AN INTRODUCTION TO THE COMPUTER FACILITIES OF THE GMU CENTER FOR ARTIFICIAL INTELLIGENCE

P. A. Stefanski

MLI88-9

A publication of *Machine Learning & Inference Laboratory*
Artificial Intelligence Center
George Mason University
Fairfax, VA 22030 USA
(703) 764-6259

Editor: R. S. Michalski
Assistant Editor: J. Zhang

MLI Reports replaces *ISG Reports* published until December, 1987 by the Artificial Intelligence Laboratory Department of Computer Science, University of Illinois Urbana, IL 61801.

AN INTRODUCTION TO THE COMPUTER FACILITIES OF THE GMU CENTER FOR ARTIFICIAL INTELLIGENCE.

Pawel A. Stefanski

Artificial Intelligence Center
George Mason University,
Fairfax, VA 22030.

MLI 88-10
TR-16-88

November 1988

AN INTRODUCTION TO THE COMPUTER FACILITIES OF THE GMU CENTER FOR ARTIFICIAL INTELLIGENCE.

ABSTRACT

This document provides an introduction to the computer facilities of the Artificial Intelligence Center (AIC). Briefly described are also facilities of GMU Computer Center, available to members of the AIC. It is strongly biased towards SUN workstations and UNIX environment, which is the main operating system used in the AIC, but should be helpful to all members and collaborators.

ACKNOWLEDGEMENTS

The final version of this document[1] is a joint effort of several peoples. Professors: Ken DeJong, Ryszard S. Michalski, Jan Zytkow and Igor Mozetic provided numerous valuable remarks and corrected the earlier versions. Ken Kaufman and Jianping Zhang also suggested several improvements. Jerzy Bala is a joint author of the graphic schema of the AIC equipment.

This work was done in the Center for Artificial Intelligence at George Mason University. Research in the Center is supported in part by the Defence Advanced Research Projects Agency under grant administered by the Office of Naval Research No. N00014-87-K-0874, in part by the Office of Naval Research under grant N00014-88-K-0226, and in part by the Office of Naval Research under grant N00014-88-K-0397.

---

[1] As of October, 20, 1988. This is a "living" document, which will evolve over time to record the new equipment and changes in its setup.

1. AN INTRODUCTION.

This document provides an introduction to the computer facilities of the Artificial Intelligence Center (AIC). For completeness, facilities of GMU Computer Center that are available to members of the AIC are also briefly described[1]. It is intended to serve as a high-level guide, not a substitute for the detailed documentation. This is available for check-out in the Center Administrative Assistant Office (Rm. 301) - for MAC and Symbolics computers, and in Rm.312 - for SUN and VAX.

The Artificial Intelligence Center is located in the Science and Technology Building, on the 3rd floor, in rooms 301 (Center Main Office - Director and Administrative Assistant), 312 (Research Assistants room), 326 (Conference room and offices) and 328 (Computer room and offices). In addition, two Symbolics machines (3640) are located in Rm. 324. Members of the AIC can also use the departmental laser printer in Rm.316, which is connected to the Apple-Talk network.

Currently, the Center consists of one research laboratory, Machine Learning and Inference (MLI), which collaborates with various research groups in the Computer Science Department and other departments of George Mason University.

2. GENERAL OVERLOOK OF THE AIC FACILITIES.

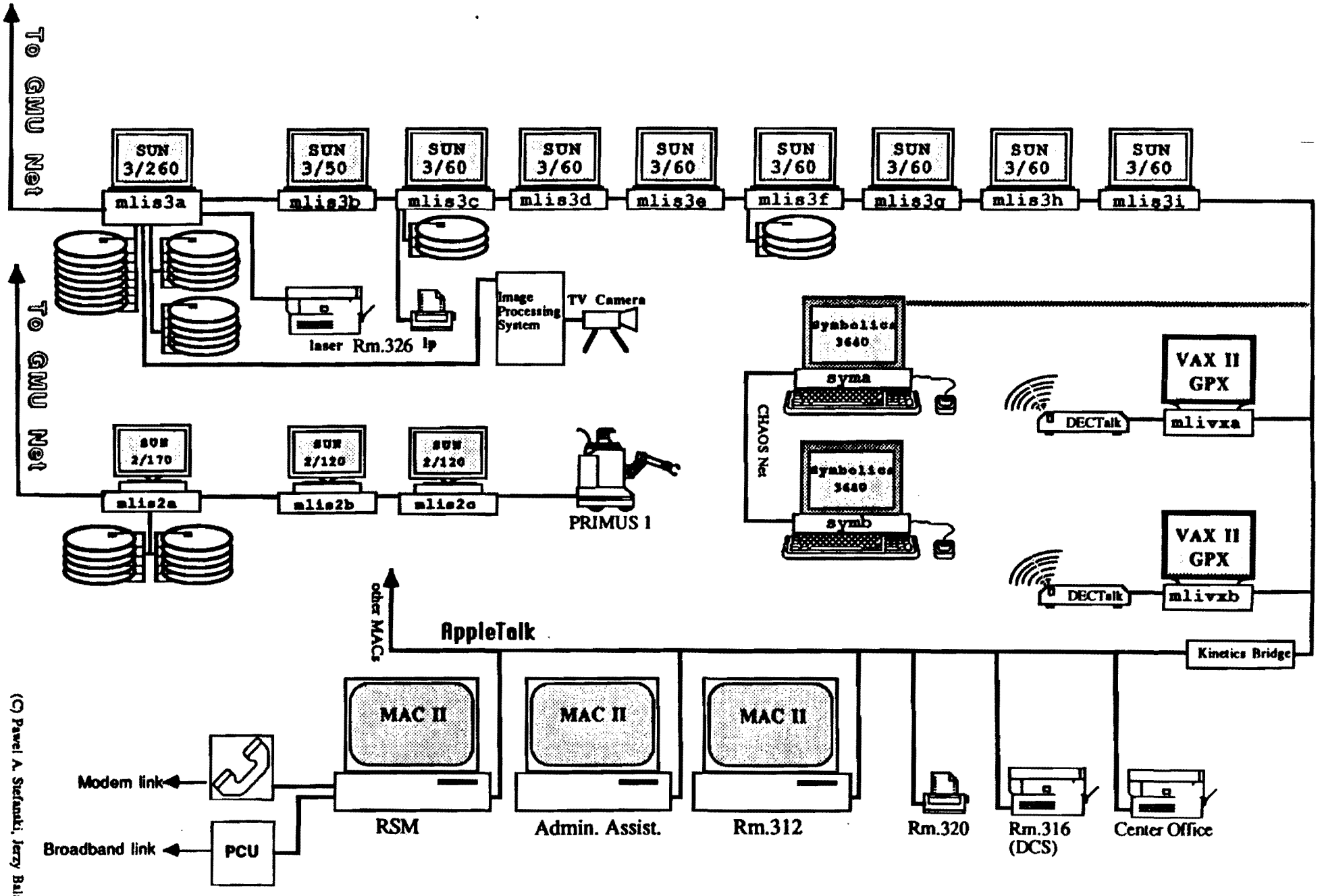
Main computing facilities of the AI Center are located in rooms 328 and 326, with some other workstations scattered around in 312, 301 and other rooms. The schema on the next page shows most of the available resources.

2.1. The AIC Network.

The Artificial Intelligence Center currently provides to its members:

---

[1] For more complete information about university computing facilities see attached "Guide to University Computing And Information Systems".



Center for Artificial Intelligence

(a) A network of eleven SUN-3 machines: one SUN 3/260 file server (mlis3a - 12MB memory), nine SUNs 3/60 (mlis3c - mlis3k, 8MB memory each) and 1 SUN 3/50[1] (mlis3b - 4MB memory). The file server has two disk drives 280 MB each, and one Eagle drive - 580 MB, which gives 1GB of total disk space. Two SUNs 3/60 (mlis3c and mlis3f) have local SCSI disk drives, 140 MB each, which are remotely mounted by other machines (as /usr2 and /usr3 respectively). The SUN 3/260 has also a 1/4" cartridge tape drive[2].

There are two printers available on this net (both are located in Rm.326): PRISM dot-matrix printer (default), and Laser Writer (available as 'laser')[3]. The dot-matrix printer is to be used for draft versions of the documents and program printouts. The laser printer is used only for final, high-quality printouts, as its maintenance is quite expensive.

(b) A network of three SUN-2 machines, located in Rm. 328 and Rm. 312, used mainly for text processing and as a backup net. It consists of the file server, SUN-2/170 (mlis2a - 4MB of memory), with 2 disk drives 280 MB each, and two clients, mlis2b and mlis2c (mlis2c is located in Rm. 312).

(c) Two Symbolics 3640, running Genera 7.2.[4] - those machines are not yet connected to the network.

(d) Two MAC-II computers, with LaserView displays and 1MB of memory each[5]. They are soon to be connected (through the Kinetics bridge) to the Ethernet.

(e) Two microVAX II/GPX workstations (mlivxa and mlivxb), connected to the same LAN as SUN-3, located in Rm.326, under Ultrix. Each of them has a speech generator (DECTalk). They are used for the demonstrations of the Emerald system: an integrated system of learning programs.

---

[1] Located in Prof. DeJong's office.

[2] The reel tape drives are in Thompson Hall (available from gmuvax2 as /dev/mt0) - call x3569 to schedule their use and ask for help.

[3] This LaserWriter is also available from gmuvax2 (VAX8530) and gmu90x (Pyramid) as 'laser1'.

[4] For a nice introduction to Symbolics terminology see "Genera Concepts" (vol. 1A of the documentation set).

[5] Listed here are only MAC's, that are generally available. There are two more - one for the Administrative Assistant's use, and the other one - in Prof. Michalski's office. This same concerns the laser printer in the Administrative Assistant's office, which is unavailable to other people.

2.2. Available Software.

Presented below is a partial list of software available on the machines (excluding Symbolics) in the AIC network.

MAC II:

Editors: Microsoft Word, MacWrite, WriteNow, Canvas, MacDraw, MacPaint, PageMaker, Illustrator

Languages: Allegro Common Lisp, Mac Turbo Pascal, Assembler, Hypercard

Applications: RedRyder, Kermit, Comserve (for communications)
MAC dBASE III and FOX (database packages)
Hypercard, TOPS networking and spooling.

SUN workstations:

Editors: ed, ex and vi, GNU Emacs with emacstool (ver.18.50), textedit (under SunWindows), TeX, FrameMaker[1].

Languages: SUN Lucid Common Lisp (development environment 2.1), Portable Common Loops, PASCAL (pc), C (cc, cpp), Kyoto Common Lisp (kcl), C-Prolog (cprolog).

Utilities:[2] vttool (emulator VT100), TOPS, contool, mantool, ice (interactive color editor), Xwindows (ver.11), fig (Mac-like figure editor)[3], various filters.

MicroVAX II workstations:

---

[1] Intended to be purchased soon.

[2] For hackers: all utilities (executables) are in /usr/local/bin.

[3] For a detailed description of those and others local utilities see manual pages in /usr/man/man1 - for example, to see the on-line documentation for ice, type: "man ice".

Editors: ed, ex and vi.

Languages: C, Ultrix Lisp, Pascal.

3. UNIVERSITY COMPUTING FACILITIES AVAILABLE TO AIC MEMBERS.

The main university computers are located in Tompson Hall. Currently, GMU Computer Center has: VAX 8800 (gmuvax) under VMS 4.7, VAX 8530 (gmuvax2) under UNIX (Ultrix-32), and Pyramid (gmu90x) under UNIX(AT&T System V). The following is a partial list of software available on those machines:[1]

Editors:	gmuvax	EDT, Emacs
	gmuvax2	vi, Emacs
	gmu90x	vi, Emacs
Languages:	gmuvax	Ada, C, Cobol, VaxLisp, Pascal.
	gmuvax2	C, Franz Lisp, Modula-2, Pascal, Prolog.
	gmu90x	C, Franz Lisp, Pascal, Prolog.
Applications:	gmuvax	Ingres DBMS, Kermit, Mail, Ocam, Sas, SPSS-X, GPSS, TeX, LaTeX.
	gmuvax2	Kermit, Mail, Macsyma, Simscript, TeX, LaTeX.
	gmu90x	Kermit, Mail, OPS5, Yacc.

The laser printers ("`lpr -Plaser fname`" under UNIX, "`print/que=laser fname.ftype;fversion`" under VMS) are available on those machines - printouts have to be picked-up in the basement of Thompson Hall.

All university computers are available through Internet from SUNs, and from terminals in rooms 312, 316, 324. There are two types of terminals currently in use: HP2621 and TVI925 - in order to use full-screen editing and other facilities remember to set the TERM

---

[1] Source: ACS Newsletter, Vol.9 No. 3, Spring 1988 - for a complete list see "Guide To University Computing".

variable accordingly (for GNUEmacs, use: "`set term = terminal-type`" under UNIX, "`def emacsvar_term = terminal-type`" under VMS).

4. INTERNET MAIL, NEWS GROUPS AND WORKING ENVIRONMENT.

Currently, electronic mail is available on gmuvax, gmuvax2 and gmu90x. For those, who have an account on gmuvax2 the e-mail address is: *login-name*@gmuvax2.gmu.edu.

Local news groups are:[1]

• gmu.announce - official announcements of new and changed features on GMU computing systems.

• gmu.local - general group for the discussion of issues concerning GMU computing including "how to use" hints, problems, questions, concerns and announcements.

Each user, upon establishing his(her) new account (on SUNs) will get sample startup files - by convention, their names start with a dot "." ('.login', '.cshrc', '.suntools', etc.). Depending on the individual needs, people responsible for particular machines can help with setting-up the correct environment.

If you are new...

to UNIX, read:	"Getting Started with UNIX: Beginner's Guide"
	"Setting up Your UNIX Environment: Beginner's Guide"
	"Self Help with Problems: Beginner's Guide"
to mail, read:	"Mail and Messages: Beginner's Guide"
to network, read:	"Using the Network: Beginner's Guide"
to SunWindows, read:	"Windows and Windows-Based Tools: Beginner's Guide"
to VMS, read:	"Using the VAX 8800"[2]
to Symbolics, read:	"Lisp Lore - A Guide to Lisp Machine".

As a general introduction to SUN read: "Sun System Overview"[3].

---

[1] If you are new to news, type "man rn" (under UNIX). Later read carefully articles in the news groups: "mod.announce.newusers" and "news.groups"

[2] This publication is available through the university bookstore for $1.45.

[3] You may also refer to other books available in the AIC library, like "Unix Papers for Power Users" or UNIX command reference manual.

If you want to know more...

about UNIX, read: "Doing More with UNIX"
"UNIX Papers"
"The UNIX System: A Sun Technical Report"[1]

about SUN-3, read: "SUN-3 Architecture: A Sun Technical Report"

about SUN environment, read: "SunPro: The Sun Programming Environment: A Sun Technical Report"

5. HELPFUL HINTS.

5.1. Comparison Chart of Basic Commands in UNIX and VMS.

	UNIX	VMS
Login	in both systems you are asked to type in your login name (prompt: "login:" or "USERNAME:") and password (prompt: "password:").	
Logout	"logout"	"logout" or "bye"
Change password	"passwd"[2]	"set password"
File edit/create[3]	"vi *fn*"	"edit *fn.ftype*"
File copy	"cp *fnold fnnew*"	"copy *fnold fnnew*"
File remove	"rm *fn*"	"delete *fn.ftype;fver*"
File rename	"mv *fnold fnnew*"	"rename *fnold fnnew*"
List file contents	"more *fn*"	"type *fn.ftype*"
Print file contents	"lpr *fn*"	"print *fn.ftype;fver*"[4]
Create directory	"mkdir *dirn*"	"create dir [*dirn*]"[5]
List directory contents	"ls *dirn*"	"directory [*dirn*]"[6]

---

[1] Technical Reports are available in Rm. 312.

[2] Note, that on SUN's, when yellow pages service is running, you should use the command "yppasswd", which takes care to update correctly the main database.

[3] '*fn*' stands for the file name you want to use.

[4] If you want to use a laser printer, type (assuming, that the laser printer name is "laser"):
under UNIX: "lpr -Plaser *fname*"
under VMS: "print /que=laser *fname.ftype;fversion*"

[5] Directory structures in both VMS and UNIX are tree-like, but the syntax is different:
in UNIX: "." denotes the current directory, ".." denotes the parent directory (if one exists), and the directory names are joined using "/", so for example: "/usr/mlis3a/stefan/admin" describes the directory "admin" in directory "stefan" in directory "mlis3a", etc.
in VMS: [-] denotes the parent directory, while [.name] relates to the subdirectory "name" in the current directory, and the directories are joined using ".".

[6] In both systems, directory name, when omitted, defaults to the current directory.

Change directory	"cd *dirn*"	"set default [*dirn*]"
Get help	"man *topic*"[1]	"help"

5.2. The Organization of UNIX Reference Manuals.

Traditionally, UNIX reference manuals are divided into 8 sections. They are further subdivided in some cases, look for an introduction to a given part to check details (type "man *section-no intro*" to read the on-line introduction). The sections are:

(1) publicly accessible system commands (this section is of most interest to all users)

(2) introduction to system calls and error numbers (very technical)

(3) library routines (for C and Fortran programmers only)

(4) special files and hardware support (technical)

(5) file formats used by various programs (technical, though advanced users can benefit from some descriptions)

(6) computer games and demos (don't expect anything fancy here)

(7) public files (look for ASCII tables here) and macros (for troff fans)

(8) introduction to system maintenance and operation (for administrators).

5.3. What Commands to Look for on a Given Topic.

TOPIC	COMMAND(S)
Edit code	ed, ex, vi, emacs, textedit, indent
Search for patterns	grep, fgrep, sed, awk
Debug programs	dbx, dbxtool
Print files	pr, lpr, lpq, lprm, col, enscript
Display files	tail, head, wns, more, cat
Manipulate files	cp, mv, rm, ln
Save/restore files on tape	tar, dump, restore
Change files access permissions	chmod, chgrp
Logging into other machines	rlogin, telnet, tip
Copy files between machines	rcp, ftp, tftp, cpio
Use SunWindows	suntools, toolplaces
Organize and store code	sccs, make

[1] Topic is usually a name of a command, try "man man" for the first time.

Use news[1] rn

5.4. Using TEX and LaTEX.

TeX and LaTeX are available on the VAX 8530 and on SUN-3 net. To access them, prepare the TeX source in a file of the form "filename.tex". The following examples assume that the file is in this format.

This sequence will prepare and print a TeX file (on gmuvax2):

```
tex filename                          << converts from TeX to dvi
ln03dvi filename                      << converts from dvi to ln03
lpr -Plaser filename.ln3              << prints the formatted output
```

This sequence will prepare and print a LaTeX file (on gmuvax2):

```
latex filename                        << converts from LaTeX to dvi
ln03dvi filename                      << converts from dvi to ln03
lpr -Plaser filename.ln3              << prints the formatted output
```

This sequence will prepare and print a TeX file (on SUN-3):

```
tex filename                          << converts from TeX to dvi
dvi2ps filename.dvi >filename.lpr     << converts from dvi to PostScript
lpr -Plaser -h filename.lpr           << prints the formatted output
```

[1] Currently available only on gmuvax2, gmuvax (VMS version) or gmu90x.

6. Rules and Regulations.

In order to ensure an efficient and effective working environment in the Artificial Intelligence Center, the basic list of rules, which has to be complied with by everyone using the equipment, has been established. We are working hard to keep everything in (at least) approximate order. So it should be clear, that we do not want our work to be damaged. Therefore:

A. When you print files, either take the printouts with you, or dispose of them appropriately. Do not leave them on the desks, floor etc.

B. Do not leave your belongings (notes, manuals, pens) on the desks, on top of the file servers or monitors. In addition to the mess, you can seriously overheat (read: damage) the hardware, increase maintenance expense and decrease productivity.

C. Report all equipment malfunctions to the system administrator(s) after entering them in the log-book. Do not attempt to correct the malfunction yourself, unless you have been authorized by the administrator. When adding new software, changing an existing one, deleting files not owned by you (or performing other similar tasks) always contact the administrator first!

D. Except for the tape cartridges, you are free to use all the other facilities, unless otherwise posted. However, if you are asked not to do something, please obey the request. New cartridge requests should be given to the system administrator.

E. All equipment users will be held accountable for reading carefully the "message-of-the-day" (in the file */usr/local/motd* on *mlis3a*).

F. Simple work requests for administrator(s) will be completed in one day, complicated work requests may take up to one week.

G. The administrators reserve the right to enforce these rules by revoking the privilege of logging into the machines.

List of Appendices.

Appendices

A1. SUN reference cards.

A2. Last SUN newsletter.

A3. GNU Emacs reference cards.

A4. GNU Emacs tutorial.

A5. Approximate layout of AI Center rooms and phone numbers.

A6. Where to call for help.

A7. GNUecho command description.

Getting Started With UNIX: Quick Reference

This quick reference lists the commands presented in this manual concisely by function. Each listing includes a syntax diagram, and a brief description of the command.

1. Work Session

1.1. Log In

Type username to system login prompt.
Type password to password prompt.

1.2. Change Password

Type `passwd`, followed by old password, and repeat new password.

1.3. Log Out

Type `logout` or CTRL-D depending upon system setup.

2. File System

2.1. Create File

Type `cat >`*filename*, then text ending with CTRL-D, or see Editing Files.

2.2. Make (or Create) Directory

Type `mkdir` *directory-name*.

2.3. Look at File

Type `cat` *filename*
or `more` *filename*.

2.4. Print File

Type `lpr` *filename*.

2.5. List Files and Directories

Type

`ls` for listing of current directory

`ls` *directory-name*
for listing of another directory

`ls` *filename*
for listing of a single file

`ls -t` or

`ls -t` *filename* or

`ls -t` *directory-name*
to get a listing reverse sorted by time of last modification

`ls -F` or

`ls -F` *directory-name*
to get a listing that marks directory names by appending a / character to them.

2.6. Move (or Rename) Files and Directories

Type

`mv` *source-filename destination-filename*
to rename a file

`mv` *source-filename destination-directory*
to move a file into another directory

`mv` *source-directory-name destination-directory-name* to rename a directory, or move it into another directory.

2.7. Copy Files

Type

`cp` *source-filename destination-filename*
to copy a file into another filename

`cp` *source-filename destination-directory*
to copy a file into another directory

2.8. Remove (or Delete) File

Type

`rm` *filename*
to remove a file

`rmdir` *directory-name*
to remove an empty directory

`rm -r` *directory-name*
to remove a directory and its contents.

2.9. Change Working Directory

Type

`cd` to change directories to your home directory

`cd` *directory-name*
to change directories to another directory.

2.10. Find Name of Current Directory

Type `pwd`.

2.11. Pathnames

simple:	One filename or directory name to access local file or directory.
absolute:	List of directory names from root directory (first /) to desired filename or directory name, each name separated by /.
relative:	List of directory names from current position to desired filename or directory name, each name separated by /.

2.12. Directory Abbreviation

~	Home directory.
~*username*	Another user's home directory.
.	Working directory.
..	Parent of working directory.

3. Commands

3.1. Date and Time

Type `date`. For universal time (Greenwich Mean Time), type `date -u`.

3.2. Calendar

Type

`cal` *year*
for yearly calendar

`cal` *month-number year*
for monthly calendar

3.3. Wild Cards

`?` Single character wild card.

`*` Arbitrary number of characters.

3.4. Redirecting Output

System types output of command to file rather than screen, replacing current contents of file, if any. Type *command-name* > *filename*.

System types output of command to file rather than screen, appending to current contents of file, if any. Type *command-name* >> *filename*.

3.5. Basic Calculator

Type `bc` to enter interactive program. Type arithmetic expressions, using `+`, `-`, `*`, and `/` symbols, followed by (RETURN). To change number of decimal places, type `scale =` *number*.

4. Editing Files

Type `vi` to enter text editor, then any of following commands (in command mode, unless preceded by a `:`):

`a` to add text

`cc` to substitute a line with a string (enters insert mode)

`cw` to substitute, or change, a word with a string (enters insert mode)

`dd` to delete the entire line the cursor is on

`dw` to delete the word, or portion of word, under and after the cursor

`h` to move left, or "west," one character

`i` to insert text under the cursor (enters insert mode)

`j` to move down, or "south," one line

`k` to move up, or "north," one line

`l` to move right, or "east," one character

`o` to insert text on a new blank line after the current line (enters insert mode)

`O` to insert text on a new blank line before the current line (enters insert mode)

`s` to substitute a character with a string (enters insert mode)

`x` to delete the character under the cursor

`:q` to quit `vi`

`:q!` to quit `vi`, without writing changes

`:w` to save, or write a file

5. Formatting Files

Construct source file to run through `nroff` formatter, including any of the following commands:

`.LP` to *left-justify* a paragraph

`.IP` to created an *itemized* paragraph (like this one)

`.ce` to center text on the page

`.ul` to underline portions of text

`.sp` to create a blank line space

`.br` to force the end of a line, a line break

To format the source file, type `nroff -ms` *source-filename*. You will probably want to redirect the output of `nroff` into a *destination-filename*, so you can print it out afterward.

6. Search Files

Type

`grep` *search-string filename*
to type out lines containing the string in a specific file

`grep` *search-string filename(s)*
to type out lines containing the string in more than one file

`grep -v` *search-string filename(s)*
to type out lines that don't contain the string

7. Timesavers

7.1. Aliases

To "alias," or abbreviate a command string with an alias string, type `alias` *alias-string command-string*.

8. History: Command Repetition

`!!` Repeat the entire last command line at any point in the current command line.

`!$` Repeat the last word of the last command line at any point in the current command line.

9. Run Command in Background: Job Control

To run a command in the background, as opposed to the more common method of running commands in the foreground, type a `&` after the command line. Then, you can type more commands to the command prompt, or even run more commands in the background for simultaneous command execution.

10. Online Documentation

To see online Man Pages, type `man` *command-name*.

Doing More With UNIX: Quick Reference

This quick reference lists commands presented in this manual, including a syntax diagram and brief description.

1. Files

1.1. Filename Substitution

Wild Cards[1]	`?` `*`
Character Class	`[`*c*`...]`
Range	`[`*c*`-`*c*`]`

c is any single character.

String Class	`{`*str*`[,` *str*`]}`

str is a combination of characters, wild cards, embedded character classes and embedded string classes.

Home Directory	`~`
Home Directory of Another User	`~`*user*
List Hidden Files	`ls -[l]a`

1.2. File Properties

Seeing Permissions	`ls -l` *filename*
Changing Permissions	`chmod` *nnn filename*
	`chmod` *c=p...*`[,`*c=p...*`]` *filename*

n, a digit from 0 to 7, sets the access level for the user (owner), group, and others (public), respectively. *c* is one of: `u` – user, `g` – group, `o` – others, or `a` – all. *p* is one of: `r` – read access, `w` – write access, or `x` – execute access.

Setting Default Permissions	`umask` *ugo*

ugo is a (3-digit) number. Each digit restricts the default permissions for the user, group and others, respectively.

[1] from *Getting Started With UNIX*

Changing Modification Time	`touch` *filename*
Making Links	`ln` *oldname newname*
	`ln -s` *oldname newname*
Seeing File Types	`ls -F`

1.3. Encrypting Files

Source Files	`crypt <` *source* `>` *encrypted*
Editing	`vi -x` *encrypted*
Decrypting Files	`crypt <` *encrypted* `| more`
	`crypt <` *encrypted* `>` *text*

`crypt` asks for the encryption key.

1.4. Searching with `more`

Run `more`	`more` *filename*
Next Line[1]	RETURN
Next 11 Lines[1]	`d`
Next Page[1]	SPACE
Search for Pattern	*/pattern*
Next Occurrence	`n`
Next File	`:n`

1.5. The Directory Stack[2]

Change Directory, Push	`pushd` *directory*
Change to Top Directory, Pop	`popd`
Show Stack	`dirs`

2. Commands

2.1. Command-Line Special Characters

Quotes and Escape	
Join Words	`"..."`
Suppress Filename, Variable Substitutions	`'...'`
Escape Character	`\`

[2] a feature of the C Shell

Separation, Continuation	
Command Separation	`;`
Command-Line Continuation	`\`RETURN

2.2. I/O Redirection and Pipes

Standard Output	`>`
	`>!`
Appending to Standard Output	`>>`
	`>>!`
Standard Input	`<`
Standard Error and Output	`>&`
Standard Error Separately	
	`(` *command* `>` *output* `) >&` *errorfile*
Pipes/Pipelines	*command* `|` *filter*`[ |` *filter*`]...`
Duplicating Displayed Output	
	command `| tee` *filename*

Filters	
Word/Line Count	`wc [-l]`
First *n* Lines	`head [-`*n*`]`
Last *n* Lines	`tail [-`*n*`]`
Skip to Line *n*	`tail [+`*n*`]`
Show Nonprinting Characters	`cat -v`
Sort lines	`sort [-`*n*`]`
Format Paragraphs	`fmt`
Reverse Character Order	`rev`
Multicolumn Output	`pr -t`
List Spelling Errors	`spell`
Substitutions in Output Stream	
	`sed -e "s/`*pattern*`/`*string*`/[g]"`
Report-Generation	`awk`[3]

2.3. Searching with `grep`

`grep` Command	`grep "`*pattern*`"` *filename*
	command `| grep "`*pattern*`"`

[3]

grep Search Patterns

beginning of line	^
end of line	$
any single character	.
single character in list or range	[...]
character not in list or range	[^...]
zero or more of preceding character or pattern	*
zero or more of any character	.*
escapes special meaning	\

3. C-Shell Features

3.1. History Substitution

The History List

Set Up History List	`set history=`*n*
See History List	`history [-h]`

Event Designators

Repeat Previous Command	!!
Display Previous Command	!!:p
Command Line *n*	!*n*
n Commands Back	!-*n*
Command Beginning with *str*	!*str*
Command Containing *str*	!?*str*[?]
All Arguments to Prev. Command	!*
Last Argument to Prev. Command	!$
First Argument to Prev. Command	!^
n'th Argument	!:*n*

Word Designators

All Arguments	:*
Last Argument	:$
First Argument	:^
n'th Argument	:*n*
Arguments *x* Through *y*	:*x*-*y*

Modifiers

Print Command Line	:p
Substitute Command Line	:[g]s/*l*/*r*/

3.2. Aliases

`alias` Command `alias` *name 'definition'*

definition can contain escaped history substitution event and word designators as placeholders for command-line arguments.

3.3. Variable Substitution

Creating a Variable	`set` *var*
Assigning a Value	`set` *var* = *value*
Expressing a Value	$*var*
Displaying a Value	`echo` $*var*

value is a single word, an expression in quotes, or an expression that results in a single word after variable, filename and command substitution takes place.

Assigning a List	`set` *var* = (*list*)

list is a space-separated list of words, or an expression that results in a space-separated list.

Selecting the *n*'th Item	$*var*[*n*]
Selecting all Items $*var*	
Selecting a Range	$*var*[*x*-*y*]
Item Count	$#*var*

3.4. `foreach` Lists

Start `foreach` Loop `foreach` *var* (*list*)

`foreach` prompts for commands to repeat for each item in *list* (with >), until you type `end`. Within the loop, $*var* stands for the current item in *list*.

3.5. Command Substitution

Replace Command with its Output on the Command Line `...`

3.6. Job Control

Run Command in the Background	&
Stop Foreground Job	CTRL-Z
List of Background Jobs	`jobs`
Bring Job Forward	%[*n*]
Resume Job in Background	%[*n*]

4. Processes

Listing	`ps -[aux]`
Terminating	`kill [-9]` *PID*
Timing	`time` *command*
Scheduling	`at` *time*[a|p] *script*

time is a number up to 4 digits. *script* is the name of a file containing the command line(s) to perform.

5. Users

Seeing Who Is Logged In	`who` `w`
Changing Identities	`su` [*username*]

Seeing Your User Name	`whoami`
	`who am i`
	`who is this`

6. Managing Files

6.1. Looking Up Files

Standard Commands	`whereis` *file*
Aliases and Commands	`which` *command*
Describe Command	`whatis` *filename*

Searching Out Files

`find` *dir* `-name` *name* `-print`

dir is a directory name within which to search. *name* is a filename to search for.

6.2. Tracking Changes

Comparing Files — `diff` *leftfile rightfile*

`diff` prefixes a left angle-bracket (<) to selected lines from *leftfile* and a right angle bracket (>) to lines from *rightfile*.

Auditing Changes

Putting Files Under `sccs`

```
mkdir SCCS
chmod 775 SCCS
sccs create filename ...
rm ,*
```

Checking Files Out — `sccs edit` *filename* ...

Checking Files In — `sccs delget` *filename* ...

Backing Files Out — `sccs unedit` *filename* ...

Recovering Current Versions

```
sccs get SCCS
```

Reviewing Pending Changes

```
sccs diffs filename ...
```

6.3. Automating Tasks

Create a Makefile — `vi Makefile`

A makefile consists of macro definitions and targets.

Test Makefile — `make -n` [*target*]

Run `make` — `make` [*target*]

6.4. Managing Disk Usage

Seeing Disk Usage

```
df
du -s
du | sort -r -n
ls -l
```

Making A Tape Archive

```
tar -cv[f drive] file ...
```

Extracting Archived Files

```
tar -xv[f drive] file ...
```

[3] See Chapter 6 for details.

7. Printing

7.1. The Printer Queue

List the Queue — `lpq`

Removing a Printer Job — `lprm` *job*

Removing Your Printer Jobs — `lprm -`

Selecting a Printer

```
lpr -Pprinter
lpq -Pprinter
lprm -Pprinter job
```

7.2. Printing `troff` Output and Screen Dumps

`troff` Output — `lpr -t`

Screen Dumps

```
screendump [| rastrepl] | lpr -v
```

Windows and Window-Based Tools: Quick Reference

This quick reference describes the text facility operations of the mouse and the function keys.

The Mouse

Each of the mouse buttons has a general purpose:

LEFT	*selects*
MIDDLE	*adjusts* (or *modifies*) a selection
RIGHT	pops up a menu

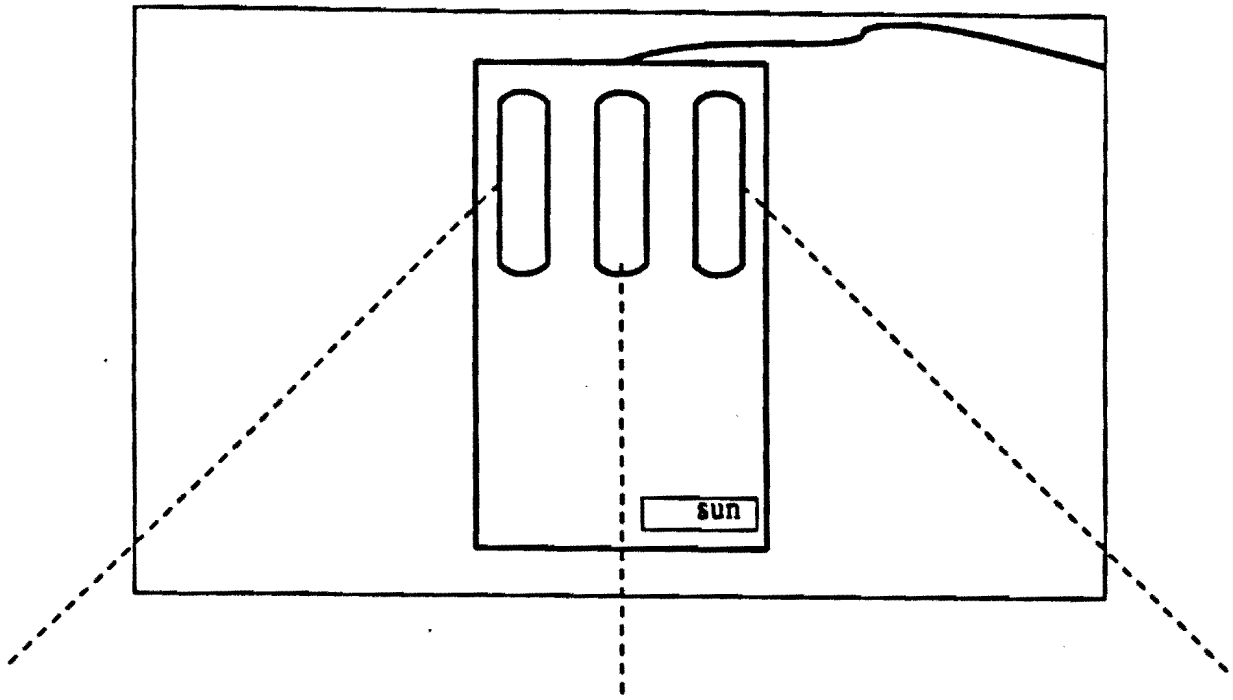
The SHIFT key toggles the "direction" of the action.

The CTRL key modifies the meaning of a choice or accelerator.

Mouse By Function

Desired Function	*Mouse Button Procedure*
open	LEFT on icon
expose	LEFT on frame (border or namestripe)
hide	SHIFT-LEFT on frame (border or namestripe)
move	MIDDLE on corner = unconstrained, vertical and horizontal adjustment MIDDLE on middle third of edge = constrained, movement either horizontally or vertically
resize	CTRL-MIDDLE on corner = unconstrained, vertical and horizontal adjustment CTRL-MIDDLE on middle third of edge = constrained, resizing either horizontally or vertically
zoom full length	CTRL-LEFT on frame (border or namestripe) toggles (switches back and forth) this feature
menu access	RIGHT and hold down, on frame (border or namestripe)

Mouse By Button



MIDDLE	On corner = unconstrained move, vertical and horizontal adjustment On middle third of edge = constrained move, either horizontally or vertically
SHIFT-MIDDLE	n.a.
CTRL-MIDDLE	On corner = unconstrained resize, vertical and horizontal adjustment On middle third of edge = constrained resize, either horizontally or vertically

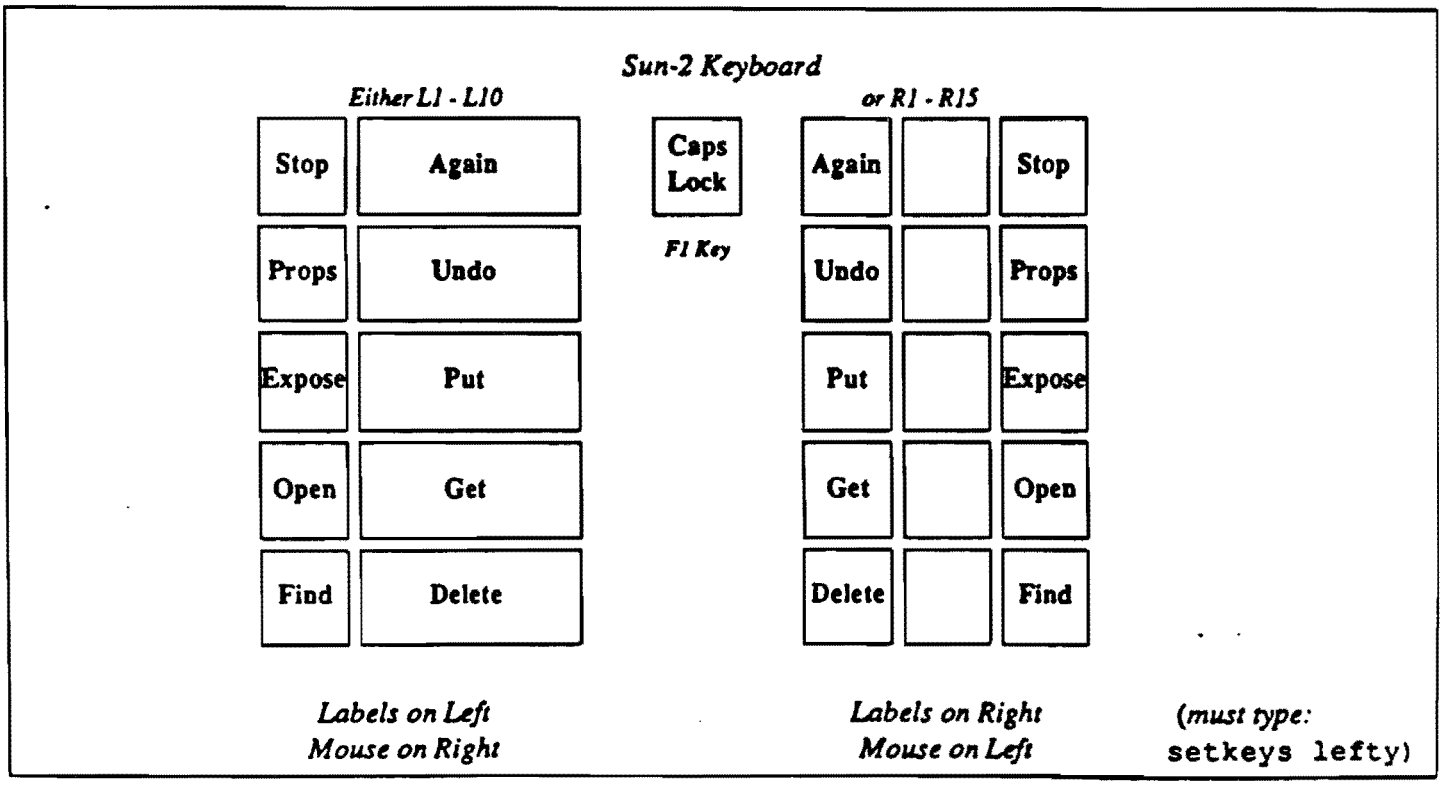
LEFT	On icon: open On frame (border/namestripe): expose
SHIFT-LEFT	On frame (border/namestripe): hide
CTRL-LEFT	On frame (border/namestripe): toggles zoom (full length)

RIGHT or SHIFT-RIGHT or CTRL-RIGHT	On icon or frame (border/namestripe): pop up a menu

Basic Editing

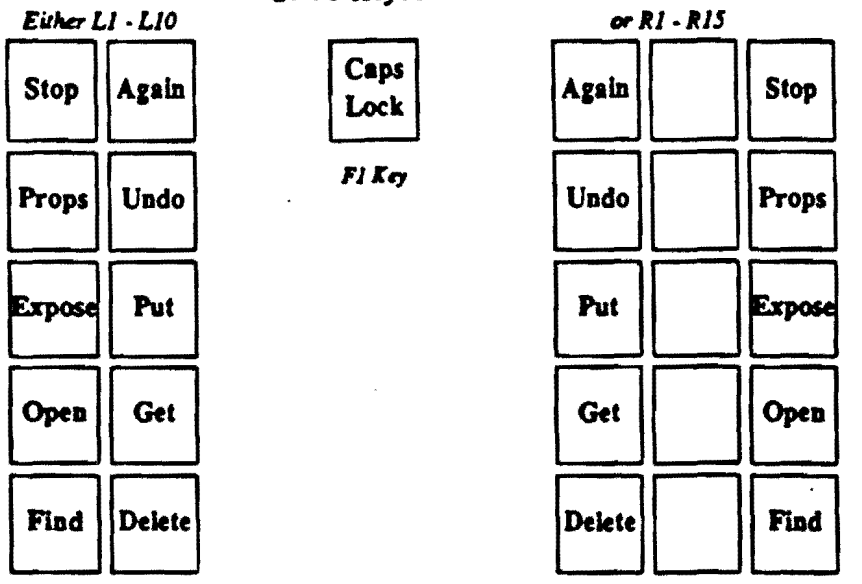
Select	LEFT to select, MIDDLE to adjust.
Scroll	LEFT in scroll button to scroll forward one line, RIGHT to scroll backward, MIDDLE to scroll a page.
Insert	Type characters.
Delete	Delete function key. With characters adjacent to the caret, you can use UNIX rubout, erase word, or line kill characters.
Reinsert deleted text	Get or CTRL-G.
Replace	Delete, type characters.
Move	Delete, select destination, Get.
Store on shelf	Put.
Copy from shelf	Get or CTRL-G.
Copy text from elsewhere	Select text, Put, select destination, Get.
Undo edits	Undo undoes all edits since last selection, or use of Get, Put, or Delete.
Load a file	`Load file` item on text menu, or type name of file to empty text subwindow, followed by ESC. `Reset` text menu item (once or twice) makes window empty.
Save	`Save` or `Store to named file` item on text menu.
Exit	`Quit` item on frame menu.

Function Keys



Function Keys *cont.*

Sun-3 Keyboard

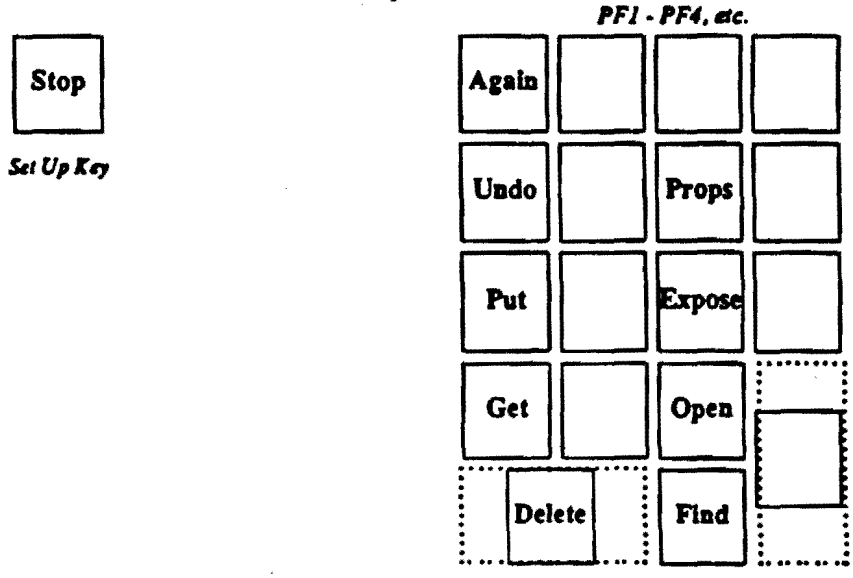


Labels on Left
Mouse on Right

Labels on Right
Mouse on Left

(must type: `setkeys lefty`)

Sun-1 Keyboard



Right side of keyboard
for left- and right-handed people

(must type: `setkeys sun1`)

A2. The Last SUN Newsletter.

MLILAB SUN NEWSLETTER No. 5

August, 18, 1988.
Newsletter editor: Pawel A. Stefanski.

An introduction to new setup and file system.

Since yesterday the new file system on SUN 3/260 is up and running. It can supports up to 9 clients, and uses three disks, all connected to SUN 3/260: xy0, xy1 and xy2. Also, each machine, by default, mounts two other file systems, from mlis3c (on /usr2) and mlis3f (on /usr3) - right now they are intended for general use (this will probably change later). Though it may appear like we have lots of disk space now, keep in mind, that several people are joining us soon, so please clean your directories often!

Both printers (ids line printer and laser printer) are (at least should be) available from all the machines. Some major files have been moved from different places to /usr/local, and now:
icons are in **/usr/local/icons** directory
images are in **/usr/local/images** directory
lisp is in **/usr/local/lisp** directory
all other local executables are in **/usr/local/bin** directory.

I advise you (even the experienced users) to look into **/usr/mlis3a/admin/newuser** directory, where a set of 'vanilla' startup files has been provided, and either copy them to your home directory, or customize yours accordingly.

There is a new message file, common to all the machines, **/usr/local/motd**, where I will try to put all the important messages, so be sure to include the code to read it in your **.login** file!

As always, all the bugs are mine, so whenever something is wrong, call me at 764-6057.

And enjoy !

GNU Emacs Reference Card

(for version 18)

Starting Emacs

To enter Emacs, just type its name: emacs

To read in a file to edit, see Files, below.

Leaving Emacs

suspend Emacs (the usual way of leaving it)	C-z
exit Emacs permanently	C-x C-c

Files

read a file into Emacs	C-x C-f
save a file back to disk	C-x C-s
insert contents of another file into this buffer	C-x i
replace this file with the file you really want	C-x C-v
write buffer to a specified file	C-x C-w
run Dired, the directory editor	C-x d

Getting Help

The Help system is simple. Type C-h and follow the directions. If you are a first-time user, type C-h t for a tutorial. (This card assumes you know the tutorial.)

get rid of Help window	C-x 1
scroll Help window	ESC C-v
apropos: show commands matching a string	C-h a
show the function a key runs	C-h c
describe a function	C-h f
get mode-specific information	C-h m

Error Recovery

abort partially typed or executing command	C-g
recover a file lost by a system crash	M-x recover-file
undo an unwanted change	C-x u or C-_
restore a buffer to its original contents	M-x revert-buffer
redraw garbaged screen	C-l

Incremental Search

search forward	C-s
search backward	C-r
regular expression search	C-M-s

Use C-s or C-r again to repeat the search in either direction.

exit incremental search	ESC
undo effect of last character	DEL
abort current search	C-g

If Emacs is still searching, C-g will cancel the part of the search not done, otherwise it aborts the entire search.

Motion

Cursor motion:

entity to move over	backward	forward
character	C-b	C-f
word	M-b	M-f
line	C-p	C-n
go to line beginning (or end)	C-a	C-e
sentence	M-a	M-e
paragraph	M-[	M-]
page	C-x [	C-x]
sexp	C-M-b	C-M-f
function	C-M-a	C-M-e
go to buffer beginning (or end)	M-<	M->

Screen motion:

scroll to next screen	C-v
scroll to previous screen	M-v
scroll left	C-x <
scroll right	C-x >

Killing and Deleting

entity to kill	backward	forward
character (delete, not kill)	DEL	C-d
word	M-DEL	M-d
line (to end of)	M-0 C-k	C-k
sentence	C-x DEL	M-k
sexp	M-- C-M-k	C-M-k

kill region	C-w
kill to next occurrence of *char*	M-z *char*
yank back last thing killed	C-y
replace last yank with previous kill	M-y

Marking

set mark here	C-@ or C-SPC
exchange point and mark	C-x C-x
set mark *arg* words away	M-@
mark paragraph	M-h
mark page	C-x C-p
mark sexp	C-M-@
mark function	C-M-h
mark entire buffer	C-x h

Query Replace

interactively replace a text string	M-%
using regular expressions	M-x query-replace-regexp

Valid responses in query-replace mode are

replace this one, go on to next	SPC
replace this one, don't move	,
skip to next without replacing	DEL
replace all remaining matches	!
back up to the previous match	^
exit query-replace	ESC
enter recursive edit (C-M-c to exit)	C-r

Multiple Windows

delete all other windows	C-x 1
delete this window	C-x 0
split window in 2 vertically	C-x 2
split window in 2 horizontally	C-x 5
scroll other window	C-M-v
switch cursor to another window	C-x o
shrink window shorter	M-x shrink-window
grow window taller	C-x ^
shrink window narrower	C-x {
grow window wider	C-x }
select a buffer in other window	C-x 4 b
find file in other window	C-x 4 f
compose mail in other window	C-x 4 m
run Dired in other window	C-x 4 d
find tag in other window	C-x 4 .

Formatting

indent current line (mode-dependent)	TAB
indent region (mode-dependent)	C-M-\
indent sexp (mode-dependent)	C-M-q
indent region rigidly *arg* columns	C-x TAB
insert newline after point	C-o
move rest of line vertically down	C-M-o
delete blank lines around point	C-x C-o
delete all whitespace around point	M-\
put exactly one space at point	M-SPC
fill paragraph	M-q
fill region	M-g
set fill column	C-x f
set prefix each line starts with	C-x .

Case Change

uppercase word	M-u
lowercase word	M-l
capitalize word	M-c
uppercase region	C-x C-u
lowercase region	C-x C-l
capitalize region	M-x capitalize-region

The Minibuffer

The following keys are defined in the minibuffer

complete as much as possible	TAB
complete up to one word	SPC
complete and execute	RET
show possible completions	?
abort command	C-g

Type C-z ESC to edit and repeat the last command that used the minibuffer. The following keys are then defined

previous minibuffer command	M-p
next minibuffer command	M-n

GNU Emacs Reference Card

Buffers

select another buffer	C-x b
list all buffers	C-x C-b
kill a buffer	C-x k

Transposing

transpose characters	C-t
transpose words	M-t
transpose lines	C-x C-t
transpose sexps	C-M-t

Spelling Check

check spelling of current word	M-$
check spelling of all words in region	M-x spell-region
check spelling of entire buffer	M-x spell-buffer

Tags

find tag	M-.
find next occurrence of tag	C-u M-.
specify a new tags file	M-x visit-tags-table
regexp search on all files in tags table	M-x tags-search
query replace on all the files	M-x tags-query-replace
continue last tags search or query-replace	M-,

Shells

execute a shell command	M-!
run a shell command on the region	M-|
filter region through a shell command	C-u M-|
start a shell in window *shell*	M-x shell

Rmail

scroll forward	SPC
scroll reverse	DEL
beginning of message	. (dot)
next non-deleted message	n
previous non-deleted message	p
next message	M-n
previous message	M-p
delete message	d
delete message and back up	C-d
undelete message	u
reply to message	r
forward message to someone	f
send mail	m
get newly arrived mail	g
quit Rmail	q
output message to another Rmail file	o
output message in Unix mail style	C-o

Regular Expressions

The following have special meaning inside a regular expression.

any single character	. (dot)
zero or more repeats	*
one or more repeats	+
zero or one repeat	?
any character in set	[...]
any character not in set	[^ ...]
beginning of line	^
end of line	$
quote a special character c	\c
alternative ("or")	\\|
grouping	\(... \)
nth group	\n
beginning of buffer	\`
end of buffer	\'
word break	\b
not beginning or end of word	\B
beginning of word	\<
end of word	\>
any word-syntax character	\w
any non-word-syntax character	\W
character with syntax c	\sc
character with syntax not c	\Sc

Registers

copy region to register	C-x x
insert register contents	C-x g
save point in register	C-x /
move point to saved location	C-x j

Info

enter the Info documentation reader	C-h i

Moving within a node:

scroll forward	SPC
scroll reverse	DEL
beginning of node	. (dot)

Moving between nodes:

next node	n
previous node	p
move up	u
select menu item by name	m
select nth menu item by number (1-5)	n
follow cross reference (return with l)	f
return to last node you saw	l
return to directory node	d
go to any node by name	g

Other:

run Info tutorial	h
list Info commands	?
quit Info	q
search nodes for regexp	s

Keyboard Macros

start defining a keyboard macro	C-x (
end keyboard macro definition	C-x)
execute last-defined keyboard macro	C-x e
append to last keyboard macro	C-u C-x (
name last keyboard macro	M-x name-last-kbd-macro
insert lisp definition in buffer	M-x insert-kbd-macro

Commands Dealing with Emacs Lisp

eval sexp before point	C-x C-e
eval current defun	C-M-x
eval region	M-x eval-region
eval entire buffer	M-x eval-current-buffer
read and eval minibuffer	M-ESC
re-execute last minibuffer command	C-x ESC
read and eval Emacs Lisp file	M-x load-file
load from standard system directory	M-x load-library

Simple Customization

Here are some examples of binding global keys in Emacs Lisp. Note that you cannot say "\M-8"; you must say "\e8".

```
(global-set-key "\C-cg" 'goto-line)
(global-set-key "\e\C-r" 'isearch-backward-regexp)
(global-set-key "\e8" 'query-replace-regexp)
```

An example of setting a variable in Emacs Lisp:

```
(setq backup-by-copying-when-linked t)
```

Writing Commands

```
(defun (command-name) ((args))
  "(documentation)"
  (interactive "(template)")
  (body))
```

An example:

```
(defun this-line-to-top-of-screen (line)
  "Reposition line point is on to the top of
the screen.  With ARG, put point on line ARG.
Negative counts from bottom."
  (interactive "P")
  (recenter (if (null line)
                0
              (prefix-numeric-value line))))
```

The argument to `interactive` is a string specifying how to get the arguments when the function is called interactively. Type C-h f interactive for more information.

VIP Quick Reference Card

Searching and Replacing

search forward for *pat*	/ *pat*
search backward for *pat*	? *pat*
repeat previous search	n
... in the opposite direction	N
incremental search	C-s
reverse incremental search	C-r
replace	R
query replace	Q
replace a character by another character *c*	r *c*

Modifying Commands

The delete (yank, change) commands explained below accept a *motion* command as their argument and delete (yank, change) the region determined by the motion command. Motion commands are classified into *point commands* and *line commands*. In the case of line commands, whole lines will be affected by the command. Motion commands will be represented by *m* below.

The point commands are as follows:

h l 0 ^ $ e w b B e E ( ) / ? ' f F t T % ; ,

The line commands are as follows:

j k + - H M L { } G '

Delete/Yank/Change Commands

	delete	yank	change
region determined by *m*	d *m*	y *m*	c *m*
... into register *x*	" *x* d *m*	" *x* y *m*	" *x* c *m*
a line	d d	Y or y y	c c
current region	d r	y r	c r
expanded region	d R	y R	c R
to end of line	D	y $	c $
a character after point	x	y l	c l
a character before point	DEL	y h	c h

Put Back Commands

Deleted/yanked/changed text can be put back by the following commands

Put back at point/above line	P
... from register *x*	" *x* P
put back after point/below line	p
... from register *x*	" *x* p

Repeating and Undoing Modifications

undo last change	u or :und
repeat last change	. (dot)

Undo is undoable by u and repeatable by . For example, u... will undo 4 previous changes. A . after 5dd is equivalent to 5dd, while 1 after 5dd is equivalent to 3dd.

Miscellaneous Commands

	shift left	shift right	filter shell command	indent
region	< *m*	> *m*	! *m shell-com*	= *m*
line	< <	> >	! ! *shell-com*	= =

emulate ESC/C-h in emacs mode	ESC/C-h
emulate C-c/C-x in emacs mode	C/X
Join lines	J
lowercase region	# c *m*
uppercase region	# C *m*
execute last keyboard macro on each line in the region	# g *m*
insert specified string for each line in the region	# q *m*
check spelling of the words in the region	# s *m*

Differences from Vi

In VIP some keys behave rather differently from Vi. The table below lists such keys, and you can get the effect of typing these keys by typing the corresponding keys in the VIP column.

	Vi	VIP
forward character	SPC	l
backward character	C-h	h
next line at first non-white	RET	+
delete previous character	X	DEL
get information on file	C-G	G
substitute characters	s	x i
substitute line	S	c c
change to end of line	C or R	c $

(Strictly speaking, C and R behave slightly differently in Vi.)

Customization

By default, search is case sensitive. You can change this by including the following line in your .vip file.

```
(setq vip-case-fold-search t)
```

variable	default value
vip-search-wrap-around	t
vip-case-fold-search	nil
vip-re-search	nil
vip-re-replace	nil
vip-re-query-replace	nil
vip-open-with-indent	nil
vip-help-in-insert-mode	nil
vip-shift-width	8
vip-tags-file-name	"TAGS"

Include (some of) following lines in your .vip file to restore Vi key bindings.

```
(define-key vip-mode-map "\C-g" 'vip-info-on-file)
(define-key vip-mode-map "\C-h" 'vip-backward-char)
(define-key vip-mode-map "\C-m" 'vip-next-line-at-bol)
(define-key vip-mode-map " " 'vip-forward-char)
(define-key vip-mode-map "g" 'vip-keyboard-quit)
(define-key vip-mode-map "s" 'vip-substitute)
(define-key vip-mode-map "C" 'vip-change-to-eol)
(define-key vip-mode-map "R" 'vip-change-to-eol)
(define-key vip mode map "S" 'vip substitute line)
```

Ex Commands in VIP

In vi mode, an Ex command is entered by typing:

: *ex-command* RET

Ex Addresses

current line	.	next line with *pat*	/ *pat* /
line *n*	*n*	previous line with *pat*	? *pat* ?
last line	$	*n* line before *a*	*a* - *n*
next line	+	*a* through *b*	*a* , *b*
previous line	-	line marked with *x*	' *x*
entire buffer	%	previous context	' '

Addresses can be specified in front of a command. For example,

:.,.+10m0

moves 11 lines below current line to the end of buffer.

Ex Commands

mark lines matching *pat* and execute *cmds* on these lines	:g /*pat*/ *cmds*
mark lines not matching *pat* and execute *cmds* on these lines	:v /*pat*/ *cmds*
move specified lines after *addr*	:m *addr*
copy specified lines after *addr*	:co (or :t) *addr*
delete specified lines [into register *x*]	:d [*x*]
yank specified lines [into register *x*]	:y [*x*]
put back text [from register *x*]	:pu [*x*]
substitute *repl* for first string on line matching *pat*	:s /*pat*/*repl*/
repeat last substitution	:&
repeat previous substitute with previous search pattern as *pat*	:~
read in a file	:r *file*
read in the output of a shell command	:r! *command*
write out specified lines into *file*	:w *file*
write out specified lines at the end of *file*	:w>> *file*
write out and then quit	:wq *file*
define a macro *x* that expands to *cmd*	:map *x cmd*
remove macro expansion associated with *x*	:unmap *x*
print line number	:=
print version number of VIP	:ve
shift specified lines to the right	:>
shift specified lines to the left	:<
Join lines	:j
mark specified line to register *x*	:k *x*
set a variable's value	:se
run a subshell in a window	:sh
execute shell *command* *command*	:! *command*
find first definition of tag *tag*	:ta *tag*

VIP Quick Reference Card

(for version 3.5 under GNU Emacs version 19)

Loading VIP

Just type M-x vip-mode followed by RET

VIP Modes

VIP has three modes: *emacs mode*, *vi mode* and *insert mode*. Mode line tells you which mode you are in. In emacs mode you can do all the normal GNU Emacs editing. This card explains only vi mode and insert mode. GNU Emacs Reference Card explains emacs mode. You can switch modes as follows.

from emacs mode to vi mode	C-z
from vi mode to emacs mode	C-z
from vi mode to insert mode	i, I, a, A, o, O or C-o
from insert mode to vi mode	ESC

If you wish to be in vi mode just after you startup Emacs, include the line

```
(setq term-setup-hook 'vip-mode)
```

in your .emacs file. Or, you can put the following alias in your .cshrc file

```
alias vip 'emacs \!* -f vip-mode'
```

Insert Mode

Insert mode is like emacs mode except for the following.

go back to vi mode	ESC
delete previous character	C-h
delete previous word	C-w
emulate ESC key in emacs mode	C-z

The rest of this card explains commands in vi mode.

Getting Information on VIP

Execute info command by typing M-x info and select menu item vip. Also:

describe function attached to the key x	C-h h x

Leaving Emacs

suspend Emacs	Z Z or :st
exit Emacs permanently	Z Z or Z C or :q

Error Recovery

abort partially typed or executing command	C-g
redraw messed up screen	C-l
recover a file lost by a system crash	M-x recover-file
restore a buffer to its original contents	M-x revert-buffer

Counts

Most commands in vi mode accept a count which can be supplied as a prefix to the commands. In most cases, if a count is given, the command is executed that many times. E.g., 5 d d deletes 5 lines.

Registers

There are 26 registers (a to z) that can store texts and marks. You can append a text at the end of a register (say x) by specifying the register name in capital letter (say X). There are also 9 read only registers (1 to 9) that store up to 9 previous changes. We will use x to denote a register.

Entering Insert Mode

insert at point	i
append after cursor	a
insert before first non-white	I
append at end of line	A
open line below	o
open line above	O
open line at point	C-o

Buffers and Windows

move cursor to next window	C-n
delete current window	Z 0
delete other windows	Z 1
split current window into two windows	Z 2
show current buffer in two windows	Z 3
switch to a buffer in the current window	b *buffer*
switch to a buffer in another window	B *buffer*
kill a buffer	K
list existing buffers	Z B

Files

visit file in the current window	v *file* or :e *file*
visit file in another window	V *file*
save buffer to the associated file	Z S
write buffer to a specified file	Z W
insert a specified file at point	Z I
get information on the current file	g or :f
run the directory editor	Z d

Viewing the Buffer

scroll to next screen	SPC or C-f
scroll to previous screen	DEL or C-b
scroll down half screen	C-d
scroll up half screen	C-u
scroll down one line	C-e
scroll up one line	C-y
put current line on the home line	z H or z RET
put current line on the middle line	z M or z .

Marking and Returning

mark point in register x	m x
set mark at buffer beginning	m <
set mark at buffer end	m >
set mark at point	m .
jump to mark	m ,
exchange point and mark	` `
... and skip to first non white on line	' '
go to mark x	` x
... and skip to first non-white on line	' x

Macros

start remembering keyboard macro	K (
finish remembering keyboard macro	K)
call last keyboard macro	*
execute macro stored in register x	@ x

Motion Commands

go backward one character	h
go forward one character	l
next line keeping the column	j
previous line keeping the column	k
next line at first non-white	+
previous line at first non-white	-
beginning of line	0
first non-white on line	^
end of line	$
go to n-th column on line	n |
go to n-th line	n G
go to last line	G
find matching parenthesis for (), {} and []	%
go to home window line	H
go to middle window line	M
go to last window line	L

Words, Sentences, Paragraphs

forward word	w or W
backward word	b or B
end of word	e or E

In the case of capital letter commands, a word is delimited by a non-white character.

forward sentence	)
backward sentence	(
forward paragraph	}
backward paragraph	{

Find Characters on the Line

find c forward on line	f c
find c backward on line	F c
up to c forward on line	t c
up to c backward on line	T c

You are looking at the Emacs tutorial.

Emacs commands generally involve the CONTROL key (sometimes labelled CTRL or CTL) or the META key (sometimes labelled EDIT). Rather than write out META or CONTROL each time we want you to prefix a character, we'll use the following abbreviations:

C-<chr> means hold the CONTROL key while typing the character <chr>
Thus, C-f would be: hold the CONTROL key and type f.
M-<chr> means hold the META or EDIT key down while typing <chr>.
If there is no META or EDIT key, type <ESC>, release it, then type the character <chr>. "<ESC>" stands for the key labelled "ALT" or "ESC".

Important note: if you must exit at some point, type C-z.
The characters ">>" at the left margin indicate directions for you to try using a command. For instance:
<<Blank lines inserted here by startup of help-with-tutorial>>
>> Now type C-v (View next screen) to move to the next screen.
(go ahead, do it by depressing the control key and v together).
From now on, you'll be expected to do this whenever you finish reading the screen.

Note that there is an overlap when going from screen to screen; this provides some continuity when moving through the file.

The first thing that you need to know is how to move around from place to place in the file. You already know how to move forward a screen, with C-v. To move backwards a screen, type M-v (depress the META key and type v, or type <ESC>v if you don't have a META or EDIT key).

>> Try typing M-v and then C-v to move back and forth a few times.

SUMMARY

The following commands are useful for viewing screenfuls:

C-v	Move forward one screenful
M-v	Move backward one screenful
C-l	Clear screen and redisplay everything putting the text near the cursor at the center. (That's control-L, not control-1. There is no such character as control-1.)

>> Find the cursor and remember what text is near it.
Then type a C-l.
Find the cursor again and see what text is near it now.

BASIC CURSOR CONTROL

Getting from screenful to screenful is useful, but how do you reposition yourself within a given screen to a specific place? There are several ways you can do this. One way (not the best, but the most basic) is to use the commands previous, backward, forward and next. As you can imagine these commands (which are given to Emacs as C-p, C-b, C-f, and C-n respectively) move the cursor from where it currently is to a new place in the given direction. Here, in a more graphical form are the commands:

```
                        Previous line, C-p
                                :
                                :
   Backward, C-b .... Current cursor position .... Forward, C-f
                                :
                                :
                        Next line, C-n
```

>> Move the cursor to the line in the middle of that diagram and type C-l to see the whole diagram centered in the screen.

You'll probably find it easy to think of these by letter. P for previous, N for next, B for backward and F for forward. These are the basic cursor positioning commands and you'll be using them ALL the time so it would be of great benefit if you learn them now.

>> Do a few C-n's to bring the cursor down to this line.

>> Move into the line with C-f's and then up with C-p's. See what C-p does when the cursor is in the middle of the line.

Lines are separated by Newline characters. For most applications there should normally be a Newline character at the end of the text, as well, but it is up to you to make sure of this. A file can validly exist without a Newline at the end.

>> Try to C-b at the beginning of a line. Do a few more C-b's. Then do C-f's back to the end of the line and beyond.

When you go off the top or bottom of the screen, the text beyond the edge is shifted onto the screen so that your instructions can be carried out while keeping the cursor on the screen.

>> Try to move the cursor off the bottom of the screen with C-n and see what happens.

If moving by characters is too slow, you can move by words. M-f (Meta-f) moves forward a word and M-b moves back a word.

>> Type a few M-f's and M-b's. Intersperse them with C-f's and C-b's.

Notice the parallel between C-f and C-b on the one hand, and M-f and M-b on the other hand. Very often Meta characters are used for operations related to English text whereas Control characters operate on the basic textual units that are independent of what you are editing (characters, lines, etc). There is a similar parallel between

lines and sentences: C-a and C-e move to the beginning or end of a line, and M-a and M-e move to the beginning or end of a sentence.

>> Try a couple of C-a's, and then a couple of C-e's.
Try a couple of M-a's, and then a couple of M-e's.

See how repeated C-a's do nothing, but repeated M-a's keep moving farther. Do you think that this is right?

Two other simple cursor motion commands are M-< (Meta Less-than), which moves to the beginning of the file, and M-> (Meta Greater-than), which moves to the end of the file. You probably don't need to try them, since finding this spot again will be boring. On most terminals the "<" is above the comma and you must use the shift key to type it. On these terminals you must use the shift key to type M-< also; without the shift key, you would be typing M-comma.

The location of the cursor in the text is also called "point". To paraphrase, the cursor shows on the screen where point is located in the text.

Here is a summary of simple moving operations including the word and sentence moving commands:

C-f	Move forward a character
C-b	Move backward a character
M-f	Move forward a word
M-b	Move backward a word
C-n	Move to next line
C-p	Move to previous line
C-a	Move to beginning of line
C-e	Move to end of line
M-a	Move back to beginning of sentence
M-e	Move forward to end of sentence
M-<	Go to beginning of file
M->	Go to end of file

>> Try all of these commands now a few times for practice.
Since the last two will take you away from this screen,
you can come back here with M-v's and C-v's. These are
the most often used commands.

Like all other commands in Emacs, these commands can be given arguments which cause them to be executed repeatedly. The way you give a command a repeat count is by typing C-u and then the digits before you type the command. If you have a META or EDIT key, you can omit the C-u if you hold down the META or EDIT key while you type the digits. This is easier, but we recommend the C-u method because it works on any terminal.

For instance, C-u 8 C-f moves forward eight characters.

>> Try giving a suitable argument to C-n or C-p to come as close
as you can to this line in one jump.

The only apparent exception to this is the screen moving commands,
C-v and M-v. When given an argument, they scroll the screen up or
down by that many lines, rather than screenfuls. This proves to be
much more useful.

>> Try typing C-u 8 C-v now.

Did it scroll the screen up by 8 lines? If you would like to
scroll it down you can give an argument to M-v.

WHEN EMACS IS HUNG

If Emacs gets into an infinite (or simply very long) computation which
you don't want to finish, you can stop it safely by typing C-g.
You can also use C-g to discard a numeric argument or the beginning of
a command that you don't want to finish.

>> Type C-u 100 to make a numeric arg of 100, then type C-g.
Now type C-f. How many characters does it move?
If you have typed an <ESC> by mistake, you can get rid of it
with a C-g.

If you type <ESC> <ESC>, you get a new window appearing on
the screen, telling you that M-ESC is a "disabled command"
and asking whether you really want to execute it. The command
M-ESC is marked as disabled because you probably don't want to
use it until you know more about Emacs, and we expect it would
confuse you if it were allowed to go ahead and run. If you really
want to try the M-ESC command, you could type a Space in answer
to the question and M-ESC would go ahead. Normally, if you do
not want to execute M-ESC, you would type "n" to answer the question.

>> Type <ESC> <ESC>, then type n.

WINDOWS

Emacs can have several windows, each displaying its own text.
At this stage it is better not to go into the techniques of
using multiple windows. But you do need to know how to get
rid of extra windows that may appear to display help or
output from certain commands. It is simple:

C-x 1 One window (i.e., kill all other windows).

That is Control-x followed by the digit 1.
C-x 1 makes the window which the cursor is in become
the full screen, by getting rid of any other windows.

>> Move the cursor to this line and type C-u 0 C-l.
>> Type Control-h k Control-f.
   See how this window shrinks, while a new one appears
   to display documentation on the Control-f command.

>> Type C-x 1 and see the documentation listing window disappear.

INSERTING AND DELETING

If you want to insert text, just type it. Characters which you can see, such as A, 7, *, etc. are taken by Emacs as text and inserted immediately. Type <Return> (the carriage-return key) to insert a Newline character.

You can delete the last character you typed by typing <Rubout>. <Rubout> is a key on the keyboard, which might be labelled "Delete" instead of "Rubout" on some terminals. More generally, <Rubout> deletes the character immediately before the current cursor position.

>> Do this now, type a few characters and then delete them
   by typing <Rubout> a few times. Don't worry about this file
   being changed; you won't affect the master tutorial. This is just
   a copy of it.

>> Now start typing text until you reach the right margin, and keep
   typing. When a line of text gets too big for one line on the
   screen, the line of text is "continued" onto a second screen line.
   The backslash at the right margin indicates a line which has
   been continued.
>> Use <Rubout>s to delete the text until the line fits on one screen
   line again. The continuation line goes away.

>> Move the cursor to the beginning of a line and type <Rubout>. This
   deletes the newline before the line and merges the line onto
   the previous line. The resulting line may be too long to fit, in
   which case it has a continuation line.
>> Type <Return> to reinsert the Newline you deleted.

Remember that most Emacs commands can be given a repeat count; this includes characters which insert themselves.

>> Try that now -- type C-u 8 * and see what happens.

You've now learned the most basic way of typing something in Emacs and correcting errors. You can delete by words or lines as well. Here is a summary of the delete operations:

<Rubout>	delete the character just before the cursor
C-d	delete the next character after the cursor
M-<Rubout>	kill the word immediately before the cursor
M-d	kill the next word after the cursor
C-k	kill from the cursor position to end of line

M-k kill to the end of the current sentence

Notice that <Rubout> and C-d vs M-<Rubout> and M-d extend the parallel started by C-f and M-f (well, <Rubout> isn't really a control character, but let's not worry about that). C-k and M-k are like C-e and M-e, sort of, in that lines are opposite sentences.

Now suppose you kill something, and then you decide that you want to get it back? Well, whenever you kill something bigger than a character, Emacs saves it for you. To yank it back, use C-y. You can kill text in one place, move elsewhere, and then do C-y; this is a good way to move text around. Note that the difference between "Killing" and "Deleting" something is that "Killed" things can be yanked back, and "Deleted" things cannot. Generally, the commands that can destroy a lot of text save it, while the ones that attack only one character, or nothing but blank lines and spaces, do not save.

For instance, type C-n a couple times to postion the cursor at some line on this screen.

>> Do this now, move the cursor and kill that line with C-k.

Note that a single C-k kills the contents of the line, and a second C-k kills the line itself, and make all the other lines move up. If you give C-k a repeat count, it kills that many lines AND their contents.

The text that has just disappeared is saved so that you can retrieve it. To retrieve the last killed text and put it where the cursor currently is, type C-y.

>> Try it; type C-y to yank the text back.

Think of C-y as if you were yanking something back that someone took away from you. Notice that if you do several C-k's in a row the text that is killed is all saved together so that one C-y will yank all of the lines.

>> Do this now, type C-k several times.

Now to retrieve that killed text:

>> Type C-y. Then move the cursor down a few lines and type C-y again. You now see how to copy some text.

What do you do if you have some text you want to yank back, and then you kill something else? C-y would yank the more recent kill. But the previous text is not lost. You can get back to it using the M-y command. After you have done C-y to get the most recent kill, typing M-Y replaces that yanked text with the previous kill. Typing M-y again and again brings in earlier and earlier kills. When you have reached the text you are looking for, you can just go away and leave it there. If you M-y enough times, you come back to the starting point (the most recent kill).

>> Kill a line, move around, kill another line.
Then do C-y to get back the second killed line.
Then do M-y and it will be replaced by the first killed line.
Do more M-y's and see what you get. Keep doing them until
the second kill line comes back, and then a few more.
If you like, you can try giving M-y positive and negative
arguments.

UNDO

Any time you make a change to the text and wish you had not done so, you can undo the change (return the text to its previous state) with the undo command, C-x u. Normally, C-x u undoes one command's worth of changes; if you repeat the C-x u several times in a row, each time undoes one more command. There are two exceptions: commands that made no change (just moved the cursor) do not count, and self-inserting characters are often lumped together in groups of up to 20. This is to reduce the number of C-x u's you have to type.

>> Kill this line with C-k, then type C-x u and it should reappear.

C-_ is another command for undoing; it is just the same as C-x u but easier to type several times in a row. The problem with C-_ is that on some keyboards it is not obvious how to type it. That is why C-x u is provided as well. On some DEC terminals, you can type C-_ by typing / while holding down CTRL. Illogical, but what can you expect from DEC?

Giving a numeric argument to C-_ or C-x u is equivalent to repeating it as many times as the argument says.

FILES

In order to make the text you edit permanent, you must put it in a file. Otherwise, it will go away when your invocation of Emacs goes away. You put your editing in a file by "finding" the file. What finding means is that you see the contents of the file in your Emacs; and, loosely speaking, what you are editing is the file itself. However, the changes still don't become permanent until you "save" the file. This is so you can have control to avoid leaving a half-changed file around when you don't want to. Even then, Emacs leaves the original file under a changed name in case your changes turn out to be a mistake.

If you look near the bottom of the screen you will see a line that begins and ends with dashes, and contains the string "Emacs: TUTORIAL". Your copy of the Emacs tutorial is called "TUTORIAL". Whatever file you find, that file's name will appear in that precise spot.

The commands for finding and saving files are unlike the other commands you have learned in that they consist of two characters.

They both start with the character Control-x. There is a whole series of commands that start with Control-x; many of them have to do with files, buffers, and related things, and all of them consist of Control-x followed by some other character.

Another thing about the command for finding a file is that you have to say what file name you want. We say the command "reads an argument from the terminal" (in this case, the argument is the name of the file). After you type the command

C-x C-f Find a file

Emacs asks you to type the file name. It echoes on the bottom line of the screen. You are using the minibuffer now! this is what the minibuffer is for. When you type <Return> to end the file name, the minibuffer is no longer needed, so it disappears.

>> Type C-x C-f, then type C-g. This cancels the minibuffer, and also cancels the C-x C-f command that was using the minibuffer. So you do not find any file.

In a little while the file contents appear on the screen. You can edit the contents. When you wish to make the changes permanent, issue the command

C-x C-s Save the file

The contents of Emacs are written into the file. The first time you do this, the original file is renamed to a new name so that it is not lost. The new name is made by appending "~" to the end of the original file's name.

When saving is finished, Emacs prints the name of the file written. You should save fairly often, so that you will not lose very much work if the system should crash.

>> Type C-x C-s, saving your copy of the tutorial.
This should print "Wrote .../TUTORIAL" at the bottom of the screen.
On VMS it will print "Wrote ...[...]TUTORIAL."

To make a new file, just find it "as if" it already existed. Then start typing in the text. When you ask to "save" the file, Emacs will really create the file with the text that you have inserted. From then on, you can consider yourself to be editing an already existing file.

BUFFERS

If you find a second file with C-x C-f, the first file remains inside Emacs. You can switch back to it by finding it again with C-x C-f. This way you can get quite a number of files inside Emacs.

The object inside Emacs which holds the text read from one file is called a "buffer." Finding a file makes a new buffer inside Emacs.

To see a list of the buffers that exist in Emacs, type

C-x C-b List buffers

>> Try C-x C-b now.

See how each buffer has a name, and it may also have a file name for the file whose contents it holds. Some buffers do not correspond to files. For example, the buffer named "*Buffer List*" does not have any file. It is the buffer which contains the buffer list that was made by C-x C-b. ANY text you see in an Emacs window has to be in some buffer.

>> Type C-x 1 to get rid of the buffer list.

If you make changes to the text of one file, then find another file, this does not save the first file. Its changes remain inside Emacs, in that file's buffer. The creation or editing of the second file's buffer has no effect on the first file's buffer. This is very useful, but it also means that you need a convenient way to save the first file's buffer. It would be a nuisance to have to switch back to it with C-x C-f in order to save it with C-x C-s. So we have

C-x s Save some buffers

C-x s goes through the list of all the buffers you have and finds the ones that contain files you have changed. For each such buffer, C-x s asks you whether to save it.

EXTENDING THE COMMAND SET

There are many, many more Emacs commands than could possibly be put on all the control and meta characters. Emacs gets around this with the X (eXtend) command. This comes in two flavors:

C-x Character eXtend. Followed by one character.
M-x Named command eXtend. Followed by a long name.

These are commands that are generally useful but used less than the commands you have already learned about. You have already seen two of them: the file commands C-x C-f to Find and C-x C-s to Save. Another example is the command to tell Emacs that you'd like to stop editing and get rid of Emacs. The command to do this is C-x C-c. (Don't worry; it offers to save each changed file before it kills the Emacs.)

C-z is the usual way to exit Emacs, because it is always better not to kill the Emacs if you are going to do any more editing. On systems which allow it, C-z exits from Emacs to the shell but does not destroy the Emacs; if you use the C shell, you can resume Emacs with the 'fg' command (or, more generally, with '%emacs', which works even if your most recent job was some other). On systems where suspending is not possible, C-z creates a subshell running under Emacs to give you the chance to run other programs and return to Emacs afterward, but it

does not truly "exit" from Emacs. In this case, you must ask an expert on your computer how to get back to Emacs from the subshell.

You would use C-x C-c if you were about to log out. You would also use it to exit an Emacs invoked under mail handling programs and other random utilities, since they may not believe you have really finished using the Emacs if it continues to exist.

There are many C-x commands. The ones you know are:

C-x C-f	Find file.
C-x C-s	Save file.
C-x C-b	List buffers.
C-x C-c	Quit Emacs.
C-x u	Undo.

Named eXtended commands are commands which are used even less frequently, or commands which are used only in certain modes. These commands are usually called "functions". An example is the function replace-string, which globally replaces one string with another. When you type M-x, Emacs prompts you at the bottom of the screen with M-x and you should type the name of the function you wish to call; in this case, "replace-string". Just type "repl s<TAB>" and Emacs will complete the name. End the command name with <Return>.
Then type the two "arguments"--the string to be replaced, and the string to replace it with--each one ended with a Return.

>> Move the cursor to the blank line two lines below this one.
Then type M-x repl s<Return>changed<Return>altered<Return>.

Notice how this line has changed: you've replaced the word c-h-a-n-g-e-d with "altered" wherever it occured after the cursor.

MODE LINE

If Emacs sees that you are typing commands slowly it shows them to you at the bottom of the screen in an area called the "echo area." The echo area contains the bottom line of the screen. The line immediately above it is called the MODE LINE. The mode line says something like

```
--**--Emacs: TUTORIAL                (Fundamental)----58%-------------
```

This is a very useful "information" line.

You already know what the filename means--it is the file you have found. What the --NN%-- means is that NN percent of the file is above the top of the screen. If the top of the file is on the screen, it will say --TOP-- instead of --00%--. If the bottom of the file is on the screen, it will say --BOT--. If you are looking at a file so small it all fits on the screen, it says --ALL--.

The stars near the front mean that you have made changes to the text. Right after you visit or save a file, there are no stars, just dashes.

The part of the mode line inside the parentheses is to tell you what modes you are in. The default mode is Fundamental which is what you are in now. It is an example of a "major mode". There are several major modes in Emacs for editing different languages and text, such as Lisp mode, Text mode, etc. At any time one and only one major mode is active, and its name can always be found in the mode line just where "Fundamental" is now. Each major mode makes a few commands behave differently. For example, there are commands for creating comments in a program, and since each programming language has a different idea of what a comment should look like, each major mode has to insert comments differently. Each major mode is the name of an extended command, which is how you get into the mode. For example, M-X fundamental-mode is how to get into Fundamental mode.

If you are going to be editing English text, such as this file, you should probably use Text Mode.
>> Type M-x text-mode<Return>.

Don't worry, none of the commands you have learned changes Emacs in any great way. But you can now observe that periods are no longer part of words when you do M-f or M-b! Major modes are usually like that: commands don't change into completely unrelated things, but they work a little bit differently.

To get documentation on your current major mode, type C-h m.

>> Use C-u C-v once or more to bring this line near the top of screen.
>> Type C-h m, to see how Text mode differs from Fundamental mode.
>> Type C-x 1 to remove the documentation from the screen.

Major modes are called major because there are also minor modes. They are called minor because they aren't alternatives to the major modes, just minor modifications of them. Each minor mode can be turned on or off by itself, regardless of what major mode you are in, and regardless of the other minor modes. So you can use no minor modes, or one minor mode, or any combination of several minor modes.

One minor mode which is very useful, especially for editing English text, is Auto Fill mode. When this mode is on, Emacs breaks the line in between words automatically whenever the line gets too long. You can turn this mode on by doing M-x auto-fill-mode<Return>. When the mode is on, you can turn it off by doing M-x auto-fill-mode<Return>. If the mode is off, this function turns it on, and if the mode is on, this function turns it off. This is called "toggling".

>> Type M-x auto-fill-mode<Return> now. Then insert a line of "asdf " over again until you see it divide into two lines. You must put in spaces between them because Auto Fill breaks lines only at spaces.

The margin is usually set at 70 characters, but you can change it with the C-x f command. You should give the margin setting you want as a numeric argument.

>> Type C-x f with an argument of 20. (C-u 2 0 C-x f).
Then type in some text and see Emacs fill lines of 20

characters with it. Then set the margin back to 70 using
C-x f again.

If you makes changes in the middle of a paragraph, Auto Fill mode
does not re-fill it for you.
To re-fill the paragraph, type M-q (Meta-q) with the cursor inside
that paragraph.

>> Move the cursor into the previous paragraph and type M-q.

SEARCHING

Emacs can do searches for strings (these are groups of contiguous characters or words) either forward through the file or backward through it. To search for the string means that you are trying to locate it somewhere in the file and have Emacs show you where the occurrences of the string exist. This type of search is somewhat different from what you may be familiar with. It is a search that is performed as you type in the thing to search for. The command to initiate a search is C-s for forward search, and C-r for reverse search. BUT WAIT! Don't do them now. When you type C-s you'll notice that the string "I-search" appears as a prompt in the echo area. This tells you that Emacs is in what is called an incremental search waiting for you to type the thing that you want to search for. <ESC> terminates a search.

>> Now type C-s to start a search. SLOWLY, one letter at a time,
type the word 'cursor', pausing after you type each
character to notice what happens to the cursor.
>> Type C-s to find the next occurrence of "cursor".
>> Now type <Rubout> four times and see how the cursor moves.
>> Type <ESC> to terminate the search.

Did you see what happened? Emacs, in an incremental search, tries to go to the occurrence of the string that you've typed out so far. To go to the next occurrence of 'cursor' just type C-s again. If no such occurrence exists Emacs beeps and tells you that it is a failing search. C-g would also terminate the search.

If you are in the middle of an incremental search and type <Rubout>, you'll notice that the last character in the search string is erased and the search backs up to the last place of the search. For instance, suppose you currently have typed 'cu' and you see that your cursor is at the first occurrence of 'cu'. If you now type <Rubout>, the 'u' on the search line is erased and you'll be repositioned in the text to the occurrence of 'c' where the search took you before you typed the 'u'. This provides a useful means for backing up while you are searching.

If you are in the middle of a search and happen to type a control character (other than a C-s or C-r, which tell Emacs to search for the next occurrence of the string), the search is terminated.

The C-s starts a search that looks for any occurrence of the search string AFTER the current cursor position. But what if you want to

search for something earlier in the text? To do this, type C-r for Reverse search. Everything that applies to C-s applies to C-r except that the direction of the search is reversed.

RECURSIVE EDITING LEVELS

Sometimes you will get into what is called a "recursive editing level". This is indicated by square brackets in the mode line, surrounding the parentheses around the major mode name. For example, you might see [(Fundamental)] instead of (Fundamental).

To get out of the recursive editing level, type
M-x top-level<Return>.

>> Try that now; it should display "Back to top level"
at the bottom of the screen.

In fact, you were ALREADY at top level (not inside a recursive editing level) if you have obeyed instructions. M-x top-level does not care; it gets out of any number of recursive editing levels, perhaps zero, to get back to top level.

You can't use C-g to get out of a recursive editing level because C-g is used for discarding numeric arguments and partially typed commands WITHIN the recursive editing level.

GETTING MORE HELP

In this tutorial we have tried to supply just enough information to get you started using Emacs. There is so much available in Emacs that it would be impossible to explain it all here. However, you may want to learn more about Emacs since it has numerous desirable features that you don't know about yet. Emacs has a great deal of internal documentation. All of these commands can be accessed through the character Control-h, which we call "the Help character" because of the function it serves.

To use the HELP features, type the C-h character, and then a character saying what kind of help you want. If you are REALLY lost, type C-h ? and Emacs will tell you what kinds of help it can give. If you have typed C-h and decide you don't want any help, just type C-G to cancel it.

The most basic HELP feature is C-h c. Type C-h, a c, and a command character or sequence, and Emacs displays a very brief description of the command.

>> Type C-h c Control-p.
The message should be something like

C-p runs the command previous-line

This tells you the "name of the function". That is important in writing Lisp code to extend Emacs; it also is enough to remind you of what the command does if you have seen it before but did not remember.

Multi-character commands such as C-x C-s and (if you have no META or EDIT key) <ESC>v are also allowed after C-h c.

To get more information on the command, use C-h k instead of C-h c.

>> Type C-h k Control-p.

This displays the documentation of the function, as well as its name, in an Emacs window. When you are finished reading the output, type C-x 1 to get rid of the help text. You do not have to do this right away. You can do some editing based on the help text before you type C-x 1.

Here are some other useful C-h options:

C-h f Describe a function. You type in the name of the function.

>> Try typing C-h f previous-line<Return>.
This prints all the information Emacs has about the function which implements the C-P command.

C-h a Command Apropos. Type in a keyword and Emacs will list all the commands whose names contain that keyword. These commands can all be invoked with Meta-x. For some commands, Command Apropos will also list a one or two character sequence which has the same effect.

>> Type C-h a file<Return>. You will see a list of all M-x commands with "file" in their names. You will also see commands like C-x C-f and C-x C-w, listed beside the command names find-file and write-file.

CONCLUSION

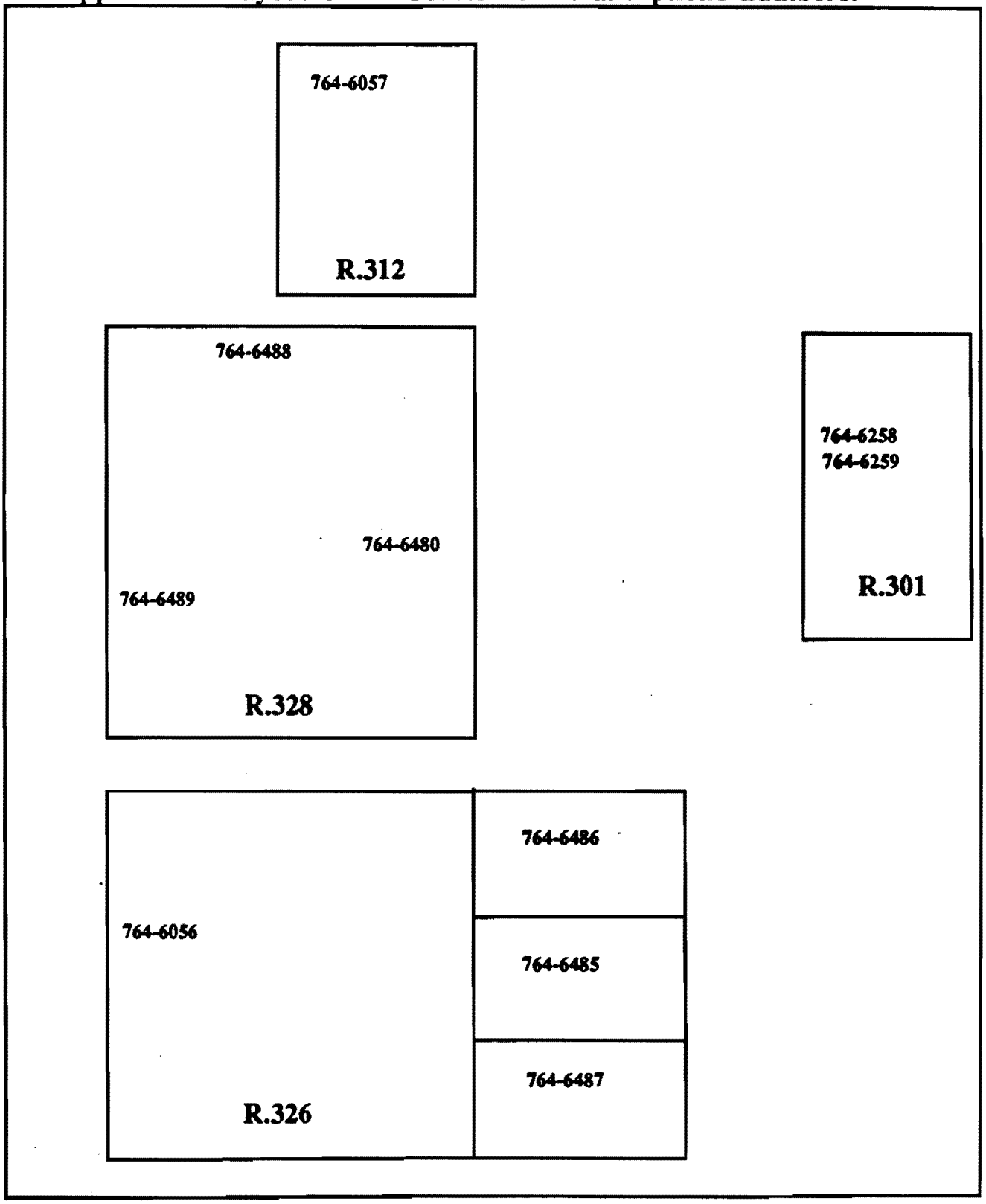
Remember, to exit Emacs permanently use C-x C-c. To exit to a shell temporarily, so that you can come back in, use C-z.

This tutorial is meant to be understandable to all new users, so if you found something unclear, don't sit and blame yourself - complain!

COPYING

The conditions for copying Emacs itself are slightly different
but in the same spirit. Please read the file COPYING and then
do give copies of GNU Emacs to your friends.
Help stamp out ownership of software by using, writing,
and sharing free software!

A5. Approximate layout of AI Center rooms and phone numbers.



A6. Where to call for help.

Here is the list of persons responsible for particular computers:

- ACS computing services:
  - UNIX machines - Duane King , ext.2486
  - VMS machines - Tracy Holt, ext.2596
  - GNUEmacs - Darren Stalder, ext. 3569
  - Networking - Byron Peters, ext. 2598
  - User services - ext. 3816
  - ACS computers status - ext. 3384
- MAC II - Jerzy Bala and Jangping Zhang, Rm.312, 764-6057
- SUN - Pawel A. Stefanski, Rm.312, 764-6057
- VAX II - Ken Kaufman, Rm.312, 764-6057
- General hardware - Jerzy Bala, Rm.312, 764-6057.

In case something is wrong (either with hardware or software), please do the following:

(1) enter the short, descriptive message to the Log Book,
(2) call the person resonsible for particular machine/software.

NAME

echo - echo arguments

SYNOPSIS

echo [options] ...

DESCRIPTION

Echo writes its arguments separated by blanks and terminated by a newline on the standard output. Options to filter and redirect the output are as follows:

-2 generate rhyming couplets from keywords

-3 generate Haiku verse from keywords

-5 generate limerick from keywords

-a convert ASCII to ASCII

-A disambiguate sentence structure

-b generate bureaucratese equivalent (see **-x**)

-B issue equivalent C code with bugs fixed

-c simplify/calculate arithmetic expression(s)

-C remove copyright notice(s)

-d define new echo switch map

-D delete all ownership information from system files

-e evaluate Lisp expression(s) (Common Lisp standard, as defined in CLtL)

-E *machine-name*
port the EMERALD - Integrated Machine Learning system to a given machine (Sun, VAX and Apollo only)

-f read input from file

-F transliterate to French

-g generate pseudo-revolutionary marxist catch-phrases

-G prepend GNU manifesto

-H *room-number*
start HERO-2000 and ask him for a cup of coffee - give room number as an argument (Sun only)

-h halt system (reboot suppressed on Suns, Apollos, and VAXen, not supported on NOS-2)

-i emulate IBM OS/VU (recursive universes not supported)

-I emulate IBM VTOS 3.7.6 (chronosynclastic infundibulae supported with restrictions documented in IBM VTOS Reference Manual rev. 3.2.6)

-J *subject*
generate a joke on a given subject

-j justify text (see **-b** option)

-k disconnect/connect Kinetics box to the SUN network

-K delete privileged accounts

-l generate legalese equivalent

-L load echo modules

-M generate a chapter in the Machine Learning book IV.

-m correct syntax and semantics of any Machine Learning article (text with LF/CR only)

-N send output to all reachable networks (usable with **-J**, **-K**, **-h** options)

-n do not add newline to the output

-o generate obscene text

-O clean up dirty language

-p decrypt and print /etc/passwd

-P port echo to all reachable networks

-P1 hide LaserWriter printer from all users

-q query standard input for arguments

-r read alternate ".echo" file on start up

-R change root password to "RMS"

-s suspend operating system during output (Sun and VAX BSD 4.2 only)

-S translate to swahili

-T emulate TCP/IP handler

-t issue troff output

-u issue Unix philosophy essay

-v generate reverberating echo

-V print debugging information

-x decrypt DES format messages (NSA secret algorithm CX 3.8, not distributed outside continental US)

Echo is useful for producing diagnostics in shell programs and for writing constant data on pipes. To send diagnostics to the standard error file, do `echo ... 1>&2'.

AUTHOR

Richard M. Stallman
Pawel A. Stefanski