The SEST OS-9 quick guide

Last Update: 18th of October 2000

Mikael Lerner

Contents

Chapter 1

OS-9 basic commands

1.1 General information about OS-9

1.1.1 Control keys

The following control keys are useful:

`<Ctrl>-a`	recall last input line
`<Ctrl>-d`	redisplay current input line
`<Ctrl>-x`	delete current input line
`<Ctrl>-h`	backspace
`<Ctrl>-s`	halt output
`<Ctrl>-q`	resume output
`<Ctrl>-w`	halt output, resume by hitting any key
`<Escape>`	EOF (including logout)
`<Ctrl>-c`	interrupt program
`<Ctrl>-e`	abort program

1.1.2 Directories

The directory structure is similar to that in UNIX. Addressing is done in similar ways both for full and relative addresses. However, OS-9 defines both a current working directory and a current directory with executables. You can refer to the current execution directory by adding the option "`-x`" to many commands.

1.1.3 File names

File names must contain between 1 and 28 characters and can only consist of any of the following symbols: `A`–`Z`, `a`–`z`, `0`–`9`, period (`.`), underscore (_) and dollar sign (`$`). At least one character must be a letter or a digit. Unlike UNIX, OS-9 does not consider uppercase and lowercase to be different letters, so it is not possible to have two files called `TEST` and `test`. However, a good OS-9 convention is to use uppercase letters for directories and lowercase letters for files.

1.1.4 ASCII files

The line feed contra carriage return convention for ASCII files is different for OS-9 compared to UNIX and LINUX. This means that text files can't be transferred to a UNIX system, edited there and then put back to an OS-9 computer.

1.1.5 File permissions

Although OS-9 provides both user-IDs and group-IDs, OS-9 basically lacks one level of privacy, since there are no differences in privileges between users belonging to the same group. Thus, the "user" part is missing from the "user"–"group"–"others" concept available in UNIX.

Ownership and permissions can be checked with `dir -e`. Ownership is given as "`group-ID.user-ID`" and the permission attributes are listed as an 8-character string that read from right to left shows read, write and executable permissions for user/group; read, write and executable permissions for others; a flag indicating that the file can only be accessed once at a time; and finally the directory flag. When changing permissions, the different flags are referred to as "`r`", "`w`" and "`e`" for user/group; "`pr`", "`pw`" and "`pe`" for others; and an "`s`" for the single user flag. To remove a certain permission, an "`n`" is added in front of the flag.

1.1.6 Program control

The following table lists all special characters that are used on the command line to control program execution.

Wildcards	?	matches any single character (`list file?.txt`)
	*	matches any number of characters (`list a*`)
Modifiers	#	memory allocation in kB (`prog #32`)
	^	process priority (`prog ^128`)
	&	background job (`prog &`)
Redirection	<	redirect input (`prog < infile`)
	>	redirect output (`prog > outfile`)
	>>	redirect error output (`prog >> errorfile`)
	*-	overwrite file (`prog >- outfile)`
	*+	append to file (`prog >+ outfile)`
	!	piping (`prog1 ! prog2)`
Separators	;	sequential execution (`prog1 ; prog2`)
	&	parallel execution (`prog1 & prog2`)
Grouping	()	group commands (`(list file1 ; list file2) ! pr`)

1.2 Basic commands

This chapter contains a list of some basic OS-9 commands.

`attr`	change file permissions
`build`	create a short text file
`chd`	change current directory
`chx`	change current directory for executables
`cmp`	compare files binary
`code`	print hexadecimal values to input characters
`compress`	compress a text file
`copy`	copy a file
`count`	count lines (words, characters) in a file
`date`	display date and time
`del`	delete a file
`deldir`	delete a directory
`devs`	print system device table
`dir`	list the contents in a directory
`dsave`	copy a directory structure
`dump`	print a non-ASCII file
`echo`	print text string
`ex`	execute a program replacing invoking program
`expand`	uncompress a compressed file
`free`	display disk usage
`grep`	search file for a string
`help`	display help about a command
`kill`	kill a process
`link`	link a previously loaded module into memory

`list`	list a file
`load`	load a module into memory
`login`	login a user
`logout`	terminate the current shell
`makdir`	create a directory
`make`	compile programs
`mdir`	list the module directory
`merge`	merge files
`mfree`	display memory usage
`pd`	print the current working directory
`pr`	list a file in a formatted way
`printenv`	print out all environment variables
`procs`	list processes
`profile`	source a shell script
`qsort`	sort
`rename`	rename a file or a directory
`reset`	reboot the computer
`set`	set shell parameters
`setenv`	set environment variables
`setime`	set date and time (*non-Y2K version*)
`setimey2k`	set date and time
`setpr`	set process priority
`shell`	start a new shell
`sleep`	sleep
`tee`	split the output to multiple destinations
`tmode`	display or change terminal parameters
`touch`	update the modification time of a file
`tr`	transliterate characters
`umacs`	create and edit files a la emacs
`unlink`	unlink a memory module
`unsetenv`	remove an environment variable
`w`	wait until one child process has terminated
`wait`	wait until all child processes have terminated

1.3 A UNIX to OS-9 command conversion table

This chapter contains a list of the OS-9 equivalents to some commonly used commands in UNIX.

`bdf`	`free`
`cat >` *file*	`build` *file*

`cat` *file*	`list` *file*
`cat` *file1 file2*	`merge` *file1 file2*
`cd`	`chx`
`cd` *directory*	`chd` *directory*
`chmod`	`attr`
`cmp`	`cmp`
`compress`	`compress -n`
`cp`	`copy` or `copy -r` (for overwrite)
`cp -r`	`dsave`
`csh`	`shell`
`date`	`date -m`
`date` *date*	`setimey2k`
`df`	`free`
`echo`	`echo`
`emacs`	`umacs`
`exec`	`ex`
`exit`	`logout`
`grep`	`grep`
`gunzip`	`expand`
`gzip`	`compress -n`
`kill`	`kill`
`login`	`login`
`logout`	`logout`
`ls`	`dir`
`ls -1`	`dir -u`
`ls -a`	`dir -a`
`ls -l`	`dir -e`
`ls -R`	`dir -r`
`make`	`make`
`man` *command*	`help` *command*
`mkdir`	`makdir`
`more` *file*	`list` *file*
`mv`	`rename`
`nice`	`setpr`
`ps -f`	`procs`
`ps -ef`	`procs -e`
`pwd`	`pd`
`rm`	`del`
`rm -i`	`del -p`
`rm -r`	`deldir -q`
`rmdir`	`deldir`
`set` *parameter*	`set` *parameter*
`setenv`	`printenv`

`setenv` *parameter value*	`setenv` *parameter value*
`shutdown -r`	`reset`
`sleep`	`sleep -s`
`sort`	`qsort`
`source`	`profile`
`stty`	`tmode`
`tcsh`	`shell`
`tee`	`tee`
`touch`	`touch`
`tr`	`tr`
`uncompress`	`expand`
`unsetenv`	`unsetenv`
`wait`	`wait`
`wc`	`count -clw`
`wc -c`	`count -c`
`wc -l`	`count`
`wc -w`	`count -w`

1.4 OS-9 command descriptions

This chapter describes each OS-9 command with all their options. Anything given in square brackets ("`[]`") is optional and everything that is underlined can be repeated.

1.4.1 attr

Syntax:	`attr [`*opts*`]` <u>*path* `[`*opts*`]` *permissions*</u>	
Function:	display or change file attributes	
Attributes:	`d s pe pw pr e w r`	
	'`-`' turns attribute on, '`-n`' turns attribute off	
Options:	`-a`	do not print attributes after changes
	`-x`	directory to search is execution directory
	`-z`	get list of file names from standard input
	`-z=`*path*	get list of file names from *path*

1.4.2 build

Syntax:	`build` *path*
Function:	build a short text file

1.4.3 chd

Syntax:	`chd` [*directory*]
Function:	change current directory

1.4.4 chx

Syntax:	`chd` *executable-directory*
Function:	change current directory for executables

1.4.5 cmp

Syntax:	`cmp` [*opts*] *path1* *path2* [*opts*]	
Function:	compare two files	
Options:	`-b=`*size*	buffer size
	`-s`	silent mode/aborts with error on first mismatch
	`-x`	defaults to current execution directory

1.4.6 code

Syntax:	`code`
Function:	return hex value of terminal key

1.4.7 compress

Syntax:	`compress` [*opts*] *path* [*opts*]	
Function:	compress an ascii file	
Options:	`-d`	delete original file
	`-n`	create output file
	`-z`	get list of file names from standard input
	`-z=`*path*	get list of file names from *path*
Comment:	should only be used for text files	

1.4.8 copy

Syntax:	`copy` [*opts*] *srcpath* [*dstpath*] [*opts*]
Function:	copy data from one path to another

Options:	`-a`	abort on error
	`-b=`*size*	buffer size
	`-f`	rewrite destination files with no write permission
	`-p`	don't print file names copied (with -w option only)
	`-r`	rewrite destination
	`-v`	verify integrity of files written
	`-w=`*dir-name*	wild card copy to *dir-name*
	`-x`	look in execution directory for source
	`-z`	get list of file names from standard input
	`-z=`*path*	get list of file names from *path*

1.4.9 count

Syntax: `count [`*opt*`]` *path* `[`*opts*`]`
Function: count characters, lines, and words in a file

Options:	`-b`	count characters with breakdown
	`-c`	count characters
	`-l`	count lines (default)
	`-w`	count words
	`-z`	get list of filenames from standard input
	`-z=`*path*	get list of filenames from *path*

1.4.10 date

Syntax: `date [`*opts*`]`
Function: display system date and time

Options:	`-j`	print day, seconds past midnight in julian time
	`-m`	print hour:minute:sec in military format

1.4.11 del

Syntax: `del [`*opts*`]` *file* `[`*opts*`]`
Function: delete files

Options:	`-e`	erases the disk space that the file occupied
	`-f`	delete files with no write permission
	`-p`	show file name and ask before deleting
	`-x`	delete files from execution directory
	`-z`	get list of file names from standard input
	`-z=`*path*	get list of file names from *path*

1.4.12 deldir

Syntax: `deldir [`*opts*`]` *dir* `[`*opts*`]`
Function: delete a directory

Options:	`-q`	delete directories without asking questions
	`-f`	delete files with no write permission
	`-z`	get list of directory names from standard input
	`-z=`*path*	get list of directory names from *path*

1.4.13 devs

Syntax: `devs`
Function: print system device table

1.4.14 dir

Syntax: `dir [`*opts*`]` *dir names* `[`*opts*`]`
Function: display directory contents

Options:	`-a`	show all files
	`-d`	show directories with a slash
	`-e`	extended dir listing
	`-n`	treat dirs like files
	`-r`	recursive dir listings
	`-r=`*num*	recursive dir listing to depth *num*
	`-s`	unsorted dir listing
	`-u`	unformatted listing
	`-x`	directory is execution dir
	`-z`	get list of dir names from standard input
	`-z=`*path*	get list of dir names from *path*

1.4.15 dsave

Syntax: `dsave [`*opts*`] [`*to-path*`] [`*opts*`]`
Function: copy directory structure

Options:	`-a`	don't copy files beginning with '.'
	`-b[=]`*n*	copy memory size
	`-d`	only copy files with newer dates
	`-d=`*date*	copy files newer than *date*
	`-e`	execute output
	`-f`	use force option on copy
	`-i`	indent directory levels
	`-l`	current directory only
	`-m`	omit makdirs
	`-n`	don't load copy
	`-o`	OS9Gen on boot file
	`-o=`*n*	OS9Gen with given bootfile name *n*
	`-r`	use rewrite option on copy
	`-s`	skip file on error; no 'continue?' prompt
	`-v`	verify files with cmp

1.4.16 dump

Syntax: `dump [`*opts*`]` *path* `[`*starting-offset*`] [`*opts*`]`
Function: formatted display of contents of a device

Options:	`-c`	don't compress duplicate lines
	`-m`	dump from a memory resident module
	`-s`	interpret starting offset as sector number
	`-x`	path implies execution directory

1.4.17 echo

Syntax: `echo [`*opts*`] [`*text*`] [`*opts*`]`
Function: echo text to output path and convert hex to ASCII

Options:	`-n`	separate text with carriage returns
	`-r`	don't send out a return on exit
	`-z`	get text from standard input
	`-z=`*file*	get text from *file*

1.4.18 ex

Syntax: `ex` *path* `[`*arglist*`]`
Function: execute a program replacing the calling process

1.4.19 expand

Syntax: `expand [`*opts*`]` <u>*path*</u> `[`*opts*`]`
Function: expand a compressed file

Options:	`-d`	delete original file
	`-n`	create output file
	`-z`	get list of file names from standard input
	`-z=`*path*	get list of file names from *path*

1.4.20 free

Syntax:	`free [`*opts*`]` *device* `[`*opts*`]`
Function:	report free space on disk
Options:	`-b=`*size* buffer size

1.4.21 grep

Syntax:	`grep [`*opts*`] [`*expression*`] [`*path*`] [`*opts*`]`	
Function:	searches input module for lines matching expression	
Options:	`-c`	count match lines
	`-e=expr`	same as simple expression
	`-f=file`	read 'file' for expression(s)
	`-l`	print only filenames with matches
	`-n`	number each output line (file relative)
	`-s`	silent mode
	`-v`	invert sense of compare (list non-matches)
	`-z`	get list of file names from standard input
	`-z=`*path*	get list of file names from *path*
Specials:	`~`	= beginning of line
	`$`	= end of line
	`.`	= any character
	`*`	= any number of the preceding character
	`[]`	= group, e.g. `[a-z]` or `[m-pa-f]`
	`[~]`	= negating a group, e.g. `[~0-9]`

1.4.22 help

Syntax:	`help [`*file*`] [`*file*`]`
Function:	provide Help for OS-9/68000 Utilities
Comment:	to get help for a specific utility type: "`help` *filename*".

1.4.23 kill

Syntax:	`kill` *proc-ID*
Function:	kill a process
Comment:	`kill 0` kills all processes owned by the user

1.4.24 link

Syntax:	`link` [*opts*] *modname* [*opts*]	
Function:	link a module in memory	
Options:	`-z`	get list of module names from standard input.
	`-z=`*path*	get list of module names from *path*

1.4.25 list

Syntax:	`list` [*opts*] *path* [*opts*]	
Function:	list a file	
Options:	`-z`	get list of file names from standard input
	`-z=`*path*	get list of file names from *path*

1.4.26 load

Syntax:	`load` [*opts*] *module* [*opts*]	
Function:	load a module into memory	
Options:	`-d`	load file from data directory
	`-l`	print pathlist of file loaded
	`-z`	get list of file names from standard input
	`-z=`*path*	get list of file names from *path*

1.4.27 login

Syntax: `login` [*opts*] *name* [,] *password*
Function: to provide login security

1.4.28 logout

Syntax: `logout`
Function: terminate the current shell

1.4.29 makdir

Syntax:	`makdir` [*opts*] *dir name* [*opts*]	
Function:	create a directory	
Options:	`-x`	create directory in execution directory
	`-z`	get list of dir names from standard input
	`-z=`*file*	get list of dir names from *file*

1.4.30 make

Syntax:	`make` [*-opts*] [*target-file*] [*macros*]	
Function:	keep track of modules for a file	
Options:	`-b`	don't use built-in rules
	`-bo`	don't use built-in rules for object files
	`-d`	debug mode, print out the file dates in makefile
	`-dd`	double debug mode, very verbose
	`-f[=]`*xxx*	use *xxx* as the makefile (default: makefile)
	`-f-`	reads the makefile from stdin
	`-i`	ignore errors on commands and keep going
	`-n`	don't execute commands, just print them out
	`-s`	silent mode, execute commands without echoing them
	`-t`	update the dates without executing the commands
	`-u`	do the make whether it needs it or not
	`-x`	use the cross compiler
	`-z[=`*path*`]`	get list of files to make from stdin or path

1.4.31 mdir

Syntax:	`mdir` [*opts*] [*mod-names*] [*opts*]	
Function:	display module directory	
Options:	`-a`	print language instead of type
	`-e`	print extended directory listing
	`-t=`*type*	list modules only of type *type*
	`-u`	print unformatted listing

1.4.32 merge

Syntax:	`merge` [*opts*] *path* [*opts*]	
Function:	merge file to standard output	
Options:	`-b=`*size*	buffer size
	`-x`	look in execution directory for files
	`-z`	get list of file names from standard input
	`-z=`*path*	get list of file names from *path*

1.4.33 mfree

Syntax:	`mfree` [*opt*]	
Function:	display system memory information	
Options:	`-e`	extended free memory description

1.4.34 pd

Syntax:	`pd` [*opt*]
Function:	print path to data or execution directory
Options:	`-x` display execution dir path

1.4.35 pr

Syntax:	`pr` [*opts*] *path* [*opts*]	
Function:	display file in specified format	
Options:	`-c=char`	pick column separation character
	`-d=num`	set depth of page in lines (default 66)
	`-f`	do not use form feed
	`-h=num`	set number of lines after header
	`-k=num`	set number of columns
	`-l=num`	set left margin
	`-m`	print all open files one per column
	`-n=num`	set starting line number and incr
	`-o`	truncate lines longer than rmarg
	`-p=num`	set number of lines printed per page
	`-r=num`	set right margin
	`-t`	do not print title
	`-u=title`	use specified title (48 characters maximum)
	`-x=num`	set starting page number
	`-z`[=*path*]	read file names from stdin or *path* if given

1.4.36 printenv

Syntax:	`printenv`
Function:	print out all environment variables

1.4.37 procs

Syntax:	`procs` [*opts*]	
Function:	display user processes	
Options:	`-a`	display alternate data
	`-b`	display both normal and alternate data
	`-e`	display every valid process

1.4.38 profile

Syntax:	`profile` *path*
Function:	read commands from a file and return

1.4.39 qsort

Syntax:	`qsort` [*opts*] *file* [*opts*]	
Function:	in-memory quick sort	
Options:	`-c=`*char*	specify field separation character
	`-f=`*num*	specify field to sort on
	`-z`	get list of file names from standard input
	`-z=`*path*	get list of file names from *path*

1.4.40 rename

Syntax:	`rename` [*opts*] *path* *name* [*opts*]	
Function:	rename a file or directory	
Options:	`-x`	path starts from execution dir

1.4.41 reset

Syntax:	`reset`
Function:	reboots the computer
Comment:	This is a special command created by Michel

1.4.42 set

Syntax:	`set` *opts*	
Function:	set shell options	
Options:	`e=`*file*	print error messages from file *file*
	`ne`	print no error messages (default)
	`l`	must log off system with `logout`
	`nl`	can log off system with `<Escape>`
	`p`	display prompt
	`np`	do not display prompt
	`p=`*string*	set current shell prompt to *string*
	`t`	echo input lines
	`nt`	do not echo input lines (default)
	`v`	verbose mode
	`nv`	do not use verbose mode
	`x`	abort process upon error (default)
	`nx`	do not abort on error
Comment:	"`set`" is not needed if the option is preceded by a "`-`"	

1.4.43 setenv

Syntax:	`setenv` *parameter* *value*
Function:	set the environment variables

1.4.44 setime

Syntax:	`setime [`*opt*`] [yy mm dd hh mm ss] [am/pm]`
Function:	set system date and time
Options:	`-d` don't display time
	`-s` setime for battery backed-up clocks
Comment:	`setimey2k` should be used instead of this command

1.4.45 setimey2k

Syntax:	`setimey2k yyyy mm dd [hh mm ss] [am/pm]`
Function:	set system date and time
Comment:	This is a special command created by Michel

1.4.46 setpr

Syntax:	`setpr` *proc-ID priority*
Function:	set the priority for a process

1.4.47 shell

Syntax:	`shell [[set]` *arglist*`]`	
Function:	start a new shell	
Options:	`-e[=`*path*`]`	print error explanations.
	`-ne`	don't print error explanations.
	`-p[=`*prompt*`]`	print prompt.
	`-np`	don't print prompt.
	`-t`	echo input lines.
	`-nt`	don't echo input lines.
	`-x`	exit on error.
	`-nx`	don't exit on error.
	`-l`	require "logout" to logout.
	`-nl`	*eof* on input will logout.
	`-v`	print attempts to execute command.
	`-nv`	don't print attempts to execute command.

1.4.48 sleep

Syntax:	`sleep [`*opts*`] [`*count*`] [`*opts*`]`
Function:	suspend process for ticks/seconds/until signalled
Options:	`-s` count represents seconds
Note:	count defaults to 0 if not specified

1.4.49 tee

Syntax: `tee` *path*

Function: copy input to multiple output paths

1.4.50 tmode

Syntax: `tmode [`*opts*`]` *parameter* `[`*opts*`]`

Function: display and/or change terminal operating characteristics

Options:

`-w=`#*path*	specify path to perform operations on
`[no]upc`	- upper case only
`[no]bsb`	- erase on backspace
`[no]bsl`	- backspace over line
`[no]echo`	- echo input to terminal
`[no]lf`	- auto line feed
`[no]pause`	- screen pause
`null=n`	- number of null chars after CR
`pag=n`	- display page length
`bsp=h`	- input backspace char (in hex)
`bse=h`	- output backspace char (in hex)
`del=h`	- input delete char (in hex)
`eor=h`	- end of record char (in hex)
`eof=h`	- end of file char (in hex)
`reprint=h`	- reprint line character (in hex)
`dup=h`	- duplicate last line char (in hex)
`psc=h`	- pause char (in hex)
`abort=h`	- abort char (in hex)
`quit=h`	- quit char (in hex)
`bell=h`	- bell output char (in hex)
`type=h`	- ACIA initialization value (in hex)
`par=s`	- set parity (s=odd,even,none)
`cs=n`	- character size in bits (n=8,7,6,5)
`stop=n`	- number of stop bits (n=1,1.5,2)
`baud=n`	- software settable baud rate (in decimal)
`xon=h`	- DC1 resume output char (in hex)
`xoff=h`	- DC2 suspend output char (in hex)
`tabc=h`	- tab char (in hex)
`tabs=d`	- number of spaces per tab
`normal`	- return to default values

1.4.51 touch

Syntax: `touch [`*opts*`]` *path* `[`*opts*`]`

Function: update the date of a file

Options:	`-c`	don't create files
	`-q`	don't quit on error
	`-x`	search execution directory
	`-z`	get list of file names from standard input
	`-z=`*path*	get list of file names from *path*

1.4.52 tr

Syntax: `tr [`*opts*`]` *str1* `[`*str2*`] [`*path1*`] [`*path2*`] [`*opts*`]`
Function: convert all occurrences of chars in *str1* to corresponding chars in *str2*

Options:	`-c`	invert (complement) sense of characters in str1
	`-d`	delete all characters matching str1 characters
	`-s`	squeeze repeated output chars from str2 into a single char
	`-v`	identical to 'c' option above
	`-z`	get list of file names from standard input
	`-z=`*path*	get list of file names from *path*

1.4.53 uMacs

Syntax: `uMacs [`*options*`] [`*files*`]`
Function: Advanced screen editor

Options:	`-v`	*file* Use 'view' mode for named file
	`-e`	*file* Use 'edit' mode for named file
	file	Edit file

1.4.54 unlink

Syntax: `unlink [`*opts*`]` *modname* `[`*opts*`]`
Function: unlink modules from memory

Options:	`-z`	get list of module names from standard input
	`-z=`*path*	get list of module names from *path*

1.4.55 unsetenv

Syntax: `unsetenv` *parameter*
Function: delete the specified environment variable

1.4.56 w

Syntax: `w`
Function: wait until a child process terminates
Comment: `w` waits until the first child process terminates if there are several running

1.4.57 wait

Syntax:	`wait`
Function:	wait until all child processes have terminated

1.5 μMacs command list

This chapter contains a list of μMacs commands.

Page control

Ctrl-V	Scroll down
Ctrl-Z	Scroll up
Esc <	Begining of file
Esc >	End of file

Moving the cursor

Ctrl-F	Forward character
Ctrl-B	Backward character
Esc F	Forward word
Esc B	Backward word
Ctrl-A	Front of line
Ctrl-E	End of line
Ctrl-N	Next line
Ctrl-P	Previous line
Esc P	Front of paragraph
Esc N	End of paragraph

Searching

Esc S	Search forward from cursor position. Type in a string to be searched for at the prompt and end it with Esc. Either case matches either unless EXACT mode is set for the current buffer
Esc R	As above, but Reverse search from cursor position.

Replacing

Ctrl-R	Replace all instances of first typed-in string with second typed-in string. End each string with Esc.
Esc Ctrl-R	Replace with query. Answer with:
Y	replace and continue
N	no replacement and continue

!	replace the rest
Ctrl-G	cancel
.	exit to entry point
?	Get a list of options

Capitalizing and transposing

Esc U	UPPERCASE word
Esc C	Capitalize word
Esc L	lowercase word
Ctrl-T	Transpose characters
Ctrl-X Ctrl-L	lowercase region
Ctrl-X Ctrl-U	uppercase region
Esc Q	Quote next entry, so that control codes may be entered into text

Regions and the kill buffer

Esc Space	set MARK at current position
Ctrl-X Ctrl-X	eXchange mark and cursor

A REGION will then be continuously-defined as the area between the mark and the current cursor position. The KILL BUFFER is the text which has been most recently saved or deleted.

Deleting and inserting

Backspace	Delete previous character
Esc Backspace	Delete previous word
Ctrl-D	Delete next character
Esc D	Delete next word
Ctrl-K	Close (delete) to end of line
Ctrl-O	Open (insert) line
Ctrl-W	Delete region between mark (set using Esc Space) and cursor
Esc W	Copy region to kill buffer

Copying and moving

Ctrl-W	Delete (Wipe) region
Esc W	copy region to KILL buffer
Ctrl-Y	Yankback save buffer at cursor

Generally, the procedure for copying or moving text is:

1.	Mark a REGION using Esc Space at beginning and cursor at end.
2.	Delete it (with Ctrl-W) or copy it (with Esc W) into the KILL buffer.
3.	Move the cursor to the desired location and yank it back (with Ctrl-Y).

Modes of operation

Ctrl-X M	Add Mode
OVER	Replaces (overwrites) rather than inserts characters
WRAP	Turns on word wrap (automatic carraige return).
VIEW	Allows viewing file without insertion and deletion.
EXACT	All searches done with exact case matching
CMODE	Automatic indenting for C program entry (automatically set on any files ending with .C or .H)
Ctrl-X Ctrl-M	Delete Mode

On-screen formatting

Ctrl-X F	Set fill column
Esc O	Format paragraph so that text lies between margins
Ctrl-X =	Position report – displays line number, char count, size

Multiple windows

Many WINDOWS may be active at once on the screen. All windows may show different parts of the same buffer, or each may display a different one.

Ctrl-X 2	Split the current window in two
Ctrl-X 1	remove all but current window
Ctrl-X O	cursor to next window
Ctrl-X Z	Enlarge current window
Esc Ctrl-V	scroll down, other window
Esc Ctrl-Z	scroll up, other window

Multiple buffers

A BUFFER is a named area containing a document being edited. Many buffers may be activated at once.

Ctrl-X B	Switch to another buffer. Return = use just-previous buffer
Ctrl-X Ctrl-B	Type buffer directory in a window

`Ctrl-X Ctrl-F`	Find file; read into a new buffer created from filename.
`Ctrl-X K`	Delete a non-displayed buffer.
`Ctrl-X X`	Switch to next buffer in buffer list

Reading from disk

`Ctrl-X Ctrl-F`	Find file; read into a new buffer created from filename. (This is the usual way to begin editing a new file.)
`Ctrl-X Ctrl-R`	Read file into current buffer, erasing its previous contents. No new buffer will be created.
`Ctrl-X Ctrl-I`	Insert file into current buffer at cursor's location.
`Ctrl-X Ctrl-V`	Find a file to make current in `VIEW` mode

Saving to disk

`Ctrl-X S`	Save current buffer to disk, using the buffer's filename as the name of the disk file. Any disk file of that name will be overwritten.
`Ctrl-X Ctrl-W`	Write current buffer to disk. Type in a new filename at the prompt to write to; it will become the current buffer's filename.
`Esc Z`	Write out all changed buffers and exit uMACS

Accessing the operating system

`Ctrl-X !`	Send one command to the operating system and return
`Ctrl-X C`	Start a new command processer under uMACS
`Ctrl-X Ctrl-C`	Exit uMACS
`Esc Z`	Write out all changed buffers and then exit uMACS

Special keys and key bindings

`Ctrl-G`	Cancel current command and return to top level of processing.
`Ctrl-U`	Universal repeat. May be followed by an integer (default = 4)
or `Esc` *digit*	and repeats the next command that many times.
`Esc X`	Execute a named (and possibly unbound) command
`Esc K`	Bind a key to a named command
`describe-bindings`	List the current key bindings in a buffer

Chapter 2

OS-9 systems at SEST

2.1 General

We have three OS-9 machines running at SEST: ANTLIA, ICCABIN and ICCNTRL. There is also a fourth machine, FORNAX, which can be used for program development. It is also intended as a spare for ANTLIA.

The physical locations of the OS-9 computers are as follows: ANTLIA is residing in the pedestal, ICCABIN in the cabin and ICCNTRL in the control room. FORNAX is usually sitting in the lab.

All four machines are EPROM-based, although ANTLIA and FORNAX also feature a hard disk each. However, all machines boot from the EPROMs, which thus contains everything needed to boot up and access the system (for example `telnet` and `ftp` daemons). If the disk on ANTLIA becomes corrupted, it can thus be reformatted and restored on-line while running the machine from the EPROM.

The disk on ANTLIA has a size of approximately 200 MB and the disk on FORNAX (which can be used to replace a broken disk on ANTLIA) has a size of approximately 2 GB. Both disks are set up with address SCSI-1. This makes it impossible to boot from the disks, since OS-9 only allows boots from a unit occupying address SCSI-0. The disk of ANTLIA can be set to this address but the one on FORNAX can not, and to keep the systems similar, they have both been set to SCSI-1. The OS-9 system has been modified so that "`/h0`" refers to SCSI-1 instead of SCSI-0.

The whole system residing on the disk takes up about 50 MB.

The CPU load on ICCABIN and ICCNTRL is normally around 30% while it is 60–70% on ANTLIA with `startTCS` and `display` running.

Michel has added a lot of new facilities to the machines (for example "`reset`" and commands to analyse the VME bus) and they all reside under `~MICHEL` and `~MICHEL/CMDS`.

2.1.1 Setting the system clock

Since the OS-9 version we have is not Y2K compatible, Michel has created a new command to set the date and time to the new millenium. This program is called "`setimey2k`" and is installed on all our OS-9 computers. Most system applications will show the year with only two digits. Some of them may even show it as years since 1900, i.e. 2001 will be represented as year 101.

Both ANTLIA and FORNAX have battery supported clocks. ICCABIN and ICCNTRL do not have batteries and the date and time will thus be lost each time they are rebooted. There is, however, no need for them to have correct date and time (unless someone wants to recompile something using `make`).

The clock frequency on ANTLIA (but not on FORNAX) has been adjusted so that it won't drift off too much if the external time reference is lost.

The default OS-9 time slice of 10 msek has been changed on ANTLIA and FORNAX to 500 μsek to allow time critical programs like "`encoder`" and "`loop`" to run more often. The default value would have been marginally acceptable, but the new setting adds margin to this.

2.1.2 Modifying programs

Program source code is stored under `~MICHEL/C` and executables are stored under `~MICHEL/CMDS`.

A program can be compiled while another copy of the program is running, since the compiled executable is written to the disk while the active program is running from a copy residing in the RAM memory. The RAM copy is removed from the memory when the program stops executing unless the sticky bit is set (or the program resides in the EPROM).

The `make` facility is not used. Instead each program has an executable script called "*program*`.m`" that contains all the commands needed for the compilation.

Programs executed with a priority above 1000 can only be interrupted by other programs having an even higher priority.

The "`synchd`" program on ANTLIA runs on normal priority (128) and will thus be slowed down during compilations, which may lead to the time synchronisation alarm going of on VELA. The priority of this program could be raised to above 1000 to prevent this, since it consumes virtually no CPU-time at all.

2.1.3 Adding a new account

This is done by adding a new line in the password file (`/h0/SYS/password`) and by creating the new user's home directory and execution directory.

2.2 Antlia

After ANTLIA has been rebooted, the command "`starttcs`" must be run to start the TCS and the encoders must then be initialised. Otherwise, it is just to open the display program by typing "`display`".

2.2.1 startTCS

The `startTCS` program is a small script residing in `~MICHEL/CMDS` which starts up all software needed for the TCS. It runs the following programs:

- `synch` which synchronises the system clock with time reference coming from the external reference (usually the maser). This is done by making slight adjustments to the system clock ones a second until it is properly synchronised. Thus, if the date and time is off by more than a few seconds, this process will take a VERY long time. If that is the case, abort the program with `Ctrl-C` and set the date and time using "`setimey2k`" before restarting "`startTCS`".
- `synchd &` which is the server program that answer the calls for time synchronisation checks from VELA.
- `manager &` which starts a number of other servers needed for the TCS.
- `display` which finally launches the familiar TCS display.

The `manager` program called by `startTCS` then sets up the shared memory and launches 7 other programs:

- `SynchClock` which runs the `SetInt` interrupt handler that triggers synchronisation on the second pulse coming from the GPS.
- `sio` which ...
- `encoder` which reads the values from the encoders.
- `loop` which runs the actual control loop.
- `cid` which listens for commands being sent from VELA.
- `astra` which ...
- `subloo` which controls the subreflector.

Except for `SynchClock`, all these programs communicate between each other through shared memory. There are three shared memories: "`shame`" used by `sio`, `encoder`, `loop`, `cid` and `subloo` as well as "`acaps`" and "`astro`" used by `cid` and `astra`. The last two are a remnant from CFHT and Onsala software.

The `manager` program can also be run with the option "`INTERACTIVE`" which can be useful for debugging purposes.

2.2.2 Processes running on Antlia

The following processes should be running on ANTLIA:

```
1.Michel: procs -e
 Id PId Grp.Usr  Prior  MemSiz Sig S    CPU Time   Age Module & I/O
  2   0   0.0     128    4.00k   0 w        0.00  0:30 sysgo <>>>term
  3   2   0.0     128    4.00k   0 w        0.04  0:30 startup <>>>term
  4   0   0.0     128    2.00k   0 e     1:59.40  0:30 ifman
  5   0   0.0     128   24.00k   0 s        1.06  0:30 routed <term >>>nil
  6   0   0.0     128    2.00k   0 a        1.97  0:30 sockman
  7   3   0.0     128   20.00k   0 s        0.08  0:30 tsmon <>>>term
  8   0   0.0     128   16.00k   0 e        0.01  0:30 ftpd <>>>nil
  9   0   0.0     128   16.00k   0 e        0.11  0:30 telnetd <>>>nil
 10  11   0.2     128    8.00k   0 w        0.52  0:30 shell <>>>pks00
 11   0   0.0     128   24.00k   0 e     2:31.54  0:30 telnetdc <pks00
 12  10   0.2     128    8.00k   0 w        0.36  0:30 shell <h0 >>>pks00
 13  12   0.2     128    8.00k   0 e        3.54  0:29 synchd <h0 >>>pks00
 14  12   0.2     128   12.00k   0 s        0.25  0:29 manager <h0 >>>r0
 15  14   0.2     300   16.00k   0 s     3:24.10  0:29 astra <>>>r0
 16  14   0.2     200    8.00k   0 s        0.05  0:29 SynchClock <>>>r0
 17  14   0.2    1025   16.00k   0 s     1:01.31  0:29 sio <>>>r0
 18  14   0.2    1100   16.00k   0 s     1:25.96  0:29 encoder <>>>r0
 19  18   0.2     200    4.00k   0 s        0.07  0:29 SetInt <>>>r0
 20  14   0.2    1050   24.00k   0 s     1:58.89  0:29 loop <>>>r0
 21  14   0.2     180   68.00k   0 e       35.51  0:29 cid >>>r0
 22  12   0.2     128   20.00k   0 a     2:24.45  0:29 display <h0 >>>pks00
 23  14   0.2     180   16.00k   0 s     1:59.82  0:29 subloo <>>>r0
 24  25   0.2     128    8.00k   0 w        0.40  0:00 shell <>>>pks01
 25   0   0.0     128   24.00k   0 a        0.57  0:00 telnetdc <pks01
 26  24   0.2     128   24.00k   0 *        0.45  0:00 procs <>>>pks01
```

The list with the "`procs`" command is:

```
1.Michel: procs
 Id PId Grp.Usr  Prior  MemSiz Sig S    CPU Time   Age Module & I/O
```

```
 10  11   0.2     128   8.00k  0 w      0.52 0:34 shell <>>>pks00
 12  10   0.2     128   8.00k  0 w      0.36 0:34 shell <h0 >>>pks00
 13  12   0.2     128   8.00k  0 e      4.19 0:34 synchd <h0 >>>pks00
 14  12   0.2     128  12.00k  0 s      0.27 0:34 manager <h0 >>>r0
 15  14   0.2     300  16.00k  0 s   3:58.42 0:34 astra <>>>r0
 16  14   0.2     200   8.00k  0 s      0.05 0:34 SynchClock <>>>r0
 17  14   0.2    1025  16.00k  0 s   1:11.08 0:34 sio <>>>r0
 18  14   0.2    1100  16.00k  0 s   1:39.58 0:34 encoder <>>>r0
 19  18   0.2     200   4.00k  0 s      0.07 0:34 SetInt <>>>r0
 20  14   0.2    1050  24.00k  0 s   2:16.77 0:34 loop <>>>r0
 21  14   0.2     180  68.00k  0 e     39.09 0:34 cid >>>r0
 22  12   0.2     128  20.00k  0 a   2:47.42 0:34 display <h0 >>>pks00
 23  14   0.2     180  16.00k  0 s   2:19.02 0:34 subloo <>>>r0
 24  25   0.2     128   8.00k  0 w      0.36 0:00 shell <>>>pks01
 26  24   0.2     128  24.00k  0 *      0.30 0:00 procs <>>>pks01
```

2.2.3 Making back-ups

...

2.2.4 Restoring a corrupted hard disk

...

2.3 Iccabin

2.3.1 Processes running on Iccabin

The following processes should be running on ICCABIN:

```
$ procs
 Id PId Grp.Usr  Prior  MemSiz Sig S    CPU Time   Age Module & I/O
  2   0   0.0     128   0.25k  0 w        0.00   ??? sysgo <>>>term
  3   2   0.0     128   6.75k  0 w        0.01   ??? shell <>>>term
  4   3   0.0     128   9.00k  0 w        0.06   ??? startup <>>>term
  5   4   0.0     128  15.50k  0 s        0.03   ??? tsmon <>>>term
  6   0   0.0     128  20.25k  0 s     1:13.09   ??? routed <>>>nil
  7   0   0.0     128   2.00k  0 e    51:14.96   ??? ifman
  8   0   0.0     128   2.00k  0 a       29.47   ??? sockman
  9   0   0.0     128  12.00k  0 e        0.01   ??? ftpd <>>>nil
 10   0   0.0     128  12.00k  0 e        0.08   ??? telnetd <>>>nil
 11   0   0.0     180  11.00k  0 e  1:48:22.81   ??? comsub <>>>r0
 12   0   0.0    1100   9.00k  0 s  2:35:31.60   ??? subcontrol <>>>r0
```

```
 13   0   0.0      180   11.25k   0 e    17:46.83   ??? iccabin <>>>r0
 14  15   0.0      128    6.75k   0 w        0.22   ??? shell <>>>pks00
 15   0   0.0      128   22.00k   0 a        0.33   ??? telnetdc <pks00
 16  14   0.0      128   18.00k   0 *        0.48   ??? procs <>>>pks00
```

2.4 Iccntrl

2.4.1 Processes running on Iccntrl

The following processes should be running on ICCNTRL:

```
$ procs
 Id PId Grp.Usr  Prior  MemSiz Sig S    CPU Time   Age Module & I/O
  2   0   0.0      128    0.25k   0 w        0.00 2255:59 sysgo <>>>term
  3   2   0.0      128    6.75k   0 w        0.01 2255:59 shell <>>>term
  4   3   0.0      128    4.00k   0 w        0.06 2255:59 startup <>>>term
  5   4   0.0      128   15.50k   0 s        0.04 2255:59 tsmon <>>>term
  6   0   0.0      128   12.00k   0 e        0.07 2255:59 ftpd <>>>nil
  7   0   0.0      128    2.00k   0 e  1:15:00.26 2255:59 ifman
  8   0   0.0      128    2.00k   0 s     1:17.98 2255:59 sockman
  9   0   0.0      128   12.00k   0 e        0.27 2255:59 telnetd <>>>nil
 10   0   0.0     5000   11.00k   0 e  3:29:53.24 2255:59 ich <>>>r0
 11   0   0.0      500    7.00k   0 e        0.01 2255:59 icb <>>>r0
 12  13   0.0      128    6.75k   0 w        0.15  0:00 shell <>>>pks00
 13   0   0.0      128   22.00k   0 e        0.27  0:00 telnetdc <pks00
 14  12   0.0      128   18.00k   0 *        0.38  0:00 procs <>>>pks00
```