\bf Box 1:}}}
{\hskip .1in}
{\vbox to .8in{\parshape 0 \hbox par 4.2in{{\it Another box} -- In this
case you can put a word or phrase out at the left and a chunk of text
boxed up to the right, and you can adjust the size of the whole thing.
You'll see, however, when you look at the input, that what you type on
the computer screen looks not at all like what you hope to get on the
page.  That is one of the main troubles with Tex---it is not
''interactive.''  Hopefully someday it will be.}}}}

\vskip 10mm

\str

\vskip 10mm

\hbox to size{\twelvepoint\bf 2.0\ This is a section heading \hfill}

\vskip 5mm

\hbox to size{\twelvepoint\it 2.1\ This is a subsection heading
\hfill}

\vskip 8mm

\fp

Here are some math examples.

\fp

{\bf Proof.}\ Let the two lines of sight from each stereo view lie in
the $XZ$ plane and intersect at $O$, as shown in Fig.\ 3.  Any point
$P(x,y,z)$ can then be specified by its distance from $O$ and two
angles $\sigma,\tau$.  Because the views are orthographic, $\tau$ appears
in the image plane, as does the elevation of $P$, namely $y^A{p}$ and
its azimuth $x^A{p}$.  The problem then reduces to recovering $\sigma$ or
$P^A{xz}$, the projection of $P(x,y,z)$ onto the $XZ$ plane.

\pp

As seen from above, the projection of $P(x,y,z)$ onto the $XZ$ plane
is shown in Fig.\ 4.  For notational convenience $P^A{1}$ has replaced
$P^A{xy}$ and $\theta={{\pi}\over{2}}-\sigma$. Our unknowns are thus
$\theta^A{i}$ and $z^A{pi}$, because $x^A{pi}$ appears in the image
plane.

From the fact that the length $OP^A{i}$ is constant over all views, we
obtain

$$\overline{OP}^{2}^A{1}=\overline{OP}^{2}^A{2}=x^{2}^A{p1}+z^{2}^A{p1}
=x^{2}^A{p2}+z^{2}^A{p2}\eqno\hbox{(1)}$$

\fp
with unknowns $z^A{p1}$, $z^A{p2}$.

\pp

From the fact that each view is stereoscopic, we obtain the
distance-disparity relation

$${{\delta x^A{p1}}\over{\delta x^A{p2}}}={{z^A{p1}}\over{z^A{p2}}}=
r^A{p}\eqno\hbox{(2)}$$
```

```
\fp
where $\delta x^A{pi}$ is the measured disparity, thereby making
$r^A{p}$ a known constant. This relation follows from the fact that
the horizontal disparity of $P$ relative to $O$ is given by

$$\delta x^A{pi}=z^A{pi}(I/D^{2})\eqno\hbox{(3)}$$

\fp
where $I$ is the interpupil distance and $D$ is the line of sight
distance to $O$, and given that the distance $OP$ is much smaller than D.
Taking the ratio of $(3)$ for $i=1,2$ eliminates the $(I/D^{2})$
dependency.

\vskip 10mm

\str

\vskip 10mm

$$F\ast I(\vec{n})=\int
F(\vec{n}-\vec{\zeta})I(\vec{\zeta})d\vec{\zeta}.$$

\vskip 10mm

\str

\vskip 10mm

$$0=\int^K{\infty}^A{-\infty}f^K{\prime\prime}\left(
{{x-zeta}\over{\sigma}}
\right)I\zeta )d\zeta .\eqno(1)$$

\vskip 10mm

\str

\vskip 10mm

$${{dx}\over{d\sigma}}={{\int^K{\infty}^A{\infty}\left(
{{x-zeta}\over{\sigma}} \right) f^K{\prime\prime\prime} \left(
{{x-\zeta}\over{\sigma}} \right) I(\zeta
)d\zeta}\over{\int^K{\infty}^A{-\infty}f^K{\prime\prime\prime} \left(
{{x\zeta}\over{\sigma}} \right)I(\zeta )d\zeta}}\eqno(2)$$

\vskip 10mm

\str

\vskip 10mm

$$\left(\vcenter{
                \halign{$\ctr{#}$\quad
                ^V$\ctr{#}$\quad
                ^V$\ctr{#}$\cr
T(x^Ao-\zeta^A1)^V\ldots^VT(x^Ao-\zeta^An)\cr
T^Ax(x^Ao-\zeta^A1)^V\ldots^VT^Ax(x^Ao-\zeta^An)\cr
T^A{xx}(x^Ao-\zeta^A1)^V\ldots^VT^A{xx}(x^Ao-\zeta^An)\cr
T\sigma (n^Ao-\zeta^A1)^V\ldots^V\sigma (n^Ao-\zeta^An)\cr}}\right)
\left(\vcenter{
              \halign{$\ctr{#}$\cr
A^A1\cr
.\cr
.\cr
A^An\cr}}\right)
=
\left(\vcenter{
              \halign{$\ctr{#}$\cr
0\cr
0\cr
```

```
-\lscr^K2\cr
1\cr}}\right)\eqno(20)$$

\vskip 10mm

\str

\vskip 10mm

\bp

{\large_size\bf
\cdisplay{Appendix I: Example Games}}

\vskip 10 mm

Here's an example of one way to align text into columns.

\vskip 12mm

\single_space

{\large_size \bf
\cdisplay{
GAME 1}}

\vskip 21pt

{\it Habitat:}\ (previously determined to be temperate environment,
green rolling hills.  Elevation of ''thing'' is on the ground.)

\vskip 20pt

\hbox to size{\hbox to 220pt{ \hfill}\hbox to 70pt{{\bf ANIMAL} \hfill}\hbox
to 70pt{{\bf PLANT} \hfill}\hbox to 70pt{{\bf MINERAL} \hfill}}

\vskip 18pt

\hbox to size{Q1:\ Is it moving? \hfill}

\hbox to size{\hbox to 220pt{ \hfill}\hbox to 70pt{{\it translates}
\hfill}\hbox to 70pt{sway \hfill}\hbox to 70pt{no \hfill}}

\hbox to size{{\it Implication}: It's an animal in motion. \hfill}
\hbox to size{\null}

\hbox to size{Q2: How many supports? \hfill}

\hbox to size{\hbox to 220pt{ \hfill}\hbox to 70pt{{\it 2,4 or >4}
\hfill}\hbox to 70pt{1 \hfill}\hbox to 70pt{0 \hfill}}

\hbox to size{{\it Implication}: Confirms animal --- \hfill}
\hbox to size{has four ''legs''. \hfill}
\hbox to size{\null}

\hbox to size{Q3: What acoustic frequencies are emitted? \hfill}

\hbox to size{\hbox to 220pt{ \hfill}\hbox to 70pt{narrowband \hfill}\hbox
 to 70pt{{\it broad} \hfill}\hbox to 70pt{{\it broad} \hfill}}

\hbox to size{{\it Implication:} Disconfirms animal. \hfill}
\hbox to size{''Thing'' makes low frequency, broad--band \hfill}
\hbox to size{sounds, moves and has 4 legs.  Must be big. \hfill}
\hbox to size{Elephant or cow? \hfill}\par
\hbox to size{\null}
\hbox to size{Q4: Acoustic Source \hfill}

\hbox to size{\hbox to 220pt{ \hfill}\hbox to 70pt{{\it point} \hfill}\hbox
to 70pt{extended \hfill}\hbox to 70pt{extended \hfill}}
```

```
\hbox to size{{\it Implication:}\ Confirms ''animal'' or \hfill}
\hbox to size{isolated object. \hfill}

\bp

\vskip .5in

An example of a numbered list:

\vskip 10mm

{\large_size\bf
\cdisplay{Appendix II}}

\vskip 11 mm

Documentation of devices that can provide answers to each
of the twenty questions.

\vskip 4mm

{
\ialist
\aitem
{\bf Acoustic Frequency.}\ Comb filtering has been used for several
years to separate sound sources (Shields, 1970; Flanagan,
1972; Zwicker et al., 1979). Unless many broad--band sources are active
simultaneously at S.P.L.'s comparable to the narrowband sources, this
question can be answered with available technology (Klatt, 1977). As initially
formulated (Richards, 1980), the question simply addresses whether
the source is broad--band or not (such as wind through the trees,
rushing water, or an animal cry). Much more useful but also much more
difficult, would be to extract the physical properties of the source
--- i.e., its acoustic ''color'': Is it metallic, wood striking
 wood, or a footfall?

\aitem
{\bf Acoustic Modulation.}\ Tracking a sound source to determine its
modulation characteristics (Atal, 1972) also requires localization
 (as may Question $\#$1). For narrow--band, harmonic sources with
different spectral signatures, such localization is possible provided
there are only a few competing sources (Altes, 1978). Again, as in
Question $\#$1 work should be undertaken to understand how the
''textural'' properties of the source can be extracted from the
modulations. For example, is the source ''harsh'' or grating, or like
clacking sticks, or ''suave '' and ''smooth'', or ''roaring'' like a
brook or lion.

\aitem
{\bf Frequency Change.}\ Here again, as in Questions $\#$1 and $\#$2
 localization is helpful but not as necessary because only ANIMALS are
generally capable of producing sounds of variable frequency. Simple
1/3 octave filtering should allow the detection of frequency change
(Flanagan, 1972; Klatt, 1977.)

\aitem
{\bf Motion.}\ The motion of an ''object'' can be both visual and
auditory. Clearly the detection of auditory movement requires
localization (Altes, 1978; Searle et al., 1980), and may be difficult.
Visual motion detection has progressed enormously over the past ten
years, and can be detected with simple systems provided the background
is stationary (Horn and Schunck, 1981; Thompson, 1981; Ullman,
1981; Hildreth, 1982). More work is still required, however, to use
motion to segregate a visual scene, especially if sway or
scintillation is to be disambiguated from translation or rotation.

\aitem
{\bf Support.}\ Although a powerful question, to estimate the numbers
```

of ''legs'' supporting a region is quite complicated. First, the ground plane must be determined (see Question $\#$20), secondly the candidate ''support'' must be recognized (e.g., leg or trunk) and finally a region should be identified as being supported although it may have a different color or texture. In the case of stationary supports, the local parallelism of the vertical occluding edges of the support may serve as a basis for determining the supporting member (Stevens, 1980). What to do in the case of animal motion, however? Also, shrubs clearly may have many ''supports''. The computational validity of this attribute is questionable, therefore, although a strong assertion would be quite useful.

\vskip 12mm

\str

\bp

\vskip 6mm

One way in which references can be formatted:

\vskip 10mm

\ctrline{\large_size\bf REFERENCES}}

\vskip 10mm

\ivp

Agin, G., Representational description of curved objects. {\it Stanford AI Project Memo AIM\~ 173, Stanford University, 1972.}

Altes, R.A.\ Angle estimation and binaural processing in animal echolocation. {\it J.\ Acoust.\ Soc.\ Amer.,} 1978, {\it 63,} 155\~ 183.

Atal, B.S. Automatic speaker recognition based on pitch contours. {\it J.\ Acoust.\ Soc.\ Amer.,} 1972, {\it 52,}1687\~ 1697.

Bajcsy, R. and Badler, N., {\it Representation of Three-Dimensional Objects.} New York: Springer-Verlag, 1982.

Ballard, D.H. and Brown, C.M. {\it Computer Vision.} New Jersey: Prentice-Hall, 1982.

Barbe, D.F., Smart sensors. {\it Proc.\ Soc.\ Photo-Opt.\ Instrum. Eng.,} 1979, {\it 178.}

Benedict, R.P., {\it Fundamentals of Temperature, Pressure, and Flow Measurements,} New York: Wiley, 1969.

Binford, T., Inferring surfaces from images. {\it Art.\ Intell.,} 1981, {\it 17,} 205\~ 244.

Boden, M., {\it Artificial Intelligence and Natural Man.} New York: Basic Books, 1975??

Chance, J.E. and LeMaster, E.W., Suits reflectance models for wheat and cotton: *theoretical and experimental tests.* {Applied Optics,} 1977, {\it 16,} 407\~ 412.

Cook, R.L. and Torrance, K.E., A reflectance model for computer graphics. {\it ACM Trans.\ on Graphics,} 1982, {\it 1,} 7\~ 24.

Cox, C.P. and Baron, M., A variability study of firmness in cheese using the ball-compressor test. {\it Journal of Diary Research,} 1955, {\it 22,} 386\~ 390.

Davis, L. and Rosenfeld, A., Computing processes for low level usage:

a survey. {\it Art.\ Intell.,} 1981, {\it 17,} 245-263.

Dirlam, D.K., Most efficient chunk sizes. {\it Cogn.\ Psych.,} 1972, {\it 3,} 355\~ 359.

Flanagan, J.L., {\it Speech Analysis: Synthesis and Perception.} Berlin: Springer\~ Verlag, 1972.

Francis, F.J. and Clydedale, F.M., {\it Food Colorimetry: Theory and Applications.} Westport, CT: AVI Publishing, 1975.

Grimson, W.E.L., {\it From Images to Surfaces.} Cambridge, MA: MIT Press, 1981.

Harmon, L.D., Automated tactile sensing. {\it Int.\ J.\ Robotics Res.,} 1982, {\it 1 (2),} 3\~ 32.

Harmon, L., The recognition of faces. {\it Sci.\ Amer.,} 1973, {\it 229,} 70\~ 82.

Herzfeld, C.M., {\it Temperature, Its Measurement and Control in Science and Industry, vol. 3}. New York: Reinhold, NY, 1962.

Hildreth, E., The integration of motion information along contours. {\it IEEE Proceedings of a Conference of Computer Vison Representation and Control, Sept., 83\~ 91, 1982.}

Hillis, W.D., A high resolution image touch sensor, {\it Int.\ J. Robotics Res.,} 1982, {\it 1 (2),} 33\~ 44.

\end

Section Two:
MAIL

13.0 Sending and Receiving Mail*

There are several ways to send and receive mail via the OZ system. Here we'll discuss two: MM and BABYL. MM is the simplest to use, and it works well enough. But BABYL is much more sophisticated and contains many options and possibilities. Eventually you'll probably want to ignore MM and use BABYL for everything: sending mail, receiving mail, editing your file of back mail, and receiving system and bulletin board type messages. But for now it would be a good idea to send and receive a letter via MM, just so you know what it's like.

13.0 Using MM for Sending Mail

13.1 Send Yourself a Letter

A good way to begin is to send yourself a letter. Let's say your name is Maxine, and you are going to send a message to Louise. Normally, of course, you would know Louise's login name and you would address the letter to her login name. But for now a bit of pretending is in order. First, you must be at top level—that is, you must have the @ sign at the left side of the screen.

Type **MM <CR>**

Something like this will appear:

```
MIT-OZ.#Chaos MM-20 5.3(1039)
You have no MAIL.TXT.1
MM>
```

The cursor will be to the right of **MM>**, waiting for you to type something.

Type **SEND <CR>**

```
To:
```

will appear, with the cursor to the right of the colon. Here is where you would normally type Louise's login name. Instead, type **NCC <CR>**.

Now, **cc:** will appear. This is for "carbon copy." You can have copies of the message sent to yourself or anyone else on the system. It's always a good idea to send yourself a copy. Since this letter is going to yourself, you won't need a copy; but type **NCC** anyway, because that's what you'd normally do.

*An easy way to send a message is simply to type **MAIL <cr>**, followed by the address and the message, then by **CTRL Z** (↑ Z). This method has limitations, however, and readers are encouraged to familiarize themselves with MM and/or BABYL.

Next the screen will say

```
subject:
```

You might type `Bird's Nest <cr>`

Then, on the screen will appear:

```
Message (End with ESCAPE or CTRL/Z.
 Use CTRL/B to insert a file, CTRL/E to enter editor, CTRL/K to redisplay
 message, CTRL/L to clear screen and redisplay, CTRL/N to abort.):
```

For now, don't worry about these various commands; just type your message.

```
Hello Louise --

I just noticed there is a strange bird building
a nest outside my office window. I thought you
might like to come over and have a look.

Maxine
```

When you are finished with your message, type `CTRL Z` (↑ Z).

Now `S>` will appear at the left of your screen.

Type `SEND <cr>`

The screen will say

```
ncg@MIT-OZ.#Chaos -- queued
MM>
```

This means the message is being sent.

If that is the only message you wish to send, type `QUIT <CR>` and that will put you back at top level, with the @ at the left of the screen.

In about a minute (longer when the system is slow) a message will appear on your screen:

```
[You have mail from NCG at 13:50]
```

You can read the message at this point by going back into MM and typing `READ <CR>`, but for now it would be better if you kill the MM program, so that when you start it again you'll see what it is like to have a new message waiting for you. To review what "killing" programs means, turn back to page 18.)

Type `RESET MM <CR>`

You'll still be at top level.

14.2 Using MM for Receiving Mail

To read your mail, type MM <CR>

Something like this will appear:

```
MM Version 5(1457), Edit 815
Last read: 16-May-83 14:29:30, 1 message, 1 page
1 message unseen
MM>
```

To get your message:

Type READ NEW <CR>

```
Message 1 (201 characters):
Return-path:<NCG@MIT-OZ>
Mail-From: NCG created at 16-May-83 14:21:33
Date:16 May 1983 1421-EDT
From: Natural Computation <NCG@MIT-OZ>
Subject: Bird's Nest
To: <NCG@MIT-OZ>

Hello Louise --

I just noticed there is a strange bird building
a nest outside my office window. I thought you
might like to come over and have a look.

Maxine

R>
```

R> means "reply," and at this point you can type REPLY <CR> if you want immediately to answer the letter. But for now type Q <CR> which will give you

```
MM>
```

Like many of the programs on OZ, MM has a certain amount of helpful information about itself available to a user. Type ? <CR>. A list of words in capital letters will appear on the screen. These are commands available to you at this level of MM. If you are curious about what any one of them does, type HELP followed by the command name and follow it with a carriage return (<CR>). You may not always understand the explanation, but it's a good way to begin to explore the possibilities of the program. Note that there is a general command HELP available at this level of MM; you might want to try it: type HELP HELP <CR>

Incidentally, it's good to remember when you're stuck and can't figure out what to do—at top level or while in other programs—that often this type of help is available. Type a ? or HELP <CR>; sometimes it won't get you anywhere, but often it will.

Now, type QUIT <CR> to get out of MM and back to top level. Again, just for good measure kill the MM job.

14.3 MM Commands Used So Far

MM <CR> – starts up the MM program

? – gives you a list of possible commands

HELP HELP <CR> – gives you some basic information about the program

Q <CR> – when you have the R> symbol at the left of the screen, you can type Q <CR> to get you back to MM>

SEND <CR> – type this when you want to send a letter

CTRL Z (↑Z) – once you have written the letter, this actually sends it

QUIT – to get out of the MM program (does not kill it)

RESET MM <CR> – kills the MM program

15.0 Using BABYL: the Basic Idea

BABYL, like MM, is a program for sending and receiving mail. In the way it works it is a lot like EMACS. Also, like EMACS, it is fairly easy to learn to use at a beginner's level, and it has sufficiently complicated possibilities to keep you interested as you get more confident.

[NOTE – If you have your own directory and wish to begin using BABYL, merely type BABYL <CR>. A series of questions will appear on the screen, together with suggested answers. Type in the answers as suggested. But for now you should use BABYL at least once with the NCG directory.]

15.1 BABYL: Sending a Letter

At top level, type BABYL <CR>

Near the bottom of the screen it will say:

```
Reading Babyl file PS:<NCG>NCG.BABYL.1
Reading Mail file PS:<NCG>MAL.TXT.0
Appending to Babyl file PS:<NCG>NCG.BABYL.1
```

Then, above this, a line such as

```
Babyl (Message 5/7, unseen) PS:<NCG>NCG.BABYL.1
```

will appear, and above that the text of one item of mail. It will be an item of mail stored in the NCG.BABYL file. If this were your directory, it would either be a letter you have not yet read, in which case the top line at the bottom of the screen would contain the word **unseen**, or it would be an item of old mail which you hadn't deleted.

If the text of the letter which has appeared fills more than one screen, near the bottom it will say --MORE--, in which case hit the space bar to see the rest.

Rather than discuss here the various commands for reading mail, it might be a good idea first to send yourself a piece of mail and then read it, so you get the feel of actually manipulating BABYL.

NOTE

> As in EMACS, there is a way to get out of the program in the event that things go wrong and you become hopelessly confused. Type CTRL C (↑C). Try it twice if once doesn't work. This should get you out of BABYL. Once you're out, type RESET BABYL <CR> to kill the BABYL program entirely. Then start over.

Now, to send yourself a piece of mail: assuming you are still in BABYL, type M. Most commands typed while in BABYL don't need to be followed by <CR>.

This will appear at the top of the screen:

```
To:
   --Text follows this line--
```

The cursor will be next to the colon. Type NCG. Then type <CR>, which will move the cursor down, creating a blank line. As in MM, you want to put the basic address information at the top of the letter. In BABYL you must type the headings yourself.

NOTE: BABYL is very similar to EMACS, and most EMACS text-manipulating commands work in BABYL. Remember that:

DEL (Rubout) – *Deletes* the character preceding the cursor
CTRL-D (↑D) – *Deletes* the character above the cursor
CTRL-F (↑F) – moves the cursor F*orward* one space
CTRL-B (↑B) – moves the cursor B*ackward* one space
CTRL-N (↑N) – moves the cursor to the N*ext* line
CTRL-P (↑P) – moves the cursor to the *Previous* line

Okay, now type the heading for your letter:

```
To: ncg
```

Type a <CR> at the end of the line to create a blank space.

Then: `cc: ncg`

Since you're sending the letter to yourself, you don't really need a copy, but do it anyway for practice.

Then type: `subject: a strange event`

You don't need to put `from:` or a date; BABYL will take care of that.

Now you're ready to write the text of your letter. Using the EMACS-like command CTRL N (↑N) move the cursor below the line which says `--Text follows this line--` and write your letter.

```
Lorne:

Today when I woke up I discovered I was in a room different
from the one in which I went to sleep.  This confused me, and
I lay there awhile in bed trying to understand what had happened.

                                regards,

                                Melinda Bellinda
```

Having finished your letter, type CTRL Z CTRL Z (↑Z ↑Z). There will be a delay, then the screen will say (near the bottom):

```
Queuing...Done.
```

That means your letter has been sent.

To get out of the BABYL program merely type Q. After a short pause, the @ will appear at the left margin. Incidentally, a good command, which is useful both at top level and in EMACS, is CTRL L (↑L)—this "clears the screen," removing anything left from previous operations, such as a program you've just gotten out of, and it gives you a new clean working surface.

It may take a few minutes for the letter to arrive in your directory, even though you've sent it to yourself. If you stay logged in during the time you are waiting for its arrival, your terminal will beep and the message `[You have mail from NCG]` will appear on the screen.

Now, to read your letter.

15.2 BABYL: Reading Your Mail

The best thing to do now is log out. In this way you can see what it is like when you log in and have a letter waiting for you which you haven't yet read. If you don't remember how to log out, turn to page 18.

Now, log in. (Instructions for logging in begin on page 2.)

When the normal screen messages have appeared and the @ sign shows at left of the screen, type BABYL <CR>. After a delay, your letter will appear:

```
Date: Monday, 20 June 1983 14:23-EDT
From: NCG
To:   ncg
cc:   ncg

Lorne:

Today when I woke up I discovered I was in a room different
from the one in which I went to sleep.  This confused me, and
I lay there awhile in bed trying to understand what had happened.

                          Regards,
                          Melinda Bellinda

Babyl (Message 10/10, recent, unseen, last) PS:<NCG>NCG.BABYL.1
Reading Babyl File PS:<NCG>NCG.BABYL.1
Reading Mail file PS:<NCG>MAIL.TXT.0
Appending to Babyl file PS:<NCG>NCG.BABYL.1
```

This message appears on your screen automatically, when you call up the BABYL program, because it is the last one received and you haven't read it yet. All the mail received by BABYL gets stored. Therefore you have to know how to read past messages. For this you need some basic BABYL mail-reading commands:

N – goes to the next message
P – goes to the previous message
G – gets any mail that's come in since you started the BABYL program
D – deletes a message
U – undeletes a message

For a list of all possible BABYL commands, type ?

At the bottom of the screen it will say:

```
Type a Babyl command character to describe, "*" for all of them:
```

So, type *

There will appear a long list of commands, together with short explanations of what each one does. Most of these will probably seem obscure to you; but you only need a few to get going.

15.3 Basic BABYL Commands

BABYL <CR> – starts up the BABYL program (or puts you back into BABYL).

Q – gets you out of BABYL

M – type this if you want to send a letter.

N – goes to the next message

P – goes to the previous message

G – gets any mail that's come in since you started the BABYL program

D – deletes a message

U – undeletes a message

? then * – gives you a list of BABYL commands

CTRL Z CTRL Z (↑Z ↑Z) – to send a letter; type this *after* you have typed M and have typed the text of your letter.

CTRL L (↑L) – clears the screen (not just a BABYL command; it works at top level and in EMACS)

CTRL C CTRL C (↑C ↑C) – gets you out of BABYL in the event that things go wrong

Section Three:
ODDS AND ENDS

16.0 INQUIRE, WHOIS, and FINGER

When you first log in to your own directory you should run the INQUIRE program and answer the questions it asks you. To do this, type `INQUIRE <CR>` at top level. This program is fairly easy to use. It would be a good idea to type `? <CR>` once the program has started, to get an explanation of what is expected of you. Don't be confused when the program refers to itself as WATSON—this program has two names.

The reason for running INQUIRE is to provide OZ with some basic information about yourself. As INQUIRE will explain, you don't have to answer all the questions; but you should provide a minimum, such as name and MIT address.

The last question on the INQUIRE program shouldn't be overlooked. It provides a space where you can type whatever you want—a quotation, a poem, a mathematical equation, some eccentric thought you feel you must get off your chest. Interesting INQUIRE entries can help lend some individuality to the often faceless fellow-users of OZ.

To see the results of what you have typed into INQUIRE, at top level type `WHOIS [YOUR LOGIN NAME] <CR>`. This is what other people will see if they are trying to figure out how to get in touch with you. `WHOIS` also works with a person's last name.

If you type `WHOIS <CR>` you will get a listing of all the people logged in to OZ, their INQUIRE entries, and what programs they are running.

A simpler way to see who is logged in to OZ is to type `FINGER <CR>` (like most OZ commands, this can be abbreviated: type `F <CR>`). A list will appear on the screen giving everyone's login name, their real name, the program they are running, and the location of their terminal. You can also type `FINGER`, followed by a login name and `<CR>`, to get the same information for just one person.

16.1 INQUIRE, WHOIS, and FINGER Commands

`INQUIRE <CR>` – starts the INQUIRE program to fill out your WHOIS entry

`? <CR>` – An INQUIRE (or WATSON) command to get an explanation of the program

`WHOIS [YOUR LOGIN NAME] <CR>` – to see your INQUIRE (or WHOIS) entry

`WHOIS <CR>` – to see the INQUIRE information on everyone who is logged in, plus what programs they are running and where they are.

`FINGER <CR>` – to see a shorter list of everyone who is logged in

17.0 The Files in the NCG Directory

If you are logged in as NCG and you type DI <CR> there will appear on the screen a list of the files in the NCG directory. In beginning to learn how to use OZ, it would be helpful to familiarize yourself with the contents of some of these files. A good place to start is with the LOGIN.CMD file.

When you are at top level, with the atsign (@) at the left of the screen, you are ready to communicate with, or "address" the *EXEC*. The *EXEC* is a part of the *OPERATING SYSTEM*, which is the main program, or group of programs, which runs the computer. When you log in to NCG one of the first things to happen is that the operating system reads your LOGIN.CMD file and *EXECUTES* any commands it finds there. These are commands you could have typed in yourself, but it is easier and faster to have the operating system do them for you. The NCG LOGIN.CMD file, for example, tells the operating system to set up your terminal so that it is "terminal verbose," which causes the --Pause-- or --More-- to appear at the bottom of your screen, telling you to hit the space bar to see the next page. This is because the operating system executed the command terminal verbose which it found in your LOGIN.CMD file.

The NCG LOGIN.CMD file has had comments added to it in order to make it more understandable for beginners. To see the file, type TYPE LOGIN.CMD <CR>. The words appearing to the right of ! are comments; the operating system will not try and execute anything on a line following a !.

You might want to copy the NCG LOGIN.CMD file into your own directory. As you learn more about how to use OZ you can customize it to suit your own needs. To copy it, type COPY <NCG>LOGIN.CMD <YOUR LOGIN NAME>LOGIN.CMD <CR>.

There are other files in the NCG directory which perform functions similar to the LOGIN.CMD file. For example, when you start up an EMACS program (see page 5) and the line EMACS (TeX Fill) WBG.TXT (1) --BOT-- appears at the bottom of the screen, the word Fill inside the parenthesis means that your EMACS job is in "fill mode"—i.e., when typing in text you will not have to worry about inserting carriage returns every time your words approach the right edge of the screen; they will be inserted automatically. This "mode" is the result of a command in your EMACS.VARS file. Again, you should look at this file; it too has comments. You might also want to copy it into your own directory.

The files LOGOUT.CMD, BABYL.VARS, MM.INIT are also files of similar type. You should look at them. Each can be copied and later customized.

18.0 Help on the System

In learning to use OZ you should bear in mind that there is a lot of helpful information "on-line"—that is, within the computer itself. The quality of this documentation is uneven; some of it is excellent, but in other places beginners will find themselves baffled by impenetrable jargon.

In general it is a good idea, when confronted with a problem, to try first to find the answer within the system. If, for example, you are having trouble getting a mailer program such as MM to do what you want, try exploring around, looking for the "documentation" which will tell you what you want to know. This will not only increase your confidence, but it often leads to interesting discoveries.

Here are a few places to begin to look for on-line help:

1. At top level type `HELP <CR>`. On the screen you'll find listed three possible starting places: (1) `HELP ? <CR>` (2) `HELP <NAME> <CR>` and (3) `HELP DOC-HELP <CR>`. Try them all. The third, `HELP DOC-HELP <CR>`, is a good place to begin. The first, (`HELP ? <CR>`), will give you a listing of topics, any one of which can be combined with `HELP <NAME> <CR>`. For example, try typing `HELP DOCUMENTATION <CR>`.

2. At top level type `TEACH-EMACS <CR>`. This will start the TEACH-EMACS program, an excellent tutorial which will lead you step-by-step through all the basics of EMACS.

3. While in EMACS, type `CTRL X I` (↑X I). This will put you into a "tree-structured" documentation program which is full of information about the entire OZ system.

4. While in EMACS type either `CTRL ←` (↑←) or `CTRL _` (↑_), whichever is available on your terminal. At the bottom of your screen it will say `Doc (? for help):`

   Type `?`

   This will give you a list of letters you can type for various kinds of help within EMACS. These possibilities (for example, `C` explains what individual EMACS commands do) will become useful after you are familiar with EMACS. It would probably be better to go through TEACH-EMACS first.

5. Remember that most programs, such as MM and BABYL, have some built-in documentation. Often typing `HELP <CR>` will start you on the right trail.

6. The OZ directory which holds most of the documentation files is `PS:<DOCUMENTATION>`. To see the listing of these files, type `DI <DOCUMENTATION>`. If you're curious about any particular file, merely type `TYPE` (or `TY`) followed by the name of the directory, the file, and a `<CR>`. For example, `TYPE <DOCUMENTATION>JARGON.TXT <CR>`.

19.0 List of Keys and Commands

KEYS:

CTRL (↑) – (lower left) hold it down while you press other keys

ESC ($) – (upper left) press it and release it

<CR> – (large key at right) carriage RETURN

TOP LEVEL (@) COMMANDS:

CTRL C (↑C) – try this if trouble develops and you want to get back to top level. Try it twice if once doesn't work

COPY [OLD FILE NAME] [NEW FILE NAME] – makes a copy of a file and allows change of name

LOGOUT <CR> or LOGOUT <CR> – to log out

EMACS <CR> – calls up the EMACS program

EMACS COMMANDS:

General Commands

CTRL X CTRL S (↑X ↑S) – *SAVES* what you've done so far

CTRL Z CTRL Z (↑Z ↑Z) – gets you out of EMACS and back to top level (@);

Moving the Cursor

CTRL A (↑A) – moves to the beginning of the line

CTRL E (↑E) – moves to the end of the line

CTRL F (↑F) – moves forward over one character

CTRL B (↑B) – moves backward over one character

CTRL N (↑N) – moves down one line, vertically.

CTRL P (↑P) – moves up one line, vertically.

CTRL L (↑L) – clears the screen and reprints everything.

CTRL T (↑T) – transposes two characters.

ESC < ($<) – moves to the top of text.

ESC > ($>) – moves to the bottom of text.

Sentence and Paragraph Commands

ESC A ($A) – moves back to the beginning of the sentence. *(Two spaces must separate the sentences.)*

ESC E ($e) – moves forward to the end of the sentence.

ESC [($[) – moves back to the beginning of the paragraph.

ESC] ($]) – moves forward to the end of the paragraph.

CTRL O (↑O) – inserts one or more blank lines after the cursor.

Deletion and Killing

CTRL D (↑D) – deletes the next character.

DEL *(Also called RUBOUT)* – deletes previous character.

CTRL K (↑K) – kills rest of line or one or more lines.

ESC D ($D) – kills word.

ESC K ($K) – kills to end of sentence (the sentence must end with two spaces for this to work).

Un-Killing

CTRL Y (↑Y) – yanks (re-inserts) the last killed text.

TeX AND *DOVER* COMMANDS

TEX [NAME OF FILE] <CR> – tells TeX to check your file and make a *press file*

DOVER [NAME OF FILE] <CR> – sends your *press file* to the Dover

DVRQ <CR> – shows you the Dover queue and tells you if the Dover is working

COMMANDS USED FOR DIRECTORY CLEANUP

DI <CR> – shows you your directory

DELETE [NAME OF FILE] <CR> – DELETES the file specified

UNDELETE [NAME OF FILE] <CR> – UNDELETES, i.e, gets back a deleted file

ESC X ($X), then type REAP <CR>– (works *only in EMACS*) Deletes all but two copies of file. You must also go to top level and EXPUNGE

EXPUNGE <CR> – EXPUNGES all deleted files

JOB KILLING AND LOGOUT COMMANDS

INF FO <CR> – shows the jobs you have running

RESET [NAME OF JOB], or * – kills one or all jobs

KK <CR> – logs you out

MM COMMANDS

MM <CR> – starts up the MM program

? – gives you a list of possible commands

HELP HELP <CR> – gives you some basic information about the program

Q <CR> – when you have the R> symbol at the left of the screen, you can type Q <CR> to get you back to MM>

<CR> – when you have the R> symbol at the left of the screen, you can type a <CR> to get you back to MM>

send <CR> – type this when you want to send a letter

CTRL Z (↑Z) – once you have written the letter, this actually sends it

QUIT – to get out of the MM program (does not kill it)

RESET MM <CR> – kills the MM program

BABYL COMMANDS

BABYL <CR> – Starts up the BABYL program (or puts you back into BABYL).

Q – gets you out of BABYL

M – type this if you want to send a letter.

N – goes to the next message

P – goes to the previous message

G – gets any mail that's come in since you started the BABYL program

D – deletes a message

U – undeletes a message

? then * – gives you a list of BABYL commands

CTRL Z CTRL Z (↑Z ↑Z) – to send a letter (after typing M to begin the letter)

CTRL L (↑L) – clears the screen (not just a BABYL command; it works at top level and in EMACS)

CTRL C CTRL C (↑C ↑C) – gets you out of BABYL in the event that things go wrong

INQUIRE, WHOIS, AND *FINGER* COMMANDS

INQUIRE <CR> – starts the INQUIRE program to fill out your WHOIS entry

? <CR> – an INQUIRE (or WATSON) command to get an explanation of the program

WHOIS [YOUR LOGIN NAME] <CR> – to see your INQUIRE (or WHOIS) entry

WHOIS <CR> – to see the INQUIRE information on everyone who is logged in, plus what programs they are running and where they are.

FINGER <CR> – to see a shorter list of everyone who is logged in

HELP COMMANDS

HELP <CR> – general top level HELP command

HELP ? <CR> – gives you a list of HELP topics

HELP <NAME> <CR> – to get information on a specific topic (HELP ? <CR>) lists the topics

HELP DOC-HELP <CR> – tells you where to look for documentation

TEACH-EMACS <CR> – starts the TEACH-EMACS tutorial

CTRL X I (↑X I) – an EMACS command, gets you a tree-structured information program

CTRL ← (↑←) or CTRL _ (↑_), whichever is available on your terminal. At the bottom of your screen it will say Doc (? for help): Type ? – Shows you various kinds of help available in EMACS

20.0 List of Manuals

TOPS-20 User's Guide. *(AA-4179C-TM)* Digital Equipment Corp., Marlboro, MA, 1980. Available at the LCS publications room, NE43-016, as well as by mail from DEC.

A company manual from DEC, one of many which apply to OZ. This is a beginner-level manual and contains a lot of handy information.

EMACS Manual for TWENEX Users. Richard M. Stallman. MIT A.I. Memo No. 555, 1981. Available on the 8th floor of NE43, Room 818.

A good manual, well organized and for the most part clearly written. To a beginner, especially someone with no computer experience, it can seem at first overly difficult. But with just a little experience working with EMACS, it will begin to get clearer. This manual is essential and you should get a copy.

TeX and Metafont: New Directions in Typesetting. Donald E. Knuth. Digital Press, American Mathematical Society, 1979. Available through the Coop.

The second section of this book, "TeX, a system for technical text," is the manual for the version of TeX (TeX80) described in the Idiot's Guide to OZ. *Like the EMACS manual, this book is essential and you should get a copy. (See remarks below, under* The TeXbook.)

Tbase files — *While not really a "manual," there is a group of (TeX80) files you should become familiar with. The following files can be outputted by typing* `DOVER [NAME OF FILE] <CR>`; *for example,* `DOVER <TEX80.DOC>SAMPLE.TEX <CR>`. *At first these files may look puzzling to you, but keep them around. They are about the only documentation there is for "Tbase," an important series of improvements made to TeX at MIT (see below). The more you use TeX the more valuable they will become to you.*

```
PS:<TEX80.DOC>SAMPLE.TEX
              MACRO.DOC
              TBASE.INFO
              TBASE.ORDER
              ERRATA.TXT

PS:<TEX80.MACROS>HBASIC.TEX
                 LETTER.TEXLIB
                 MATH.TEXLIB
                 PAPER.TEXLIB
                 TBASE.TEXLIB
```

As stated in the Preface, there is more than one TeX program at MIT. The *IDIOT'S GUIDE TO OZ* describes how to begin to use TeX80, and some of the the additions (known as Tbase) made by Daniel Brotsky in 1982. TeX80-Tbase still works, and it is an excellent program. There are, however, two new programs of which the beginner should be aware: TeX82 and LaTeX. As improvements are made to these programs, they will undoubtedly replace TeX80-Tbase, probably in a year or two. Anyone interested in learning these new programs should consult the following manuals:

The TeXbook. Donald E. Knuth, Addison-Wesley, Reading, Mass., 1984. Available through the Coop.

This 483-page manual can be intimidating. It is similar to the second section of TeX and Metafont, *listed above, and adheres to that book's frustratingly dense and often confusing tutorial method. The idea is for the reader to proceed through a series of exercises, but the amount of work expected of a beginner can seem staggering, and few people bear with it. The alternative is to use the book's index, which for any single topic might lead you down a trail of five or six partial explanations scattered through many pages of text.*

First Grade TeX: A Beginner's Manual. Arthur L. Samuel. Report No. STAN-CS-83-985 (Version 1), Stanford Department of Computer Science, 1983.

A brief, stripped-down TeX82 manual based on The TeXbook. *There is a copy in the Laboratory for Computer Science library in building NE43.*

The LaTeX Document Preparation System. Leslie Lamport, 1983. This is a preliminary version of a future manual. Available at NE43-818.

LaTeX is a program based on TeX82, and has some very nice features, such as indexing and a method for making tables of contents. The program is designed to be used by "lay" people, i.e., those not expert in the difficult complexities of TeX82. At MIT LaTeX presently has the serious disadvantage of being available with only one family of type: computer modern.

Anyone wishing to begin to use TeX82 or LaTeX should get two documents available at NE43-818: "Using TeX82 on MIT-OZ," and "Using LaTex on MIT-OZ."

21.0 Glossary

[NOTE: Many of the following terms have highly specific, detailed meanings when applied to computers in general. No attempt has been made here to provide that kind of definition. Rather what follows are a few generalizations which OZ beginners will hopefully find useful. Persons interested in exploring the arcane world of computer terminology might begin with *The Penguin Dictionary of Computers* by Anthony Chandor, Penguin Books, 1977.]

ACCOUNT–A person's means of access to OZ. An ACCOUNT provides you with a LOGIN NAME, PASSWORD, and a certain amount of memory space. Some kinds of ACCOUNTS allow you greater privileges than others.

CHAOSNET–A system allowing communication between MIT computers.

COMMAND–An order which you give to one of OZ's programs. It says, "Do the following..." There is also a *TEX COMMAND*; see page 11.

COPY–To duplicate a FILE. The original *VERSION* of the file remains intact, and the new *VERSION* can be given a different name.

CURSOR–The little blinking line or rectangle on the screen which you move around in order to type in, remove, or manipulate characters.

DELETE–The first step in erasing a file. The next, and final, step is to EXPUNGE it.

DIRECTORY–A listing of FILES.

DOCUMENTATION–Explanations of how *PROGRAMS* work, or how to use them. See ON-LINE.

DOVER–The machine in NE43 from which you get OUTPUT. Also called a "printer," or "laser printer."

DOWN–Adjective meaning "not working." Opposite of UP.

EXEC–That part of the OPERATING SYSTEM which you address when you type at TOP LEVEL. Often used loosely to mean TOP LEVEL.

EXPUNGE–See DELETE.

FILE–A unit of stored information. A DIRECTORY is a list of FILES.

FORK–OZ terminology for *JOB*. If you have an EMACS JOB or PROGRAM running, you are also said to have an EMACS FORK.

INPUT–What you type into the computer. Used both as verb and noun.

JOB–See PROGRAM and FORK.

KILL (A JOB)–To signal OZ that you no longer need a particular program (or JOB or FORK). The TOP LEVEL command for this is RESET.

LOGIN NAME–The name, or string of characters, which you type to identify yourself to OZ. In logging in, you must combine it with a PASSWORD.

ON-LINE–Loosely, an adjective meaning "in the computer." DOCUMENTATION is said to be ON-LINE if you can make it appear on your terminal screen. It is not ON-LINE if it is only to be found printed on paper.

OPERATING SYSTEM–The main program, or group of programs, which runs the computer, and which enables many people to use the computer at once.

OUTPUT–What comes out of the Dover, i.e., the "hard copy," the paper with print on it. Also frequently used as a verb, as in "Did you output the file?"

PASSWORD–A name, or string of characters, which you type when logging in, but which does not appear (or "echo") on the screen.

PRESS FILE–A text file which has been processed by TeX and is ready to be sent to the Dover to produce OUTPUT.

PROGRAM–In its most general usage, a series of instructions to be performed by a computer. As used in the IDIOT'S GUIDE it refers to certain powerful complex tools, such as EMACS and TeX, which can be called upon to perform specific tasks.

PROMPT–The symbol which appears at the left edge of your screen when the computer is waiting for you to type something. Different programs may have different PROMPTS. At TOP LEVEL on OZ it is the "atsign" (@).

QUEUE–A line in which *FILES* are waiting and advancing one at a time to be processed.

TBASE–A group of files added to TeX80 at MIT which greatly improved that program. See the Preface and Section 20.0, "List of Manuals."

TOP LEVEL–When you are logged in, and you have not yet called up any programs, you are at TOP LEVEL. The "atsign" (@) will be at the left of the screen. See EXEC.

UP–See DOWN.

VERSION–The number assigned by the computer to a *FILE* indicating how many times that file has been stored. It does not mean that many VERSIONS currently exist, since the earlier ones will (hopefully) have been DELETED.

WAKE UP–To cause an idle terminal to be ready to communicate directly with OZ is to "wake it up."

22.0 Index

@, 2
\, 11
<CR>, 2
*, 19
?, 27, 28, 31, 35
$, 1, 22
↑, 1,23
_, 21
←, 21

^A, 23
account, *preface*,2, 41
atsign (@), 2, 41

BABYL, 28
backarrow (←), 21
backward (screen), 7
backslash (\), 11
Bellinda, Melinda, 30
Brotsky's Tbase, 40

capital letters, 2
carriage return, 2
cc, 25
Chaosnet, 13, 41
cleaning (directory), 16
clear (screen), 6, 10, 38
command line, 9, 20
commands, 8, 41
commands (to TeX), 11, 41
confused (hopelessly), 6, 29
continue, 16
conventions, 3
copy, carbon, 25
crash, 9
creating a file, 19
cursor, 6, 41
CTRL (↑), 1, 23
\ctrline, 20

delete, 17, 18, 41
directory, 3, 16, 34, 41
display (math), 22
\double_space, 21
down, 13, 41
Dover, 11
dvrq, 13, 14

editing text, 5
EMACS, *preface*,5
\end, 21
errors, 15

error messages, 15
ESC ($), 1
EXEC, 2, 34, 41
expunge, 17, 18, 41

\fdisplay, 21
files, 3, 41
finger, 33
fork, 18, 41
fork-status, 18
forward (screen), 7, 10
freed up, 18

glossary, 41

\hbox, 9
help, 35, 38
helvetica, 20
\hfill, 15
host, 13

idiosyncrasy (of keyboard), 1
information, 35
input, 20, 41
INQUIRE, 33

jobs, 6, 41

^K, 23
keys, 1, 36
key and command summary, 10, 36
killing jobs, 18, 42
kk <cr>, 19, 36
Knuth's TeX manuals, 39, 40

LaTeX, *preface*,40
LEARN.TXT (file), 4
line, location of in file, 15
location of mistake, 15
logging in, 2
login name, *preface*, 2
Louise, 25
lowercase (commands), 2, 20

mail, 25
malfunction, 6, 13, 29
manuals, 39
math equations, 22
Maxine, 26
mistake, location of, 15
MM, 25
mm commands, 28, 37
mode, math, 22
--More--, 34

NCG, 2, 34

output, 5, 11, 42
OZ, *preface*

--Pause--, 3, 34
paragraph, 21
password, *preface*, 2
pencil tip, 8
point size, 20
\pp, 21
press file, 12, 42
printer, 11, 41
printing, 11
printer's paragraph, 21
programs, *preface*, 2, 11, 42
prompt, 42

queue, 12, 42

R>, 27
reading mail, 27, 31
reap, 18
receiving mail, 25, 27, 31
reply (mail), 27
reset, 18, 19, 26
RETURN, 2
rubout, 8

S>, 26
samples (input and output), 24
SAIL (execution), 12
save (file), 6, 9, 13, 36
scratch, creating a file from, 19
sending mail, 25, 28
space bar, 3, 34
spooler, 13
Stallman's EMACS manual, 39
starting, 2
stuck (screen), 13

TeX, *preface*,11, 40
TeX80, *preface*, 40
TeX82, *preface*, 40
text editing, 5
text style math, 22
top level, 5, 42
trouble (with BABYL), 29
trouble (with EMACS), 6
typefaces, 20, 40

underscore, 21
up, 13, 42
uppercase (commands), 2, 20

version (of file), 4, 17, 20, 42
\vskip, 20

wake up (terminal), *preface*, 42
WATSON, 33
WBG.TXT, 4
Whitman, Walt, 5
WHOIS, 33
wildcard, 19

END

FILMED

10-85

DTIC