MASSACHUSETTS INSTITUTE OF TECHNOLOGY
A. I. LABORATORY

Artificial Intelligence
Memo No. 215 April 1971

HOW TO GET ON TO THE SYSTEM

A GUIDE TO THE A. I. LAB TIMESHARING SYSTEM FOR NEW USERS

Mark Dowson

Work reported herein was conducted at the Artificial Intelligence Laboratory, a Massachusetts Institute of Technology research program supported by the Advanced Research Projects Agency of the Department of Defense and was monitored by the Office of Naval Research under Contract Number N00014-70-A-0362-0002.

How To Get On To The System

Introduction

This memo is intended to get very new users started on the MAC AI system. It presents some simple rituals for making and editing files,getting print out,making microtapes and so on. Most of the rituals given are not the only ways of doing something or even neccessarily the simplest,but they do work.

Some sources of more detailed documentation are referenced at the end of this memo:read them when you want to know more.

If you don't understand something or need any kind of help *ask*. No one minds;they all know how you feel.

Cautionary Note

This is a research system intended for maximum flexibility. There are few built in safeguards and it is not idiot proof. It is up to *you* not to screw other users. If in doubt or if you get an error message you don't understand *ask*.

Microtapes are every users back up storage. They are used to safeguard data and programs which must not be lost at any cost so be especially careful with all tape routines.

Getting a User Name

Go to Minsky's secretary and fill out all the appropriate forms. Your user name should be known to the system in about a week. If it takes much longer ask someone what to do. For the rest of this memo we'll pretend that your user name is USER.

A Note On Notation

Everything you type on a console will be shown below in capital letters and indented e.g.

```
    FOO
```

Everything the system types back will be also in caps and indented a bit more e.g.

```
        BAR
```

Arbitrary strings inside commands are shown by lower case strings e.g.

```
    FOOarbitrarystringBAR
```

Spaces by _ as in

```
    FOO_BAR
```

Carriage returns by

```
    cr
```

Control characters,produced by hitting the CTRL key and the appropriate character key simultaneously by e.g.

<u>ctrl</u>A

<u>ctrl</u>K

Alt mode,got by hitting the ESC key,is represented by a circled dollar sign: ($)

Don't confuse this with the dollar sign key (shift 4) even though they both print the same when you hit the keys.

Logging In and Out

Find a free console which is showing

```
    ITS 671 CONSOLE n FREE
```

(Here and elsewhere the precise message given by the system may vary.)

Now type:

```
  ctrlZ
```

The console will respond:

```
    DDT 274
```

followed by any message directed at the whole user community.

Now type your user name followed by ($) U

```
  USER ($) U
```

You are now logged in to DDT which is the top level program in the system. Typing: ctrlZ at any time will bring you back up to DDT and it is from here that you log out by typing

```
   ($)($) U
```

The system will respond with

```
    ITS 671 CONSOLE n FREE
```

and you can go home.

Always log out before you leave the console.

Making a TECO File.

TECO is a character editor. It lets you construct and edit files and file them on disc or tape for later use. Type:

```
    TECOctrlK           the system will respond:
      TECO 100cr
```

Now you are in TECO. All TECO commands are terminated by `$$` . This says, in effect, 'Now do it.' Until you type the two alt modes you can quit (cancel) any command string by typing

```
    ctrlG
```

TECO filing and reading commands all start with an 'E'. Try:

```
    EY $$
```

This will display your file directory which should have at least one file in it (otherwise the system will garbage collect you) Lets make another one. 'I' is the insert command:

```
    Ialmostanystring$$
```

inserts 'almostanystring' in the buffer. The string can include any combination of keyboard characters, spaces, carriage returns etc except ctrlZ or alt mode. Try it. By the way, if you hit a wrong key while typing in a string, you can erase the n preceding characters by hitting the 'RUBOUT' key n times. The characters you've rubbed out will be echoed at the top left of your console.

You haven't made a file yet - you still have to get the contents of the buffer into the output buffer, give it a name and file it. Do:

```
    EW $$
```

Now look at your file directory again (EY `$$`). You've created a temporary file called TECO OUTPUT. Be careful, though,it will disappear without trace if you give it a chance. Now choose a name for your file. Files must have two names,filename1 and filename2,separated by a space. For reasons which will become clear,filename2 is almost invariably a number. We'll name this one TRYOUT_1. Type:

```
    EETRYOUT_1 $$
```

and look at your file directory again. There it is.

The complete ritual,so far,goes:

```
    ctrlZ
    USER $ U
    TECOctrlK
    Icontentsofthefile $$
    EW $$
    EEfilename1_1 $$
```

Editing a TECO File

Before you can edit a file you must read it back into the buffer. Type:

```
    ERTRYOUT_1 $ Y $$
```

Incidently the 'Y' after the first alt mode controls where the file is read to and how. A 'Y' clears the buffer before 'yanking' the new file into it;an 'A' in this position would append the file to the bottom of anything already in the buffer;an ' @' would make a listing of the file on the Line Printer.

All editing is done with respect to a pointer which can be moved around the contents of the buffer and which can be seen on the display as /\.
Pointer moving commands are as follows:

COMMAND	EFFECT ON POINTER
`J $$`	go to the top of buffer contents
`ZJ $$`	go to the bottom of buffer contents
`±nL $$`	move ±n lines,pointer left at beginning of the line. Note that n can be 0,and that n=1 is assumed in default.
`nC $$`	move n characters forward
`nR $$`	move n characters reverse
`Sstring $$`	search forwards for 'string'pointer goes to right hand end of string.
`-Sstring $$`	search backwards for 'string'pointer to left end

Now you know how to move the pointer around you can edit the contents of the file. Editing commands are:

`±nD $$`	delete ±n characters from the pointer
`±nK $$`	kill (delete) ±n lines from the pointer
`HK $$`	kill entire buffer contents

and of course

`Istring $$`	insert 'string'after the pointer.

Note that `K $$` will delete the rest of the line after the pointer including the carriage return. `±99K $$` is useful for deleting everything above or below some point in the buffer.

Greater and Greater

Now you want to file the edited version of TRYOUT 1. If you refile it under the same name and screw anything up you will lose the original as well. So file it as TRYOUT 2. Wait,though,you don't have to keep track of which number you've got up to. Use '>' as filename2 and TECO will create filename2 one greater than the previous number. You can use this for reading as well e.g.

```
EW $$
EETRYOUT_> $$
ERTRYOUT_> $ Y $$
(Do some more editing)
EW $$
EETRYOUT_> $$
```

If you look at your file directory after all this you'll find you're up to TRYOUT 3 and still have TRYOUT 1 and 2.

When you've finished for the day and are about to log out,delete all the junk (previous versions of files) with ED.

```
EDfilemame1_filename2 $$        deletes a file.
```

You can use '<' to remove the <u>least</u> numbered file:

```
EDfilename1_< $$ .
```

Page After Page After Page

So far,the files have been single pages. To make a multi-page file:

```
    Itextofpage1 $$             insert text of first page
    EW$$                        make somewhere for it to go or you'll lose
it
    P $$                        put it out there
    Itextofpage2 $$             insert text of second page
    P $$
    (etc etc)
    EEfilename1_> $$            name and file all the pages.
```

To read and edit a multi-page file:

```
    ERfilename1_> $ Y $$          yank the first page into the buffer
    EW $$                         if you are going to re-file an edited
version
    (edit page 1)
    P $$                          put out the first page and get the next
one into the buffer
    (edit page 2)
    P $$
    (etc etc)
    EEfilename1_> $$              file the edited version.
```

If you had just wanted to read the file,not edit or re-file it,you could have missed out the EW and done `Y$$`'s instead of the `P$$`'s to get each new page into the buffer,losing the previous buffer contents down the drain where all unwanted duplicates of data go.

To turn a single page into several insert <u>ctrl</u>L wherever you want a page boundary before re-filing it.

To condense a multi-page file into one page get the new pages after the first into the buffer with `AA$$` instead of P or Y.

To stitch two files together into one:

```
ERfile1_>$Y$$             read the first file
EW$$
99P$$                     get all the pages out
ERfile2_>$A$$             'A' for append
EEnewfilename_>$$
```

Finally,

```
ERfilename1_filename2$@$$
```

will put the whole file out on the line printer. (You can still use '>' as filename2 if it's appropriate.)

Evading the Grim File Gobbler

Files stored on disk have been known to disappear in really bad system crashes so anything you really want to keep,put on Microtape.

1)Get yourself a clean,marked,microtape. (Ask.)

2)Write your name on the reel. Every kind of ink I've found so far rubs off so a sticky label may be the answer.

3)Find a free tape drive.

4)Switch the off/write-lock/write switch to 'off'.

5)Load the tape. (Get someone to show you how the first time.)

6)Switch to 'write'.

7)Note the number of the tape drive. Are you sure?

Now,back at your console,give the tape a name. (One day you may have several tapes)

Do:

```
nESnam$$
```

'n' is the tape drive number. Are you <u>sure</u>?

'nam' is any three character name for the tape.

Now you can write on the tape:

```
0ERfilename1_filename2$nEIYEE$$
```

The first '0' indicates the source device (0 for disk)

The 'n' indicates the destination device;put the tape drive number here.

Filename2 must be explicit. Do not use '>'.

Now do EY$\textcircled{\$}\textcircled{\$}$ to look at your tape file directory. So far we've been taking advantage of TECO assuming you mean 'disk' unless told otherwise and then remembering what you last told it. To look at your disk file directory again you must say:

EYDSK: $\textcircled{\$}\textcircled{\$}$

And back to the tape directory:

EYUTn: $\textcircled{\$}\textcircled{\$}$

where 'n' is the tape drive number.

Lastly,to flap (remove) a tape,do:

EK $\textcircled{\$}\textcircled{\$}$

This should run the tape back onto it's reel and you can remove it (switch to 'off' first).

Check THAT IT HAS RUN RIGHT BACK.

<u>NEVER</u> remove a tape that hasn't been flapped fully.

If in doubt ASK FOR HELP.

Reading Other Users Files

Simple if you remember TECO assumes that if you don't tell it something you mean the same as last time.

For instance:

```
EY$$
```

is an abbreviation for

```
EYDSK:username;$$
```

In default TECO assumes the 'DSK' and remembers the last user name it was told (when you logged in). So to look at FRED's file directory do:

```
EYDSK:FRED;$$
```

Now anything you say will refer to FRED's files unless you tell it otherwise e.g. by:

```
ERTRYOUT_>_DSK:USER;$Y$$
```

Search Macros

A useful feature when editing TECO files is a facility that lets you search for every occurrence of a given string on a page in the buffer and do something (delete,insert etc) when it is encountered.

For instance to replace every occurrence of 'A' by 'SOME' do:

`J$$` to get to the top of the buffer if you're not already there,then:

```
<S_A_$;2RDISOME$>$$
```

This says:search for each occurrence of spaceAspace in the buffer,move two characters left,delete one character to the right and insert SOME.

In general the form is:

```
    <Sstring$;operationsIstring$>$$
```

You can miss out the 'Istring$' completely (or the operations for that matter).

Q Registers

Useful for juggling blocks of text or holding strings that you're going to insert all over the place.

```
    HK$$
```

clear the buffer (if you need to)

```
    Istring$$
    HXq$$
```

where 'q' is the name of the Q register,one character from A to Z or a number from 0 to 9.

The string remains in the selected Q register until you log out or until you change it. (or until you kill the job - see below)

```
    Gq$$
```

inserts the Q register contents wherever you (the pointer) are.

```
    n<Gq>$$
```

will insert n copies of the Q register contents in the buffer.

How To Do n Things At Once

We'll use some useful things as examples.

You're in TECO and have just written a PLANNER program.To help debug it you want to 'PrettyPrint' it. Do:

```
    ctrlZ
```

to get into DDT. Now do:

```
    NLISPctrlK
```

The system responds

```
        ALLOC ?
```

to find out how much storage you want.Type

```
    N
```

for 'normal' Now read the program which will do the work - 'GRIND' :

```
    (UREAD_GRIND_>_COM)
```

System responds:

```
        (COM USER)
```

Load the program by typing ctrlQ and when it is loaded type:

```
    (EVAL(GRIND1_filename1_>_DSK))
```

The program will tell you when it's finished by:

```
        (DSK USER)
```

to say that it has filed the new version back in TECO. Now you can look at it or print it out.Do :

```
    ctrlZ
```

to get back into DDT. YOU DON'T HAVE TO RELOAD TECO. Do:

```
    $J
```

The system responds:

```
    TECO $\textcircled{\$}$ J
```

Now do:

```
  $\textcircled{\$}$ P
```

and you are back in TECO. You still have the LISP - you can see this by doing a PEEK. ctrlZ again and type:

```
    PEEKctrlK
```

You'll see that under your user name are the names of the three jobs that you have going,TECO, LISP and PEEK. PEEK is the only one using any time. You can get back to any one of the others exactly where you left off by going back to DDT,and doing $\textcircled{\$}$ J's until you get to the one you want (This is called 'going round the job ring') and then doing $\textcircled{\$}$ P to get down into it. This is particularly useful if you are running,say, a PLANNER program (Which takes a long time to reload) but want to use TECO for a short period or to PEEK to find out what all those other users are doing to make the system so slow. By the way,it's antisocial to keep jobs open that you don't intend to use for a long time,since it takes up valuable core.You can kill a job from DDT by moving round the job ring till you get to it and doing:

```
    :KILLcr
```

which kills the job and moves you round the job ring to the next one. You can kill PEEK by doing :

```
    Q
```

at any time while you're running it :this takes you up to DDT and kills it in one fell swoop.

It'll Keep Till Tomorrow

Normally,when you log out,you lose the current state of the program you were working with. If it is,say ,a LISP that you've put a lot of work into you'll be reluctant to leave it even if it means missing dinner and/or annoying your wife. So here is a way to save the complete state of a program on disk so you can get it back tomorrow

1) Invent a file name for it.
2) Dump it on your disk with:

```
$Y filename1_filename2 cr
```

Now it is a TECO disk file so you can log out safely. To get it back again tomorrow:

1) From DDT,name the job

```
jobname$J
```

You might use YLISP (for 'yesterdays Lisp') as the job name for example. note that you don't have to load a new LISP as the complete state of your previous LISP is on disk.

2) Load the file with

```
$L filename1_filename2 cr
```

3) Start at starting location with

```
$G
```

Now you are back exactly where you left off yesterday. Don't forget to ED the file from TECO when you don't need it any more - it's taking up lots of room.

Send, Communicate and Mail

Sometimes,while you are in the middle of working away at a console,a (usually rude) message will mysteriously appear in front of you. Wouldn't you like to send people rude messages? Well,this is how.

To get into 'communicate' mode: type:

```
ctrlbackarrow C address_
```

where 'address' is either a user name or teletype number (backarrow is shift O ;yes,thats right,you were pressing three keys at once then). The system will respond with a 'G' for go or'B' for busy.If it was G,your console is now tied to the other users console until one of you types:

```
ctrlbackarrow N
```

When you start to communicate with someone your message is prefixed by:

```
MESSAGE FROM USER
```

just to let them know that they are now in communicate mode too.

'SEND' will send a message one way without tying the consoles together. Do:

```
:SEND_username_message ctrlC
```

The message will be prefixed on their console by:

```
MESSAGE FROM USER HACTRN
```

Of course the people you send messages to must be currently logged in. You can check quickly by doing:

```
TTYctrlF
```

from DDT. This doesn't take you out of DDT by the way. If you want to send a message to someone who's not logged in you can use MAIL. Do:

```
   :MAIL_username_message ctrlC
```

and they will get the message the next time they log in.

Further Documentation

Some sources of further documentation are listed below.

From DDT type:

```
INFOctrlK
```

and then:

```
TECO$\textcircled{\$}$ I
```

to get an up to date listing of TECO commands.Much useful information is reputed to live in INFO in any case.

The following A.I. Memos should be available in the ninth floor document room:

A.I. Memo 81	PDP-6 TECO
A.I. Memo 147	A Multiple Procedure DDT
A.I. Memo 161A	ITS 1.5 Reference Manual (This contains details of PEEK amongst other things)

Some people have copies of an anonymous draft internal note called 'Using the MAC PDP-6 Time-Sharing system' get a Xerox copy if you can.

This Memo will be supplemented,as and when,with memos bearing the same number and an alphabetic suffix.

The most useful source of documentation is your own notes on what other users tell you.

Finally,this Memo was produced using TJ6. See A.I. Memo 164A for details.