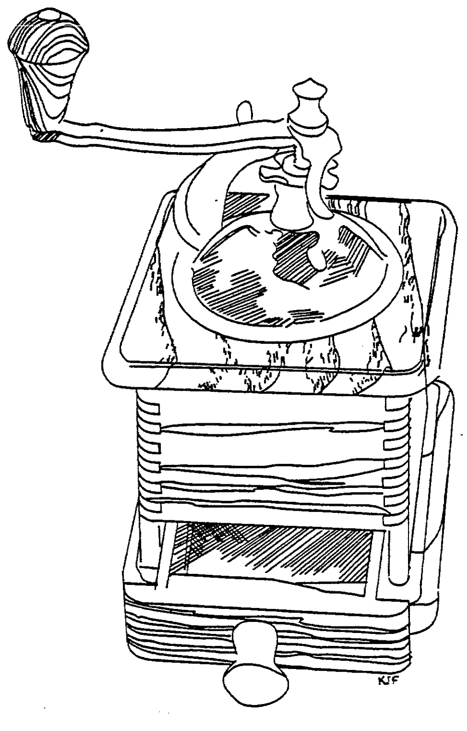
CARNEGIE-MELLON UNIVERSITY

DEPARTMENT OF COMPUTER SCIENCE

SPICE PROJECT

Spice Commands and Utilities

Edited by Mario R. Barbacci, Robert V. Baron, and Mary R. Thompson



27 August 1984

Spice Document

Keywords and index categories:

Machine-readable file: [cfs]/usr/spice/spicedoc/aug84/intro/commands/UtilMan.mss

Supported by the Defense Advanced Research Projects Agency, Department of Defense, ARPA Order 3597, monitored by the Air Force Avionics Laboratory under contract F33615-81-K-1539. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the U.S. Government.

Table of Contents

Introduction

This document provides a description of how to use almost all of the Spice commands and utilities. You will find that some of the more powerful commands and utilities are fully documented in their own, separate manuals and have only cursory documentation here.

If you have difficulties with a command or utility and you find the documentation insufficient or unclear, by all means report your problem, as described below.

Problem reports

All reports of problems related to Spice or its documentation should be mailed to the ArpaNet address **Spice@CMU-CS-Spice**. This address may be abbreviated to **Spice@Spice** on CMU Computer Science Department computers. To keep track of the latest changes in systems, documentation, and procedures, all Spice users should read the "Spice" bulletin board on any of the local host-machines.

Notation Conventions

We have attempted to use the following notations throughout this manual when describing the syntax of commands:

SYMBOL	MEANING
|	used to separate alternatives
( )	used to indicate the scope of alternatives for | where it would not be otherwise obvious.
[ ]	used to enclose an optional feature (i.e something that can appear 0 or 1 time)
{ }	used to enclose something that can appear 0 or more times
Bold	a literal, something that must be typed as it is spelled.
Italic	a metaname, something that stands for a group or class of names. For instance, a file name can be described as: *Name.Ext* that is, a *Name*, followed by the literal ".", followed by an *Ext*. *Name* stands for <u>any</u> filename (i.e. a sequence of letters, digits, underscores, periods, and hyphens), not just "N" "a" "m" "e".
CTRL	used to prefix a control key (i.e. a key that must be typed while the keyboard "control" key is down). For instance, CTRL u indicates typing the 'control' and 'u' keys together. The keyboard control key acts like a conventional typewriter shift key. Nothing happens when it is typed or pressed by itself and you can keep it down while typing several CTRL keys in a row.
ESC, INS	the "escape" key (marked as **ins** on PERQ1s and PERQ1As, and as **acc esc** on PERQ2s).
DEL	the "delete" key (marked as **rej del** on PERQ2s).
HELP	the "help" key (not the word "**h**" "**e**" "**l**" "**p**").
LF	the line-feed key
RETURN	the carriage return key

Because the documentation has been gathered from several distinct sources, we may not have been as successful as desired. Please report any discrepancies by mail to Spice@CMU-CS-Spice.

CTRL ?

KeyWords: CTRL ?, control ?, ↑?, directory, quick

Function: lists file names

Syntax: CTRL ?

Description: This command is really a part of the intra-line editing functions provided by the Shell. Since its function is closely related to that of the **Directory** command, it is included as a separate entry in this manual.

CTRL ? typed all by itself, at the beginning of a command line, will list all files in the current directory. Typed after a partially specified filename, CTRL ? will list all filenames that could complete it (if no candidate filenames are found, the screen will flash, to indicate a search failure).

If the current directory is empty, this command displays all the filenames in the first non-empty directory in the **default:** search list.

CTRL ? is faster than **Directory** but it does not provide the functionality of the latter. In particular, it does not accept switches.

See Also: Directory, Shell, SetSearch

CTRL LF and CTRL \

KeyWords:	CTRL LF, CTRL \, ↑LF, ↑\, control LF, control \, moremode, pause, terminal-pause, CTRL q, CTRL s, CTRL Q, CTRL S, control q, control s, ↑q, ↑s, ↑Q, ↑S
Function:	enable and disable "more mode" to control scrolling.
Syntax:	CTRL LF CTRL \
Description:	The shell has the ability to suspend output to a window if it would cause information to scroll off the top of that window. This is analogous to the TERMINAL PAUSE (ON) END-OF-PAGE feature in the TOPS-20 system, or "more-mode" in the Vax Unix systems. Typing CTRL LF Enables "more-mode" to control scrolling. More-mode is disabled by default. Typing CTRL \ Disables "more-mode" to control scrolling. More-mode is disabled by default. With "more-mode" enabled, whenever the shell notices that output is about to scroll off the top of the window, it stops and displays the last line in the window in reverse video (white letters over black background). Typing LF allows output to continue. Typing <u>anything</u> other than a Sapphire CTRL DEL command, will cause scrolling to continue. However, whatever was typed is saved and used as input the next time the shell or the program doing the typing wants to read something from the keyboard.
See Also:	Shell

CTRL p and CTRL n

KeyWords:	CTRL p, CTRL n, ↑p, ↑n, control p, control n, command-retrieval, command-buffer, history, repeat
Function:	retrieve previous shell command lines
Syntax:	CTRL p CTRL n
Description:	The CTRL p and CTRL n commands are used to retrieve previous command lines. Previously typed command lines are kept in a "ring" and the user can move around the ring, typing CTRL p to move backwards or CTRL n to move forwards. The old command line appears on the screen, with the cursor positioned to the right, as if the user had just typed it in. Successive CTRL p or CTRL n commands can be typed until the desired line is retrieved. It can then be edited, if necessary. Typing RETURN terminates the editing and the command line is then interpreted by the shell. Notice that the cursor does not have to be at the end of the line in order to type RETURN
See Also:	Shell

CTRL v and CTRL V

KeyWords: CTRL v, CTRL V, ↑v, ↑V, control v, control V, transcript, session, photo, replay, history

Function: retrieve previous shell screen buffers

Syntax: CTRL v

CTRL V

Description: The shell keeps a transcript of the session (i.e. user command lines and program output that appeared on a window) in an internal buffer. One can peruse this buffer by typing CTRL V and CTRL v to move backwards and forwards, one windowful at a time.

The buffer keeps about three windowfuls of text, thus the size of the window determines how much is retained in the buffer. Changing the size of the window, alters the size of the buffer for the remainder of the session (or until the next window size change).

The transcript buffer does not behave like the command line buffers retrieved by the CTRL p and CTRL n commands. It can not be used to "replay" a portion of session or to retrieve previous commands. Typing anything, while displaying some previous portion of the transcript, simply advances to the end of the transcript and inserts the user input at the place were the cursor was positioned before the user started redisplaying the session transcript.

See Also: Shell, CTRL p, CTRL n

CTRLw and CTRL W

KeyWords: CTRL w, CTRL W, ↑w, ↑W, control w, control W, wrap, truncate

Function: control wrapping or truncating of window text lines.

Syntax:

CTRL w

CTRL W

Description: Typing CTRL w turns "wrap" mode on. That is, output lines that would be too wide for the current window width are continued onto the next line, i.e. are "wrapped" around. The wrapping is not permanent; if the window width is changed, the lines are redisplayed as per the new width.

Typing CTRL W turns "wrap" mode off. That is, output lines that are too wide for the current window width are truncated. This is not a permanent truncation; if the window witdh is changed, the lines are redisplayed as per the new width.

See Also: Shell

?

KeyWords: ?, abbreviations, alias, commands, shell

Function: print the table of shell commands.

Syntax: **?**

Description: **?** prints the shell *Alias* table. If a command name is found in this table, the shell will use its definition entry instead of the command name (See **Alias**). Alias prints only the combined command and description field.

The *Alias* table is also used for command abbreviation; any unambiguous prefix of a command name found in this table may be used in place of the full command name. For a *command* not found in the *Alias* table, the *command.run* is sought in the searchlist, **run:**. No automatic command abbreviation or completion is provided. But, file name completion can be requested of the shell using CTRL ?. This mechanism provides a form of command name completion. Note, that CTRL ? uses **default:** rather than **run:** for its search list.

The *Alias* table is set at shell startup time. Entries can be added or altered using **Alias** and removed using **UnAlias**.

Note: The only safe way to replace a shell command with one of your own is to define your command as an *Alias* for the shell command.

```
Alias help /sys/user/me/myhelp.run
```

is necessary to replace the *help* command.

See Also: Alias, Define, Shell, UnAlias.

Access

KeyWords:	access, permit, remote access, protection
Function:	Sets or shows remote access to the local disk.
Syntax:	**Access** [*arg*] [-*switch*]

Description:

Access sets or shows the access privileges that remote users are allowed when accessing your local disk through the Sesamoid foreign file referencing mechanism. For a remote user to access files on your machine, he must first know the name of your file system as it is recorded in the **<boot>SysName** file on your disk. Then your Sesamoid must allow **read** or **full** access to your disk. When your machine is booted, Sesamoid grants **read** access by default. A call to **Access** is only necessary if you wish to change that permission.

If **Access** is called with no argument, it will print out the current access that remote users are allowed to the local disk. If it is called with an argument, it will set the remote access according to the value of the argument.

The input argument takes the following values:

no	Allows no remote access.
read	Allows only reading by remote users.
full | write	Allows reading and writing by remote users.

Switch:	**-Help**	Prints a brief help message.

Note: In entering the input argument, only the first letter is needed and it may be either upper or lower case.

Alias

KeyWords: alias, abbreviations, commands

Function: define a shell command alias.

Syntax: **Alias** *CommandName Definition* { *-Switches* } [*Documentation*]

Description: Alias assigns another name to an existing command. You may also specify that a default filename be set. When you type the new CommandName or an unambiguous abbreviation of it, the shell looks for the string in an Alias Table and substitutes the predefined value (the definition), possibly including the default filename.

Aliases are also useful to define commands that override defaults by using switches.

If **Alias** is invoked without an argument, it types out a syntax reminder.

The CommandName is any name you wish to assign. The Definition is a previously defined command or program name, including any arguments and switches that are valid for that command. If the Definition contains a nonalphanumeric character, including a blank, it must be quoted by it with a backslash (\). (Or you can enclose the Definition in quotes - "...".) Do not abbreviate the command name you are using in the Definition.

Do not use a **previous Alias** as the Definition.

If you wish to use one of the Alias switches described below, it must come after the switches that are part of the Definition and before the Documentation.

Documentation is any information that you wish to give about the command. The information will appear with the command when you give the ? command. If the Documentation contains any non-alphanumeric characters, enclose the Documentation in quotes ("...").

After you have set an Alias, typing the CommandName is equivalent to typing the Definition. You can add more arguments and switches before typing Return.

You may redefine existing aliases by issuing the **Alias** command again with new information (but do not use a previous **Alias** as the Definition).

To remove an **Alias** entry, use the Unalias facility.

Switches: The *-Switches* must appear before the *Documentation*.

-Setdefault Remember the first argument of the expanded command line as the *default file name*. If there is no argument, *default file name* is not set.

-Usedefault If no argument exists in the expanded command line, substitute the *default file name* into the expansion of this *Alias* after the *Definition* string.

-Help Print a help message.

Note: The only safe way to replace a shell command with one of your own is to define your command as an *Alias* for the shell command.

```
     Alias help /sys/user/me/myhelp.run
```

is necessary to replace the *help* command.

Examples:

alias ls direct\ \-sort\ = size

This command defines a command "ls" which is equivalent to the command "direct" and specifies that the entries are to be sorted in order of size. Note the use of \ to quote non-alphanumeric characters, including spaces.

alias ls "direct -sort = size"

This is equivalent to the example above. Note that the quotes may be used to enclose an entire string of characters containing non-alphanumeric characters.

alias edit run\ edito -usedefault -setdefault alias compile run\ pascal -usedefault -setdefault

This command allows
edit <file>
compile

or
compile <file>
edit

to work implicitly using the same file in the second command.

alias help /sys/user/smith/myhelp.run

This command defines the help file as the MyHelp.Run file instead of the system help file.

See Also: ?, Shell, UnAlias.

Append

KeyWords: append, cat, concatenate

Function: Concatenates two or more files

Syntax: **Append** *File1*, *File2* {,*FileN*} [*-switch*]

Description: **Append** is used to concatenate files. It takes a series of input files and puts them sequentially at the end of the first input file. You can not specify a different output file. All files must exist, though they may be empty.

Switch: -Help	Prints a brief help message.

Example:

append file1,file2

will take the contents of **file2** and add them to the end of **file1**. **file2** is left unchanged.

append file1 file2 file3

first appends **file2** to the end of **file1** and then appends **file3** to the end on the combined **file1file2.**

Asm

KeyWords:	asm, assembler, C, spoonix
Function:	**Asm** is an assembler for turning assembly files into object format files (".0").
Syntax:	**Asm** {-switch [*value*]} *InputFile* {{*-switch*} *InputFile*}
Description:	**Asm** takes 1 or more *InputFiles* and produces an *OutputFile* in a format that the C linker can understand. **Asm** can only assemble one logical file at a time, but that logical file can be broken into separate real files. **Asm** reads and parses its arguments as it goes. This means that if you have a logical file split up into several pieces you can control the optimization and or debugging of those pieces individually. Since the output file is written after the command line is entirely processed, only the last file designated as the output file will get written.

Switches:

-D	Disable debugging output.
-d	Enable debugging output.
-O	Enable optimizer.
-o *OutputFile*	Indicate that the next argument is the name of the output file. If this switch isn't specified, the output file name defaults to 'a.out'.

Notes:	**Asm** is primarily intended to process the output of the C compiler. While it can be used to assemble hand-writen code, it is not tailored for that purpose and may not do what you expected.
See Also:	cc, Lnk, "The Assembler and Related Programs"
Bugs:	Currently, the code, data, and symbol areas of the output file are limited to 32K bytes each. This is a limitation of the Pascal Compiler. Rumor has it that a new compiler is out there that will remove the restriction. We'll see. If you do overflow one of the areas, the assembler will get an uncaught exception (Expression out of Range). For the time being, you will have to split your files up if you run into this problem. It isn't very likely that you will.

Bye

KeyWords:	Bye, shutdown, trap, turnoff, leave, debug
Function:	Safely shuts down the system
Syntax:	**Bye**

Description: Writes out various system tables to disk and then calls the Accent debugger, to stop all paging. This should be called before turning off your PERQ. Once the Accent debugger has been entered (it will print out a message on the top of the screen in reverse video (white letters on a black background) it is safe to turn off the PERQ.

Bye dismounts all the partitions before calling the Accent debugger. It is not possible to return from the Accent debugger and continue working.

Switches: **-Help** Prints a help message.

Note: If you want to get into the Accent kernel debugger and be able to return, hold down all three mouse buttons and type CTRL-ESC(INS).

CC

KeyWords:	cc, pcc, cpp, spoonix, c compiler
Function:	Compiles a C program

Description: A **cc** command will eventually be written that will compile and link C programs in a manner similar to the way that it is done in Unix. The following recipe is equivalent to "cc file1.c file2.o -o file3"

1. cpp file1.c file 1.i
   --run c preprocessor
2. pcc file1.i file1.2
   --run portable C compiler
3. asm file1.2 -ofile1.o
   --assemble code
4. lnk libc.lib file1.o file2.o -ofile3
5. --link the ".o" file together
6. --libc.lib is a library of c subroutines necessary for most programs

See Also: Asm,Lnk,Spoonix, Spoonix section in *Spice Users Manual.*

Bugs: This command is not implemented yet.

ChangeOwner

KeyWords:	ChangeOwner, SetProtect, chmod, chown
Function:	Changes the owner of a file
Syntax:	**ChangeOwner** *FileName*[,*UserName*]{*-Switch*}

Description: ChangeOwner allows you to change the owner of any file that you have owner rights on. You have owner rights on a file if:

- You are logged-in and your UserID matches the UserID of the file.
- You are not logged-in, giving you a UserID of No-User and ownership rights on all files on your local workstation.
- Or the file is owned by No-User, in which case all users are granted owenership rights.

If you do not specify a *UserName*, the owner of the file is changed to the user that you are currently logged-in as. (This fails if you are not logged-in.) If you do specify a *UserName*, you will be prompted for that user's password. It is also possible to specify **No - User** (exact case required) as a *UserName*. In this case no password is required.

Switches:

-DirOnly	If *FileName* contains wildcards, this switch will cause only directories that match the name to be changed. (This is a time saving feature, since if a directory prohibits access then none of the files in that directory may be accessed. Thus it is sufficient to change owners on only the directories to restrict access to all the files in that directory.)
-Help	Prints a help message.

See Also: SetProtect, description of access rights in the *Introduction to the Spice User's Manual*

ChangeUser

KeyWords:	ChangeUser, Changepassword, password
Function:	changes your user name (the name you login under), password, shell or profile.
Syntax:	**ChangeUser**

Description:

ChangeUser modifies all or any part of the information that is stored for you in the System.Users file. This is the information that is used by the Authorization Server when you login. You must be logged-in to run this program and it will only change your user information.

ChangeUser prompts for the items that you might want to change. If you do not want to change a particular item, just type a carriage return after the prompt.

The questions this program asks are as follows:

Current password: Enter your password.

Should I change your password? Enter y or n.
If you answer y(es) the program continues:
New Password: enter your new password
Again: enter your new password again
If you answer n(o) the program continues to the next question.

Profile [Boot:Default.Profile]:
you may enter a new profile name if you wish.

Note: Currently the Profile is not used by the system. All functions normally served by a profile are handled by ShellCommands.cmd.

ShellName [Boot:Shell.s5.run]:
you may enter a new Shell name if you wish.

Once you have entered a new shell name or typed carriage return, the program will type:

Login information changed.

Switches:	**-Help**	Prints a help message.

Notes: The Authentication Server must be running and you must be logged-in for this program to work.

See Also: ChangeOwner,Login

Chat

KeyWords:	Chat, telnet, ethernet, talk, remote-login, concept, c100	
Function:	Terminal emulator with ethernet connection to Main frames.	
Syntax:	**Chat** [[*Host*]{-*Switch*}	

Description:

Chat allows you to 'talk' to various mainframes through the ethernet front end (chat service). It emulates a Concept terminal, and currently it is closest to a Concept-100.

One can 'talk' to several machines at the same time with only one chat running on your Perq. Chat maintains a separate 'screen' for each one and displays the current job, which can be switched with chat commands.

The Help key gets you out of chatting and puts you in top level. At this point you have a full choice of commands, but only a 'mini-menu' on the mode line is displayed. To display the full menu, type 'M'. This will also display the last several errors at the bottom of the screen. Note that once you have typed 'M' there is no longer a 'current job'. This will be changed at some point in future.

The Toggle Block Cursor, Toggle Tablet Mode, Invert Screen, and Reset Screen commands all affect the current job only. You must be connected to a job first, then hit the Help key and give one of these commands. In particular, once you display the full menu with the 'M' command, you must reconnect to a job before any of these commands will work.

Switches:	**-BlockCursor**	Make the block cursor the default for the Concept terminal.
	-ConceptShape	Reshape the Sapphire window in which Chat is running to 80 X 25 characters.
	-Help	Get this help and terminate.
	-ReverseVideo	Make White on Black the default for all connections.
	-Tablet	Turn on 'Tablet' so that mouse clicks send an escape sequence to the host (see /usr/local/lib/emacs/maclib/mouse.ml).
Commands:	**B**	toggle cursor mode for current job (Block vs. underline).
	C	Close a connection. You are prompted with a list of open connections, and you type the number of the one you want to close.

D	toggle Debugging mode. If debugging is turned on a new sapphire window will be prompted for and created. The Toggle Verbatim command switches between these modes of debugging operation. This is helpful if strange behavior is noticed on the current job's screen.
E	Expand window. If emacs has been used, it may redefine the 'Concept window' to be 24 X 80 characters, and one can use this command to regain full use of the sapphire window if it is bigger.
H	send Help. Sends the help character (↑G) to the current job. This will probably be disabled at some point because of its dubious usefulness.
I	Invert screen. Change the screen mode of the current job (Black on White vs. White on Black). The default is Black on White, unless the -ReverseVideo switch was given.
K	Change exit character. Not yet implemented, may not be implemented due to dubious usefulness.
M	Menu. Get a full menu of commands and a list of recent errors.
O	Open a new connection. Prompts for machine name on the mode line.
Q	Quit. Closes all open connections and terminates the program.
R	Reconnect to a job. Prompts for a job *number*.
S	reshape to Standard size. Reshapes the sapphire window to 25 X 80.
T	toggle Tablet status for the current job. The default is not to send tablet clicks, unless the -Tablet switch was given.
V	toggle Verbatim mode for debugging. See the Toggle Debugging command. If Debugging mode is enabled, and if Verbatim Debugging is on, the additional window will display all characters as they are received, without interpreting any as special (escape and control characters). Otherwise, the additional window will display the names of escape sequences as they are received from the current host.
INS	Reset current job's screen. Clears screen and redefines the Concept window to the full height of the Sapphire window.

Notes:	The INS key maps onto the Esc character, the DEL key maps onto Rub Out and the OOPS key maps to ↑U. If an Esc-Esc is transmitted to the terminal emulator, it will transmit the terminal height followed by a ↑D. This is useful for defining TERMCAP to have a larger window size.
See Also:	Concept-108 or Concept-100 manual.
Files:	chat.keytran /usr/local/lib/emacs/maclib/mouse.ml
Bugs:	Occasionally the host machine will disconnect a job if it has been sitting a while. Still throws you into the debugger on occasion when a job terminates abnormally. Sometimes it has packet errors and messages will appear on the screen instead of in the menu window; to get rid of them, reconnect to the job. Sometimes there is a delay in the host echoing your characters. It seems to catch up faster if you type more. Sometimes you will get a BSP error on initial startup, if you start typing too soon, and some of your first characters will get lost. Wait a few seconds and try again.

Chili

KeyWords: Chili, Directory, Copy, Rename, Delete, Edit, Files

Function: Chili is a file management utility program.

Syntax: **Chili**

Description: Chili is a file management utility for Spice. It allows you to browse through the files on your own or other Perqs and delete, copy, rename, or invoke an editor on selected subsets of files. Using Remdef, these facilities are also available for files on a remote Vax. While you can perform destructive file operations using Chili, it contains various safety features to lessen the chance of your doing so inadvertently. Chili's user interface is provided by COUSIN and so is a graphical form. The full set of interface facilities available through COUSIN is described in the COUSIN user's manual which is part of the Introductory Spice Manual. An introduction to Chili can be found in an appendix of the COUSIN user's manual.

The following is a description of the fields in the Chili form:

Fields:

File Spec This field controls which files are displayed in the **Files** field. The files displayed are all the ones that match the file specification entered in the **File Spec**. File specifications can have wildcards and logical names, with relative specifications interpreted relative to the **Current Directory** field.

Files Shows a list of files that can be manipulated by the user. The **Copy**, **Rename**, **Delete** and **Edit** commands operate on the selected elements of the **Files** field. Selected elements are marked in boldface, while unselected ones are displayed in the regular font. Elements can be selected or deselected by pointing at them with the mouse and pressing any button; in addition, ↑s selects everything, while ↑<shft>s deselects everything.

Display Mode This field can have only two values: *Short* and *Long*, which are used to control the display format of the **Files** field. If the value is *Short* only the names of the files is displayed. If the value is *Long* then additional information about the file is also displayed.

Sort By This field can have four values: *Name*, *Last Access Date*, *Last Change Date* and *Size*. This values are used to control the order in which the files in the **Files** are listed[1].

[1] Currently, only sorting by *Name* is implemented.

Creation Time This field is located at the bottom left of the **Files** field, and its name is not displayed. This field can have two values: *Same Creation Time* and *New Creation Time*. These values control the date that is given to the files that are created with the **Copy** command.

Copy|Rename To This field is used to specify the destination for the **Copy** and **Rename** commands. If wildcards are used they should match the wildcards used in the **File Spec**. If the value is left empty or set to a directory name then all the files are Copied or Renamed to the given directory (if it is empty the **Current Directory** is used).

Confirmation Request
This field is used to ask the user yes|no questions for destructive operations. Answers to these questions can only be given by pointing to any of the buttons labeled **Yes, No, All** and **None** which are located at the right of the **Confirmation Request** field.

Confirmation This field is located just below the **Sort By** field, and its name is not displayed. This field can have the values *Ask For Confirmation* and *No Confirmation*. The values control whether Chili should ask for confirmation about destructive events. Note that if it is set to *No Confirmation* NO questions will be asked.

Free Space This field displays the number of free blocks left in the partitions of the disk. It is automatically updated after the **Delete** and **Copy** operations.

Current Directory
The value of this field is the current directory with respect to which relative file specifications are interpreted.

Commands:

Copy The copy command copies all the selected files in the **Files** field to the files specified in the **Copy|Rename To** field.

Rename This command renames all the selected files in the **Files** field to the files specified in the **Copy|Rename To** field[2].

Delete This command deletes all the selected files in the **Files** field.

Edit This command calls Oil on the file selected in the **Files** field. Note that only one file can be selected at a time.

[2]The Rename command is not yet implemented.

	Space?	The space command updates the **Free Blocks** field.
See Also:	Cousin, Copy, Rename, Delete, Edit, Details	

CMUFTP

KeyWords:	cmuftp, file-transfer, ftp, transfer, update
Function:	ethernet file transfer program.
Syntax:	**CMUFTP** [*Command Arguments*.]

Description: **CMUFTP** transfers files between machines on the ethernet. Note that **CMUFTP** will not transfer files from Perq to Perq. However, it is possible to directly reference files on a remote Perq by replacing the **sys** in the absolute pathname with the name of the remote Perq (i.e. the name stored in the file **<boot>SystemName** on the remote Perq's boot partition).

CMUFTP prompts for commands although single commands may be executed from the command line. Commands may be unambiguously abbreviated; the first character currently suffices for all commands. Any command line that begins with a colon (:) is ignored.

CMUFTP can accept a command from the command line, so it is possible to invoke **CMUFTP** from a shell command file. For example, the shell command line:

CMUFTP *@CmdFile*

will cause **CMUFTP** to take commands from file *CmdFile*.

The shell command line:

CMUFTP *↑CmdFile*

will retrieve a *CmdFile* and execute its commands.

When a file transfer is in progress, **CMUFTP** prints a " # " for every eighth packet transferred. (There are typically, but not always, 512 bytes in each packet.) This is to let the user know that progress is being made.

To avoid filling the screen with lengthy mesages, **CMUFTP** types just a single "." whenever it timeouts during a file transfer. It then reinitializes and tries again. Similarly, **CMUFTP** will type a single "?" whenever it gets a "retry" during a file transfer. If you get too many of these (more than a line-full of them), it may indicate that the server program on the remote host crashed. If this is indeed the case, the CMU-CSD operator can restart the server program.

Commands:	*@cmdfile*	will cause **CMUFTP** to read and execute commands from *cmdfile*.
	↑cmdfile	Retrieves *cmdfile* from remote host and executes it.
	?	gives a short list of the commands available.

: starts a comment. The rest of the line will be ignored.

Global lists the values of the internal state variables set by **Name**, **Mode**, **Login**, and **Paths**.

Help [*Command*] Gives a brief description of *Command*.

Keys invokes the keyboard command-interpreter recursively. This allows command files to give control temporarily to the user. To return to the next outer level, use the **Quit** command.

Login *UserName* specifies the remote user-id and password for user validation (the password is requested on the next line and is not echoed). This information is not actually checked on the remote host until a file-transfer is requested. If a "login-failure abort" is received at that time, then the user will be asked if he wants to login again.

Mode Auto|Text|Binary|1|2|Image

sets the file transfer mode. Because different machines have different internal representations of data it is necessary to specify the method used to transfer data. **Text** mode is used to transfer ASCII files such as sources. **Binary** mode is used to transfer **.seg**, **.press**, **.bin**, and other 8-bit binary files. These two modes satisfy most requirements. The modes **1**, **2**, and **Image** are used to transfer 16-, 32- and 36-bit binary files, respectively. The **CMUFTP** user is spared of most of this complexity, because when in **Auto** mode **CMUFTP** uses two rules that usually determine the correct mode. First, whenever **CMUFTP** retrieves a file it marks the file with the mode used to retrieve it, so it can use that same mode for future transfers. Second, **CMUFTP** recognizes several standard extensions for binary files; if the file is not marked, this is used to guess the file type. If all else fails the file is shipped in **Text** mode. If a file is shipped in the wrong mode, you can always specify the correct mode explicitly and reship it.

Name *HostName* sets the remote host for file transfers, where the name is any name recognized by the Ethernet name server. Optionally the host may be specified by a numerical address, using the syntax *NetworkNumber # HostAddress # ServerSocket*. The default host for **CMUFTP** is CMU-CS-Spice.

Paths *Remote-prefix* [*Remote-suffix* [*Local-prefix* [*Local-suffix*]]]

is used to specify strings to be added to the beginning and end of the remote and local filenames. The special sequence of two apostrophes in a row ('') may be used to indicate a null string. Examples of typical usage are:

```
p /usr/spice/pos/cmuftpG/seg/
p /usr/ram/rnd/ .p '' .pas
p '' [p100rm07]
```

Quit exits the current recursion-level of the **CMUFTP** command-interpreter. Except after a **Keys** command, this exits **CMUFTP**.

Retrieve *RemoteFile* [*LocalFile*]
gets a file from the current remote host. If *LocalFile* is not specified, it is assumed to be the same as *RemoteFile*. The remote and local prefixes and suffixes (specified in the **Paths** command) are added to get the complete filenames.

Store *LocalFile* [*RemoteFile*]
sends a file to the current remote host. If *RemoteFile* is not specified, it is assumed to be the same as *LocalFile*. The remote and local prefixes and suffixes (specified in the **Paths** command) are added to get the complete filenames. One usually has to use the **Login** command before sending files to any of the department machines that have file protection.

View *RemoteFile* retrieves a file and types it. The file is not retained on the PERQ. The rest of the command line is treated as the *RemoteFile* so quotes are not necessary even if it contains spaces. One common use of this command is to execute remote shell commands on Unix systems using the "|" convention. For example,

```
view | ls /usr/rhg
```

will list the specified directory and

```
view | finger
```

will show the current users. When using this remote command facility, the remote filename prefix and suffix (set with **Paths**) must be null!

Wait asks the question

```
Ready to continue? [Yes]:
```

until the user responds positively. This is used primarily in command files.

See Also: Update.

Note: Because local and remote filenames are imbedded between local and remote prefixes and suffixes, the user must take care that they are compatible with the remote machine's system. For example: specifying a unix remote prefix of "/usr/spice" and appending a filename of "random.file" will result in "/usr/spicerandom.file", when the user likely wanted "/usr/spice/random.file". It is a safe bet that all Unix prefixes should end in a "/".

Compare

KeyWords:	Compare, diff, difference, changes
Function:	Finds the differences between two files.
Syntax:	**Compare** [*FileName1*] [*FileName2*] [~*OutputFile*] {-*Switch*[= *Value*]}
Description:	Compare looks for differences between two files and prints the results into an output file if one is specified. If no output file is specified, the results on printed on the screen. If input files are no specified, Compare will prompt for them.

Switches:

-Differences	lists the lines that do not match.
-Linelength = *n*	compares up to this length line. You may specify from 1 to 255. The default is 100 characters.
-Match = *n*	The file is compared in chunks and this switch specifies the number of lines in a chunk. *n* may be between 1 and 100. -match = 1 would compare the files on line at a time. The default value is 6.
-UnequalColumns	Marks unequal columns.
-Help	Prints a help message.

Compile

KeyWords:	compile, cc, pascal, translate
Function:	invoke the Pascal compiler
Syntax:	**Compile** [*SourceFileName*] [~ *SegmentProgramName*] {*Switch*[= *Value*]}

Description:

Compile invokes the Pascal compiler. If no *SourceFileName* is specified, it uses the filename that was last used in an **Edit**, **Type**, or **Compile** command. **Compile** is an alias for the following command line:

```
     Run Pascal -usedefault -setdefault
```

See **Pascal** for a list of available switches.

Note:

Invoking the PERQ Pascal compilation facility through the use of the Pascal command by-passes any use of the default file remembered by the shell.

See Also: Pascal, Edit, Type, Alias

Copy

KeyWords:	copy, cp, duplicate, save, update
Function:	copy a file
Syntax:	**Copy** *SourceFile* [~] *DestinationFile* {*-Switch*}

Description:

SourceFile is copied to *DestinationFile*. You can copy across devices and partitions and from one Perq to another. You can also specify the device **Console:** for either the *SourceFile* or *DestinationFile*. If **Console:** is the *SourceFile* the end-of-file character for terminal-input is "CTRL Z".

The *SourceFile* for **Copy** may contain wildcards. See the **Directory** command for a description of wildcards. If the source has wildcards, then the *DestinationFile* must have the same ones in the same order.

When *SourceFile* contains no wildcards, *DestinationFile* may contain, at most, one occurance of the (*) wildcard. In this case, the whole non-directory part of the source name replaces the * in the destination.

Copy uses the search list (**default:**) to try to find *SourceFile.* Note that this is different from **Rename** and **Delete**, which always look in only one directory (**current:**).

If an error is discovered and wildcards are used, **Copy** asks whether it should continue processing the other files that match the input. This confirmation is required regardless of what switches are specified.

Copy may be used, in conjunction with the **Update** switch (see below) to update files on the same machine or between different PERQS.

Switches:

-Ask	Ask for verification for each file that is to be copied. **-Ask** is the default if wildcards are used.
-NoAsk	Overrides verification request for individual files. **-NoAsk** is the default when wildcards are not used.
-Confirm	requests confirmation before overwriting and existing file. **-Confirm** is the default.
-NoConfirm	No confirmation is requested before overwriting an existing file. The **-NoConfirm** switch implicitly sets the **-NoAsk** switch; the latter is probably safer for general use.
-NoSupersede	Do not replace a newer version by an older one.

-Update Keep the same date as the *SourceFile* on the *DestinationFile*. If the *DestinationFile* already exists and has the same date as *SourceFile*, do not perform the copy. If the existing *DestinationFile* has a later date than the *SourceFile*, you are asked if you wish to perform the copy. This achieves the same effect of the **Update** program.

Examples:

copy *.kst boot:*.kst
copies all the files with extension .kst in the current directory into the boot directory, giving them the same names there.

copy console: ~ newfile
Allows you to input a new file from the console. Type CTRL z when you are done with input.

copy -update /Giuse/User/Dp/*.run MyDp/*.run Updates a group of **.run** files from a remote machine. If the files are already up-to-date, no copying takes place.

copy dir1/prog.pas * Copies **prog.pas** from the directory **dir1** into the current directory with the name **prog.pas**

See Also: SetSearch, Rename, Directory, *Introduction to the Spice User's Manual* section on PathNames, and Network PathNames.

Bugs: The **Console:** feature does not currently work.

Cousin

KeyWords:	Cousin, user interface, command interpreter, form
Function:	provides form-filling user interfaces
Syntax:	**Cousin**

Description:

Cousin is a Spice server that can be used to implement uniform, "form-filling" interfaces between users and programs. Any program can become a "**Cousin**-application" by providing to this server information describing the "form" used by the program and communicating with the user through this form, using **Cousin** as a mediator.

As a server, **Cousin** is not meant to be invoked directly by the users, rather it should be started once, typically after booting the machine, and left running on its own.

Invoking a "Cousin-application" program is done in the normal way (e.g. typing the application name as a shell command). The application program can then ask **Cousin** to take over and mediate the interactions between the user and the application.

When the user starts the execution of the application program, the server is invoked and it displays on the screen the appropriate, application-specific form to be filled. The user types parameters, switches, commands, etc. in the appropriate slots in the form. **Cousin** takes care of verifying that the data contained in the slots is complete or consistent before passing it to the application program proper.

The application also uses the form to type information back to the user by passing these data to the **Cousin** server. The server then displays the program output in the proper format, as specified in the form description.

For additional details on interacting with a **Cousin** based application, see the *Cousin Users Manual* in the *Spice Users Manual.* If you wish to construct a **Cousin** interface to one of your own application programs, you must consult the *Cousin Application Builders Manual* in the *Spice Programmers Manual.*

CPP

KeyWords:	cpp, # define, # include, cc, c, macro, preprocessor
Function:	run the C language macro preprocessor
Syntax:	**cpp** <filename.c> <filename.1>

Description:

Cpp is the C programming language pre-processor. Using **Cpp** is the first step in preparing a C program to run on the PERQ workstation.

The first argument is the name of the C file to be processed and the second argument is the intermediate file created by **Cpp** (which is then used as input to the Pcc facility).

The pre-processor allows compiler directives such as # include and # define files to be handles prior to compilation.

Note:
In a future release **Cpp** and Pcc will be replaced by a CC (C Compiler) facility similar to the UNIX software cc facility.

See Also: CC

CUpdate

KeyWords:	Cupdate, update, transfer, move, save, archive, store, install, restore, retrieve, backup, dump, cousin
Function:	move a collection of files from or to a Vax host.
Syntax:	**Cupdate**

Description:

CUpdate is a Form-based version of **Update** using a COUSIN interface. Update commands are executed by filling in the appropriate spaces in the form and activating a command button. To see which fields need to be filled, the button corresponding to the desired command should be pressed (by mouse click) once. This will cause the fields which are parameters to the command to be highlighted with a heavy border. Once the parameter fields have been correctly filled, pressing the command button a second time will execute the command.

For details concerning the use of the COUSIN interface, consult the COUSIN Users Manual.

The following is a description of the fields in the Chili form:

Fields:

Host This field selects the remote Vax to transfer the files to and from.

Version Selects which version of the set of files to **Receive**. The value must be any of "old", "current", or "test".

LogicalName This field contains the logical name or pattern used by the update commands.

VaxDirectoryName
The value of this field is used by the **Enter** command and, along with **CommandFileName**, by the **Store** and **Receive** commands instead of a **LogicalName**.

LoginName, Password
These fields are used to specify the vax login and password, needed by most of the commands.

CurrentDirectory
the value of this field is expanded into a path name which is used as the directory to receive the files during a **Receive** and to supply the files during a **Store**.

Push | Overwrite Previous Versions
This field selects, during a **Store**, whether the new directory displaces the *Test* version or whether the *Test* version is moved to *Current* and **Current** to **Old**.

Transfers Enabled | Feedback Only
This field selects whether files can be transfered or the transfers are listed but not executed.

Terse | Verbose Feedback
This field selects the level of feedback given during command execution.

Ask | Dont Ask Questions
If this field is set to *Ask*, Update will ask the user some questions during the **Store** and **Retrieve** commands. If it is set to *Never Ask*, Update will choose the most non-destructive response.

Commands:

Retrieve	This command retrieves the set of files specified by the **LogicalName** or by the b[VaxDirectoryName] and **CommandFileName** fields.
Store	stores the set of files specified by the **LogicalName** or by the b[VaxDirectoryName] and **CommandFileName** fields.
Install	moves the *Test* version to *Current*. No files are transfered.
List	lists the Logical Names on the specified **Host** which match the pattern in **LogicalName**.
Kill	removes the **LogicalName** from the **Host**.
Enter	enters the **LogicalName** with the **VaxDirectoryName** on the **Host**.
Abdicate	disassociates the **LoginName** from the **LogicalName**.
Quit	exits the Cupdate program.

See Also: Update, Cousin

Debug

KeyWords: debug, adb, db, ddt, six12, pdb, sdb, kraut, pasddt

Function: invokes Pascal debugger.

Syntax: **Debug** [*Process*]

Description: **Debug** invokes the debugger which enables you to control, observe, and modify a process interactively. The debugger can also be invoked

1. through Sapphire by using the sapphire pop-up menus or by typing CTRL DEL followed by D (striking the Del key while depressing the Ctrl Key, then striking "**d**" or "**D**".)

2. automatically by the Process Manager when an uncaught exception is encountered.

3. by ending the command line that starts the execution of a process with a "[**-debug**]". In this case, the debugger will be entered just before starting the process.

For online documentation after you enter the debugger, type the "?" command.

For *Processes*, you may use the process name, the name in the Icon window, or the process number. The process name is usually the name used to invoke the program; any unique prefix will suffice. In specifying an Icon window name, all processes controlled by that window will be affected. The only way to affect a specific process, when its process name is not unique or the Icon window influences several processes, is to use the unique process number. The command "details - systat" shows the process name, unique process number, and controlling window name for every process.

Debug is an alias with the definition:

```
    Run Debugger.Run
```

The **Alias** command may be used to redefine **Debug** to use a different debugger. The default system has Kraut as *Debugger.Run* (see Kraut documentation in the Spice Programmer's Manual), although it is possible to use another debugger (Mace).

Note: This user-level debugger is totally distinct from the Accent kernel debugger.

See Also: Systat, Trap, Kraut Documentation, Sapphire Documentation.

Define

KeyWords:	define, environment, environment variables, variable, search list
Function:	define a simple environment variable or search list
Syntax:	**Define** *Name* { *NameSwitch* } [*Element* { *ElemSwitch* }] {, *Element* { */ElemSwitch* } }

Description: **Define** is used to set an *environment variable*. If **define** is invoked without any argument, it prints a help message.

Environment variables are used to communicate information between processes, typically between the shell and programs it runs. The *Environment Manager* serves as a repository for the *environment variables*.

There are two types of *environment variables*:

strings	for simple information.
search lists	for ordered lists of directories. These search lists are indicated by a name with a Trailing ":" in shell command lines which refer to them, viz. **current, default, run.**

Search lists are used to interpret filenames which are not specified as absolute pathnames, i.e. names not starting with '/'. In general, programs will either interpret the filename relative to **current** or look for the filename in each of the directories in **default** in turn until a file is found. Note: The interpretation of relative pathnames is handled by the individual programs and some variation from this model is possible. Note: **current** is a single element search list.

An *Environment variable's* scope effects how the variable is found and how the variable is shared. The scope values are:

Global	The variable is shared for read and write operations with all processes referencing the variable.
Local	All access to the variable is restricted to the process that created the variable. A child process can inherit a copy of all of its parent's local variables at the time that it is *spawned*. For example, when a shell creates a subshell, the subshell copies the parent's local variables. Write operations do not effect the parent's copy.

When a variable is created it must be given *Global* or *Local* scope. When the variable is referenced a third option is available: *Normal* "scope". This looks for a *Local* variable of the given name and if that fails looks for a *Global* variable.

It is possible to have both a local and global environment variable with the same name. In this case the local variable takes precedent just as in Pascal scope rules.

The evaluation of a *search list* produces a list of strings. These strings may contain imbedded *search lists*.

Define can be used to set either a simple *environment variable* or a *search list*. In the first case, *Name* is assigned the value *element*. *Name* must be a simple name without ":". '>'). The scope can be specified as *Local* or *Global*; it defaults to *Local*. *Element* can only be set to a single value -- not a list. Switch options, *-After*, *-Replace*, *-Resolve*, and *-Full* are not allowed.

If *Name* is a *search list*, two options exist: the search list may be assigned to or it may be modified by prepending or inserting new values into the old search list. Since the *Name* is a search list, it end with a ":". The scope is declared as with simple variables. The value of the *search list* is one or more *Elements*, each separated by a comma. An element may either be a simple string specifying an absolute pathname of a directory or a *search list*. At present, the scope of any referenced *search lists* must be same as the *search list* being defined. The referenced *search list* will appear as it was typed unless it is explicitly *Resolved* to force evaluation. Since the search list may contain more than one directory, it is necessary to specify whether the first value (using switch **-resolve**), or all the values (using switch **-full**) are to be include in the new search list. **-Full** implies each element is *resolved*.

An old *search list* can be updated. This is indicated by supplying either **-after** = *index* or **-replace** = *count* or both. The *Elements* supplied are evaluated as above. This list is spliced into the old search list after *index* and the next *count* elements of the original search list are deleted. The result replaces the old *environment variable*.

An Environment variable may be removed by setting its value to nothing. This can

be done either by define
<*var* define
var -replace = *count* when
count equals the number of values that
var currently has.

Switches:		
	-Local	create a local variable.
	-Global	create a global variable.
	-Normal	try to find a local variable and if it does not exist look for a global variable, but always create a local variable.
	-Resolve	fully evaluate the variable replacing imbedded *search lists* recursively but return only the first element of the resultant evaluation.
	-Full	fully evaluate the variable replacing imbedded *search lists* recursively and return all elements of the resultant evaluation.
	-After = *index*	specifies the position in the old search list where insertion or deletion (replacement) is to be done. The default value for *index* is 0.

-Replace = *count*	specifies the number of elements to be removed from the original list before any new items are inserted. The default value for *count* is 1.
-Help	prints a help message.

Examples:

Define current: /sys/user/blair
    same as path /sys/user/blair/

Define default: -repl -after = 1 foo
    same as sets -pop foo

Define sys:,current:,dev:User/System/,boot-System/
    New slist

Define default: -show
    Examine default searchlist

Define terminal PERQ
    A string-valued variable

Note: "Show options" is equivalent to "define options -show"

See Also: Details, SetSearch, Path

Delete

KeyWords:	delete, rm, remove, rmdir, del, expunge, purge
Function:	delete a file.
Syntax:	**Delete** *FileName* {,*FileName*}{ -*Switch*}

Description: **Delete** irrevocably destroys the specified file(s). It deletes the file name from the directory and places the blocks it occupied on the free list making those blocks available for use in new files. *FileName* may also refer to an empty directory.

Wildcards may be used in the file name proper, but not in the specification of the parent directory names. See the **Directory** command for a description of wildcards. If wildcards are used, **Delete** asks the user to confirm the deletion of each file individually unless the **-NoConfirm** switch is specified.

Delete only operates on the **current:** directory. Thus, it can not delete files in the **default:** search list unless you specify the path name explicitly.

Switches:

-Confirm	Request confirmation before deleting a file. **-Confirm** is the default if wildcards are used.
-NoConfirm	Overrides requests for confirmation before deleting a file. **-NoConfirm** is the default for filenames without wildcards.
-Help	Print a short summary of the syntax and switches.

Note: The **Delete** command offers a **-NoConfirm** switch; this is unlike **Copy** and **Rename** which offer both **-NoConfirm** and **-NoAsk** switches.

See Also: Copy, Rename, Directory, and *Introduction to the Spice User's Manual* section on PathNames, and Network PathNames.

Details

KeyWords:	details, environment, mount, status, time, information, systat	
Function:	provide system information	
Syntax:	**Details** {*-Switch* }	
Description:	**Details** provides information about the current state of your system. The default information includes the **Details** version number, system boot character, UserName, the partition information and a list of all the valid boot files.	
	Additional state information may be requested by the use of switches.	
Switches:	**-All**	Gives all of the information available with the other switches.
	-BootChar	Gives the boot character for the current system.
	-Boots	Gives all the valid boot characters and files.
	-DiskType	Gives the type of disk.
	-Ethernet	Gives the Ethernet address and machine name.
	-FreePage	Gives the size of the free paging space.
	-IOBoard	Gives the type of the I/O borad.
	-MachineName	Gives the name of the workstation (the name in the file SysName).
	-Monitor	Gives the type of monitor.
	-Names	Gives all names (machine, profole, etc.)
	-OopsKeyDown	Shows whether ignore-run-file is set.
	-Partition	Names of all devices and partitions known (includes the size and the number of free blocks in each partition).
	-Path	Gives the name of the system partition (**boot:**) and the current directory (**current:**).
	-Profile	Gives the name of the current profile file.
	-RS232Status	Gives the number of RS232 connections.

-Search	Prints the current search list (**default:**).
-SerialNumber	Gives the serial number of the workstation.
-ShellName	Gives the name of the current shell run file.
-Systat	Gives the status of all processes.

A status line has six fields:

1. The first field is a unique process number. (Note: the Process Manager's port number for the Kernel Port of the process being described.) This field may always be used for *Process* on any process control command.
2. The second field contains two subfields: 1) the process's privilege state:
   - S - All supervisory privileges.
   - P - Physical memory access capability.
   - U - No privileges -- user.

   2) the process's execution queue:
   - '# #' - two digit (0-15) run queue index.
   - ' P' - pending queue.
   - ' S' - sleeping queue.
   - ' O' - other.
3. The third field is the priority of the process. (0 is the lowest priority and 15 is the highest priority).
4. The fourth field is the elapsed run time, in seconds.
5. The fifth field is the icon name of the window.
6. And lastly, the sixth field is the complete name of the process. This field may be used for *Process* on any process control command as long as it is unique.

-Time	Gives current date and time.
-UserName	Gives the current user's name.
-Version	Gives the version numbers of the net server, process manager and Sapphire (the window manager).
-WCSSize	Gives the writable control store size.
-Help	Prints this list of parameters.

See Also: Define, Path, SetSearch, Show

Bugs: DiskType - returns IO board type, but disk type unknown
SerialNumber - returns 255 on lots of different Perqs.

Directory

KeyWords:	directory, dir, direct, ls, list, catalog, files
Function:	list files.
Syntax:	**Directory** [*FileSpec*] [[~]*Output File*] {-*Switch*[= *Value*]}

Description:

Directory lists files in a directory on your local machine or on a remote Perq machine. If **Directory** is invoked without specifying a *FileSpec*, then all files in the current directory will be listed.

Wildcard characters are permitted in the *FileSpec*. The wildcard characters allowed in file names by all the file utilities are:

*	matches 0 or more characters
?	matches exactly 1 character

Note: A wildcard character may be specified as a literal character in a filename by preceeding it with a backslash (\). For example, foo*.pas for filename foo*.pas.

Unlike most other commands, **Directory** even accepts wildcards in the directory part of *FileSpec*. There are, however, some anomalies in the code that handles wildcards in the directory part. For example,

```
     directory */foo
```

does a recursive list of all the files named **foo**, starting at the current directory and looking recursively in all the subdirectories below current. On the other hand, multiple wildcards in the directory part often don't match names that one would expect. If a logical name for a search list (such as **default:**) is used in *FileSpec*, then only the first directory on the search list in which some matching files are found will be searched. Note that wildcards cannot be used for matching the topmost (*Device*) name in absolute pathnames. That is, "**/*/User**" is illegal. On the other hand, "**/sys/***" works just fine (assuming there is a device named **/sys**).

If *FileSpec* ends with a "**/**", then all files in the matching directories will be listed instead of just the directory names themselves. Similar action is taken if *FileSpec* does not end with "**/**" but names an existing directory (without using wildcards). Thus, the interpretation of

```
     directory foo
```

depends on whether a directory named **foo** exists.

If an *Output File* is specified, then output goes to the indicated file instead of the display screen. Furthermore, **-OneColumn** is forced even if the user specified **-MultiColumn**.

Switches:	**-All**	List all available information about the files matched (name, size, creation date & time, access date & time, and access rights.).
	-Delimiter	List each filename twice, separated by "[|]". *This must be useful as input to some program but we have not figured out which one!*
	-Directories	List only directories, not other files.
	-Fast	List just the names of the files matched. **-Fast** is the default.
	-Help	Print a short description of this command.
	-ListDirectories	List the names of every directory that matched the directory part of *FileSpec* even if no files in that directory matched the rest of *FileSpec*.
	-MultiColumn	List four files per line. **-MultiColumn** is the default for a **-Fast** listing on the screen. Note that **-all**, **-size**, and **-Prot** cause a one column listing.
	-OneColumn	List only one file per line.
	-Partitions	Also list the mounted partitions and their sizes. (see Details)
	-Prot	Display the file owner's user name and user ID, and the access rights for the file.
	-Size	List the names and the sizes (first in blocks, then in bytes) of the files matched.
	-Sort = NoSort|Name|AccessDate|CreateDate|Size	Specifies which field should be used for sorting the files listed. The default setting is **-Sort = Name.**
Examples:	**Dir**	lists every file in the current directory.
	Direct */*	lists all the files in all the directories starting with the current directory and including all subdirectories.
	Direct boot:x*/*.run ~ run.list	looks in the boot partition for all the run files in directories whose names start with 'x' and writes all of these names into the file **run.list.**
	Dir program* -size	lists files beginning with **program** and tells how much disk space each occupies.

Note: CTRL ? may give you a faster list of all the files in the current directory. See the section on the **Shell** for a description of file name completion using CTRL ? or ESC. See also *Introduction to the Spice User's Manual* section on PathNames and Network PathNames, and the section on Automatic File Name Completion.

See Also: Details, DirTree

DirTree

KeyWords:	dirtree, directory, list	
Function:	Show the directory structure	
Syntax:	**DirTree** [*DirectoryName*] {*-Switch*}	

Description:

DirTree creates a picture of the directory structure of your machine. It erases the entire window and then draws the picture.

If you specify a device, partition or directory on the command line, **DirTree** will use that as the root of the tree that it draws. If you do not give a name on the command line, **DirTree** uses the current boot device as the root of its tree.

If there is no room for a directory at the right margin its parent is marked with a * after the **/**.

Switches:	**-Help**	Prints a help message.
	-Blocks	Counts all the blocks in each directory. For parent directories prints the total all all blocks used by the directory and all its sub-directories as well. Does not include overhead of file headers or directory blocks. This takes about 4 times longer than a regular **DirTree**.
Commands:	**H**	Lists all the commands and what they do.
	B	Recalculates directory tree with blocks.
	Q	Quits the program.
	R	Returns to original directory tree.
	U	Recalculates the directory tree.

Note: It is very slow because it does a lot of disk reading!

See Also: Path, Define and Directory.

Dismount

KeyWords:	dismount, remove, unload, unmount, umount
Function:	detach a partition from the filesystem.
Syntax:	**Dismount**
Description:	**Dismount** detaches a partition from the filesystem. **Dismount** will prompt for the partition name.
	Once a partition has been dismounted, the file on it can no longer be accessed by the filesystem.
Note:	The Mount command and the Dismount command are not related.
Bug:	There is no way to remount a partition once you have dismounted it. You must reboot your machine before you can access files on that partition again.

DP

KeyWords:	dp, draw, picture, graphic, plot
Function:	Drawing Program.
Syntax:	**DP** [[@]*FileName*]{-*Switch*[= *Value*]}

Description: DP is a program for editing illustrations and circuit diagrams. See the *DP Users Manual* in the **Spice Users Manual** for more information. In the command line,

@	indicates that *FileName* is a transcript file to be replayed.
FileName	is the name of a **.dp** file (extension optional) or of a transcript file (if preceded by the character "@")

Switches:

-Profile = *ProfileFileName*
ProfileFileName is a DP profile file.

-Help Print a very short help message on DP (there are however, extensive on-line help facilities inside DP).

See Also: Mint.

Edit

KeyWords:	edit, editor, change, ed, vi, visual, sos, lined, teco, flash
Function:	edit a file
Syntax:	**Edit** [*FileName*]
Description:	Edit is used to create or alter text files using Oil.

Three basic uses of oil are discussed briefly below: creating a new file, changing an existing file, and reading a file at leisure. Exiting the editor is also explained briefly. This utility is explained in detail in the document "Oil" in this manual.

If you omit the filename, Oil assumes that you want to edit the default file remembered by the shell. The **-Replay** switch runs a transcript of the last editing session. This recovers editing lost due to the system crashing or your exiting a file without saving it.

Note: See **Oil** for information about session transcripts and transcript replaying.

Also oil.keytran defines the command character interpretation. Typically you use oil.emacs.keytran but oil.flash.keytran is also available.

See Also: Compile, Oil, Type, Alias.

ExpandTabs

KeyWords:	ExpandTabs, format, tabs
Function:	Replace tabs with spaces keeping the same indentation
Syntax:	**ExpandTabs** <SourceFile <DestinationFile> [-Help] Description: ExpandTabs replaces tabs in the input file with the correct number of spaces. ExpandTabs assumes tabstops every 8 columns. ExpandTabs is used when the input file was written for another system and put onto a PERQ workstation, which does not support tabs. THE ONLY SWITCH IS **-HELP**, WHICH DISPLAYS INFORMATION ABOUT EXPANDTABS.
Notes:	The SourceFile and DestinationFile must be different

FindString

KeyWords: findstring, grep, search, egrep, fgrep, scan

Function: Search one or more files for a string.

Syntax: **FindString** *string*, *filelist* {*-swich*} *[~outputfile]*

Description: The **FindString** command searches through a number of files for a particular string. The command operates in two modes: context and nocontext. In context mode (the default), **FindString** prints the line number and the leading and trailing characters for each occurrence of the specified string. In nocontext mode, **FindString** prints only the word *Match* when it reaches the first occurrence of the specified string. You specify the mode with the **-CONTEXT** or **-NOCONTEXT** switches.

By default, case is not significant; you can force **FindString** to match case exactly by specifying the **-CaseSensitive** switch.

The first argument to **FindString** is the string to search for. To include a non-alphanumeric character, either precede it with a backslash (\) or surround it with quotes. The next argument is the file to search. You cannot list more than one file but you can specify a wildcard. If you wish, you can direct **FindString** to write the occurrence(s) to a file by specifying an output file.

Switches:

CaseSensitive This switch specifies that case is significant (for example, if you specify the switch and the string to search for is XYZ; **FindString** does not view XYZ as a match.

-NoCaseSensitive
This switch specifies that case is not significant; **FindString** ignores upper and lower case. **-NoCaseSensitive** is the default.

-Context This switch directs **FindString** to list leading and trailing characters for each occurrence of the string. **-Context** is the default.

-NoContext This switch directs **FindString** only to notify you when it finds the first match, if any.

-Help This switch displays a description of the **FindString** command and the associated switches. Note that **-Help** does not search for string occurrences.

Examples: **FindString** screen, boot:os/*.pas~screen.users -nocontext.

This command directs **FindString** to search all files with a .Pas extension in the OS directory of the Boot partition for an occurrence of the string screen. **FindString** writes the output to the file "screen.users".

Help

KeyWords:	help, assist, man, key, doc, ?, Argh!, oops, information, socorro
Function:	display help messages.
Syntax:	**Help** [*Word*|*Phrase*].

Description:

Help Displays information about commands related to the word or phrase provided as an argument by the user. If the **Help** command is issued without an argument, the user will be given a message about how to use the **Help** command.

If a word or phrase is supplied as an argument, a list of all commands deemed to be relevant is displayed together with a brief description of each one. The user is then prompted to specify for which command he wants more detailed information. He may specify one of those commands, provide another word or phrase, or exit the help command.

Examples:

help Details or **Help Det**
will print the manual entry for the Details command.

help environment variables
will print the manual entry containing the phrase **environment variables** as a keyword (i.e. the **Define** command.)

Hemlock

KeyWords:	hemlock, emacs, edit, ed, vi, visual, sos, lined, teco, flash
Function:	Hemlock is an Emacs-like screen editor written in Lisp. Hemlock is designed for easy extension and customization by users.
Syntax:	**lisp -edit [file] From the shell, or From Lisp (ed ["file"])**
Description:	Hemlock is built into the standard Spice Lisp core image. See the description of Spice Lisp for instructions on retrieving and running this system. To run Hemlock from the shell, invoke Lisp with the -edit switch and optional file name. To run Hemlock from Lisp, call the ed function with an optional file name. To exit back to Lisp, just type CTRL C. To resume editing where you left off, type (ed) again to Lisp. To terminate the Hemlock/Lisp process, type (quit) to Lisp. You probably don't really want to kill the Hemlock process anyway -- see below. Like Emacs, Hemlock is extensible and largely self-documenting. Press the HELP key to ask what any command does, what any key is bound to, etc. For in-depth documentation, see The Hemlock User's Guide by Rob Maclachlan. Hemlock's command set is very close to that of TOPS-20/ITS Emacs. Users of Goslings "Emacs" for the Vax may find some differences in key bindings. The extension language for Hemlock is Common Lisp. See The Hemlock Command Implementor's Manual by Rob Maclachlan and Skef Wholey for details on customizing or extending Hemlock.
See Also:	Lisp, Hemlock
Note:	Both Lisp and Hemlock take a while to start up on a 1 Mbyte Perq, so most users will prefer to create a Hemlock job in one Sapphire window and keep it around for the duration of a session. Hemlock will run noticeably faster if its screen is completely uncovered.
KeyWords:	IconWallclock, clock, time

KeyTranCom

KeyWords:	keytrancom, keytran, keytext, ktran, ktext, keybinding
Function:	compiles key translation files.
Syntax:	**KeyTranCom** [*FileName*]
Description:	Creates a "Key Translation File" (**.keytran**) file from a **.ktext** source file.

A key translation file is a mapping of keyboard keys or key sequences to commands accepted by an application program. Examples of programs using this facility include the Shell (technically, the **TypeScript** process), DP, Oil, and Chat

For additional details and examples see the corresponding Appendix in the *Introduction to the Spice User's Manual* and the *Window Manager* in the Spice Programming Manual.

See Also: Sapphire documentation.

Kill

KeyWords: kill, stop

Function: terminate a process.

Syntax: **Kill** [*Process*]

Description: Kill terminates the specified process.

Sapphire will suspend the listener process when CTRL DEL is typed. The Sapphire symbol is displayed while Sapphire waits for a command. If a "k" is typed, the process is killed.

For *Process*, you may use the process name, the name in the Icon window, or the process number. The process name is usually the name used to invoke the program; any unique prefix will suffice. In specifying an Icon window name, all processes controlled by that window will be affected. The only way to affect a specific process, when its process name is not unique or the Icon window influences several processes, is to use the unique process number. The command "details - systat" shows the process name, unique process number, and controlling window name for every process.

If Kill is invoked without an argument. the user will be prompted for a process.

See Also: Alias, Priority, Resume, Suspend, Systat, Sapphire documentation.

Launch

KeyWords: launch, shell, startup, initialization

Function: Program to start other programs

Syntax: **Launch** *FileName* {*-Switch*}

Description: **Launch** allows a user to startup programs in specific windows. Launch reads the file name given for commands to create windows and spawn programs (.run files). Launch takes each line in the file (long lines can be split with a backslash character as the last character of the split lines), creates a window according to the *window descriptor* and spawns the process according to the *shell command line*. The syntax of Launch file lines is given by the following informal BNF with the additional provision that the items in a list of name-value pairs are separated by commas:

```
<Launch-Command-Line>:== '[' <Window-Descriptor> ']'
            <Shell-Command-Line>

<Window-Descriptor>:== {<Global-Switches>} <Window-Switches>

<Global-Switches>:== ABSOLUTE | LISTENER | NOBORDER |
          NOCLIP | NOICON | NOTITLES
          REMOVE | RANK = <integer>
          SCREENLEFTX = <integer> |
          SCREENTOPY = <integer> |
          SCREENHEIGHT = <integer> |
          SCREENWIDTH = <integer>

<Window-Switches> : = = <Ask-User> | { <Use-Map-File> } |
          { <Use-Rectangle> } | <Use-Quadrant>

<Ask-User> : = = <nil>

<Use-Map-File> : = =  MAPFILE = <filename> |  INDEX = <character>

<Use-Rectangle> : = =  LEFTX = <integer> | TOPY = <integer> |
          WIDTH = <integer> | HEIGHT = <integer>

<Use-Quadrant> : = = FULLSCREEN | UPPERHALF | LOWERHALF |
         RIGHTHALF | LEFTHALF |
         LEFTUPPERQUANDRANT | LEFTLOWERQUADRANT |
         RIGHTUPPERQUADRANT | RIGHTLOWERQUADRANT
```

Note that Launch is *not* the Shell program and so cannot perform directory listings or file deletions or any other function built into the Spice Shell.

The <Global-Switches> should appear first in the file as some of them change the value of other constants. The <Global-Switches> have the following meanings:

ABSOLUTE	specifies that the tracking in the new window is to be absolute; the default is relative tracking.
LISTENER	makes the new window be the listener.
NOBORDER	this creates the window with NO borders; the default is to create the borders.
NOCLIP	this specifies that the window will not be clipped against the physical screen; the default is to clip.
NOICON	this creates the window with NO icon in the icon window; the default is to give the window an icon.
NOTITLE	this creates the window with NO title line; the default is to create a title line for the window.
REMOVE	this removes the window from the screen area after creating it; this is the same as clicking the middle mouse button in the middle of the title line; the default is NOT to do this window.
RANK	this specifies the rank (or Z-axis) for the new; rank 1 is the top of the screen, rank 2 just below that, rank 3 below that, etc.; zero or negative rank puts the window on the bottom of the stack of windows.
SCREENLEFTX	this specifies a value for the default LEFTX; if not given, the default value is 0.
SCREENTOPY	this specifies a value for the default TOPY; if not given, the default value is 0.
SCREENWIDTH	this specifies a value for the default WIDTH; if not given, the default value is the width of the screen.
SCREENHEIGHT	this specifies a value for the default HEIGHT; if not given, the default value is the height of the screen minus 100 pixels so as to not cover the icon window.

The <Window-Switches> above are mutually exclusive. If you have no window switches, Launch will ask the user to specify the window. (This is equivalent to the mechanism in Sapphire called 'Reshape Window'.)

Otherwise, if you wish to use a map-file, include one or both of the map-file switches. The default window map file is *default.wmp*. The default index character is *1*. The map file is a file of characters that specifies a proportional mapping of windows onto the screen. For instance, suppose the map file contains the following:

```
1...
....
2..1
.2..
```

Using index *1* will give you a window three-quarters of the height of the screen and full width. Using index *2* will give you the lower left quarter of the screen for the window. Note that these two windows will overlap. The index specifies what character to look for in the map file to find the *corners* of the window. Space ( ) or period (.) can be used as a filler in the file. Only one instance of the character need be present. For instance, the following map file effectively divides the screen into three horizontal bands of equal height:

```
1
2
3
```

If you wish to specify the shape of the rectangle precisely, use at least one of the rectangle switches. The default values of rectangle switches are:

- LEFTX defaults to the value of the SCREENLEFTX global switch; if that is missing, then it defaults to 0
- TOPY defaults to the value of the SCREENTOPY global switch; if that is missing, then it defaults to 0
- WIDTH defaults to the value of the SCREENWIDTH global switch; if that is missing, then it defaults to the width of the screen (portrait screen width, actually)
- HEIGHT defaults to the value of the SCREENHEIGHT global switch; if that is missing, then it defaults to the height of the screen (minus 100 pixels to leave room for the icon window).

If you wish to use the quadrant descriptors, include one of the quadrant keywords. These parameters perform the obvious mapping to values for LEFTX, TOPY, WIDTH and HEIGHT by using the values of SCREENLEFTX, SCREENTOPY, SCREENWIDTH and SCREENHEIGHT respectively for the default full screen size.

Switches:	**-Quiet**	This switch disables all but critical error messages from appearing on the screen.
	-help	Prints a help message.

Examples: A simple launch command file that will start the editor and a new shell which will be the listener is as follows:

```
[height=975]editor
[upperhalf]shell -listener
```

The following launch command file will start a Mailman process with a window in the top half of the screen, which then is covered with a window running Mercury. **Speak** and **Listen** programs are started in windows that use the top half of the screen horizontally divided. The **Editor** is given a large window covering all of the screen except the icons and then a new **Shell** is started using most of the lower part of the screen.

```
[upperhalf,rank=0]mailman -nologin
[upperhalf]<mail>mercury
[mapfile=quarters.wmp,index=1,rank=0]listen
[mapfile=quarters.wmp,index=2]speak
[height=975]editor
[topy=140,height=670]shell.s5
```

The file **quarters.wmp** consists of the lines:

```
1
2
3
4
```

Notes: One day this will be in the Shell for real....

See Also: Shell

Bugs: The switches are processed together, so you should include any global switches *first* for the others to work properly.

Link

KeyWords:	link, load, make
Function:	produce a runnable file from Pascal generated .seg files.
Syntax:	**Link** [-*GlobalSwitch*] *ProgramName* {,*ImportedModule*{*LocalSwitch*[= *Value*]}} [[~]*RunFileName*][-*GlobalSwitch*]

Description:

Link takes Pascal-generated **.seg** files (or library **.run** files) as its input and produces a runnable file with a **.run** extension. This file is created by linking together all the separately compiled modules that make up a program. The first file is normally the main program. The files imported by the main program will be added to the runfile. If other import files are specified on the command line, they replace the default imports.

If a *RunFileName* is specified that will be the name of the output run file, otherwise *ProgramName*.**run** is used as the name for the **.run** file.

The **.run** file may contain pointers to **.seg** files rather than their contents (if the -NoInclude switch is used). In this case, a program should be relinked whenever any of its **.seg** files is **Renamed**. Also, you can copy these run files on your machine, but not from one machine to another. The default is to include all the .seg files in the **.run** file. These run files may be copied from one machine to another. Any run files to be entered in the standard libraries on the Spice or CFS Vaxes must not be linked with the **-NoInclude** switch.

Switches:

-Data	Local switch. Indicates that the **.seg** file that precedes it is a data file. Data files are always included in the **.run** file.
-ForceLoad	Global switch. Causes all files on the command line to be loaded, whether they are referenced or not. Without this switch, only **.seg** files that are referenced are loaded. This switch applies to **.run** files only if a local **-Include** switch is specified for the particular **.run** file.
-Help	Global Switch. Print a help message.
-Include	Global or Local switch. Causes the bodies of all **.seg** files except those which came from a **.run** file to be copied into the output **.run** file. Individual files, including **.run** files, can be included using a local **-Include** switch. **-Include** is the default.
-Library = *name*	Global switch. Specifies a library **.run** file. This is equivalent to appending *name*.**run** to the list of input files.

-MakeLibrary	Global switch. Builds a library out of all the **.seg** files on the command line includeing all that are referenced.
-Main	Local switch. Specifies that one of the inputs other than the first contains the main program. This is used primarily when "re-linking" an existing **.run** file with some replacement modules.
-Map	Global switch. Generates a map file of the **.run** file giving the offsets at which the various **.seg** files begin.
-NoDefaultLibrary	Global switch. Unless this switch is used the library **LibPascalInit.run** is linked with your program.
-NoInclude	Global switch. Does not copy the bodies of the **.seg** files into the output **.run** file.
-NoInitialization	Global switch. Unless this switch is used the module **PascalInit** is made the initial entry point of the **.run** file. This module performs process initialization and then invokes the user main program, *ProgramName*. **PascalInit** is included in the default library.
-Quiet	Prevents the display of procedure and function names during linkage. This is the default; it may be disabled by specifying **-Verbose**.
-Relink	Global switch. Takes old **.run** files and relinks them with newer version of libraries. The libraries are listed first in the command line with the program to be relinked at the end.
-SymInclude	Global switch. Causes the **.Syms** and **.Qmap** files, if they exist, to be copied into the **.run** file directly, making the debug information present in the address space of the process when it runs. This is faster than using **-SymReference**, but causes the run file to be much larger.
-SymReference	Global switch. Causes the **.Syms** and **.QMap** files, if they exist, to be referenced in the **.run** file causing them to be present in the address space of the process when it runs. This is slow, but keeps the **.run** file small.
-System	Global switch. Builds **.run** files to be used by MakeVMBoot to build Accent boot files.
-Verbose	Global switch. Causes **Link** to print out the names of the files it is linking as it goes along.

Examples:

link test
: builds Test.RUN from Test.SEG and .SEG files it imports

link -incl test,x,y,util.run
: builds Test.RUN which includes x.SEG and y.SEG, and references util.run for utilities

link -lib = LibPas test,d1-data ~main
: builds main.RUN which links to test.SEG, references library LibPas.RUN, and includes data file "d1"

link -forceload test,oldmain.RUN -main -include ~newmain
: builds newmain.RUN as a copy of oldmain.RUN, substituting test.seg for the module test in oldmain. The ordering of the inputs is important; the opposite order would use the module test in oldmain, not the replacement in test.seg.

link -relink libpascalinit.run, calc.run -main
: rebuilds **calc.run** from an existing version, relinking it with **libpascalinit.run** (the standard driver for Pascal programs). You might need to do this if the current version of **calc.run** was linked with a version of **libpascalinit.run** that is different from what you have now.

link -map prog,seg1,seg2,...,run1,run2,...,data1,... ~runfile
: builds file "runfile.RUN" (rather than the default "prog.RUN"), which includes prog.SEG, seg1.SEG, seg2.SEG, ... then resolves any files imported by them, by first looking in run1.RUN, ... and then by doing file name lookups.

See Also: Make

Lisp

KeyWords:	lisp, common lisp, spice lisp, languages
Function:	The Lisp command invokes the Spice Lisp interpreter.
Syntax:	**Lisp**
Description:	Spice Lisp is an implementation of the new Common Lisp dialect for the Perq. Lisp is the language of choice for most AI programming, and is also useful for other programming of an interactive or exploratory nature. The Spice Lisp system includes extensive debugging aids, online documentation, the Hemlock text editor, and a compiler for individual functions or entire files of Lisp code.
Notes:	In order to run Spice Lisp, you must have a Perq 2 or a Perq 1 with a 16K control store, and you must have an up to date Accent system. Decide where you want the Spice Lisp files to live. There must be close to 6500 pages free in the partition in which you wish to put Spice Lisp. It is suggested that you make a subdirectory called slisp in the user partition. Set your path to the directory in which you want to put Spice Lisp, then run the update program on Slisp - A. When Spice Lisp is on your Perq, put the directory that it resides in on your search list and then just type lisp to the Accent shell.
See Also:	The entry in this manual for Hemlock, an Emacs-like text editor written in Spice Lisp. For information on the Common Lisp Language, see Common Lisp: The Language by Guy L. Steele, Jr. For additional information on the Spice/Perq implementation, see the Spice Lisp User's Guide. Frequent users of any Common Lisp implemenation at CMU should watch the CLISP bulletin board for announcements. Up-to-date status information on Spice Lisp can be found in the files on CMU-CS-SPICE in directory /usr/slisp/docs.

Listen

KeyWords:	listen, speak, send, rsend, finger, who, wall, broadcast
Function:	Program for receiving short messages from Speak, and Spice finger service.
Syntax:	**Listen** [*UserName*] {-*Switch*[= *value*]}

Description:

This program accepts strings sent to its listener port and prints them on its window (and optionally to a file). The format of this message is available if you have a need for remote message delivery.

UserName is the name by which you will be known to anyone who is trying to send you a message using the **Speak** command. If you invoke **Listen** without a *UserName*, you will be known by your **MachineName** which is an Environment Variable that is currently set to whatever name is found in **boot:SysName.**

Switches:

-Transcript = *FileName*
This causes Listen to take a transcript of everything it types to the user. The transcript is written to *FileName*.

-Help Prints a help message.

Notes:

BE PATIENT! Some of the mechanisms used to communicate over the network fail and time-out. Let them! The internal mechanism for finding other people on the network may change (for the better) if and when various bugs related to ports are found and fixed. However, in the meantime the mechanism in use works, though slower than it should.

See Also: Speak

Bugs:

There seems to be a bug with the first request for who service - it sometimes does not respond until your second or third attempt. This is actually the time it takes the Listen program to establish itself with the nameserver. BE PATIENT!

Lnk

KeyWords:	lnk, linker, c, cc, spoonix
Function:	**Lnk** is a linker for C on the Perq: It takes the output from asm and previous runs on **Lnk** to form either a library file or executable file.
Syntax:	**lnk** {*-switch* [*value*]} *InputFile* {{*-switch*} *InputFile*}

Description:

Lnk takes .o files produced by the assembler and puts them together into executable files. **Lnk** supports a simple library mechanism. The output of any run of **Lnk** can be used as a library for subsequent runs of **Lnk**. **Lnk** libraries are not like ar libraries on the Vax. **Lnk** libraries are made up of .o files and references to other libraries. Libraries are not allowed to have unresolved symbols in them, so all the .o files that go into a library must be self-contained, or, the external symbols must be resolvable with another pre-existing library. You cannot replace a part of a library except by rebuilding the library from scratch.

Lnk allows you to have a symbol defined both in a library and in a .o file. The .o file version takes precedence. However, since the library is already linked, and all symbol references internal to it are already resolved, any reference to said symbol by something in the library will have been resolved to the library version of the symbol.

Libraries are quick to load, so process startup does not suffer due to the use of libraries. A single copy of the library code is shared by all the programs that use it in the same manner as LibPascalInit.run is shared by Pascal programs.

Switches:

-o *outputfile*	indicates that the next argument is the name of the output file.
-m	indicates a library is being linked.
@indirectfile	reads a list of files to be loaded from "indirectfile". Each line of the file may contain the name of a file to load, another "@indirectfile" command or a comment. Comments begin with a exclamation point(!) in column 1. Blank lines are ignored. NOTE: You may NOT put switches into indirect files. The lines in the file are taken verbatim. No stripping of leading or trailing blanks is done.
-g *routine*	indicates that the program is to start by calling the routine who's name is the next argument. If this switch is not used, the routine name defaults to "main".
-s *routine*	indicates process is to start by calling the routine whose name is the next argument. If this switch is not used, the routine name defaults to "crt0". This switch exists for the convenience of gurus, use at your own risk.

-x	means do not preserve local symbols in the output symbol table. This saves a little space in the output file, but makes debugging harder.
-L	means to save all local symbols in the output symbol table, even those starting with "L" and "'", which are usually compiler generated temporary symbols. Since the compiler doesn't put underscores in front of user's symbols, it is possible that a user could have variable or routine names that start with "L" and as such it would be discarded from the symbol table.
-map	produce a memory map. (not implemented yet)

The following switches are only for debugging the linker.

-dfu	turn on debugging output for fixups.
-dplc	turn on debugging output for placement.
-dsym	turn on debugging output for symbols.
-dif	turn on debugging output for input files.
-dof	turn on debugging output for output file.
-hash	show hash table statistics.
-HASH	run only long enough to test hash table. turns on -hash.
-dall	turn on all debugging output.
-help	this message (in a different format).

Notice that there isn't a "library file" switch. This is because **Lnk** can determine if a file is a library file by looking at its contents, so it doesn't need a special switch.

Examples: A couple of examples should make things a little clearer:

```
lnk -o foo.exe libc.exe foo.o
```
link foo.o with reference to library libc.exe into foo.exe.

```
lnk -m -o libxx.exe @libfiles.lmd
```
link library libxx.exe from the files listed in libfiles.lmd

```
lnk -o bar.exe bar1.o bar2.o bar3.o libbar.exe
```
link bar.exe from bar1.o, bar2.o, bar3.o and library libbar.exe

```
lnk -o bartest.exe bar.exe bartest.o
```

link bartest.exe from bartest.o and previously linked program bar.exe

Notes: Putting libraries in the front of the command line will make things go a little faster.

If a symbol is defined in both a library and a .o file, the .o file definition takes precedence. This means that you can have your own version of a routine that is also in a library. However, any references to that routine by other routines in the library will be to the routine in the library and not to your new routine.

The output file from a **Lnk** run can always be used as a library. You will get warnings about main being redefined if you do but that doesn't hurt anything, its just to let you know what is going on.

A word of warning, libraries can reference other libraries. Since the linker assigns to all output files the memory to be used when running the file, it is possible (due to the tree nature of library dependencies) to have library A reference both libraries B and C which are self-contained and linked seperately. Most likely B and C will overlap and you'll get a fatal error from the linker. The appropriate solution to this problem is not blatantly obvious. However, since it isn't much of a problem it has just been ignored. If people start having problems with library overlap, I'll have to go in and fix it.

See Also: Asm, CC, the Spoonix manual.

Login

KeyWords:	login, logon, attach, connect
Function:	Establishes a UserName and UserID for the session.
Syntax:	**Login** [*UserName*]

Description:

The **Login** command initiates a session at a PERQ workstation. **Login** should be called as part of the profile initialization at boot time.

After you type **login** the system propts for your password. **Login** then searches the System.Users file on the authentication workstation and validates the UserName and password. If the validation succeeds you will be known to the filesystem by your UserID.

If the **Login** failed, you will be known to the filesystem as **NoUser** and will have ownership rights to all files on your local workstation, and world rights to files on remote workstations.

Mailman

KeyWords:	mailman, mercury
Function:	Provides mail delivery service to and from Perqs
Syntax:	**Mailman** {-*Switch*[= *Value*]}
Description:	This program provides a mail delivery service to and from Perqs. The Mercury mail program (described elsewhere) allows the user to read, compose and organize his mail, but provides no mechanism for getting the mail off of or onto the Perq. This program manages incoming and outgoing mail on one Perq for one or more users.

Switches:

-log = *filename*	This will keep a log of mail activities in the given filename
-nologin	This disables all functions that require logging in to a Vax (currently, this disables retrieving mail from the Vax)
-postofficename = *name*	This sets the postoffice name to be that given. The default is MFS-CMU-CS-SPICE. In general, MFS-<machine name> is used to pick up mail from and deliver it to VAX <machine name>.
-nogetmail	This disables all retrieval of mail, from the Vax or from other Perqs
-nosendmail	This disables all sending of mail, to the Vax or to other Perqs
-vaxaccount = *account*	This sets the Vax account name to that given; you must have a Vax account name to get mail, even if you do not have a Vax account - this name is used to pick up mail from other Perqs; if you do not give this switch, you will be prompted at the terminal for the value; the default if you just type RETURN is the value of the global environment variable **MachineName**
-novax	This disables all operations to the Vax
-mailinbox = *filename*	This sets the name of the file to write incoming mail for this user (default name is **perqinbox**)
-help	Prints a help message.

Notes: When running more than one mailman on one Perq, you should run all but one with the **-nosendmail** switch. That is, only one mailman should be responsible for shipping all mail from the Perq.

See Also: Mercury

Make

KeyWords:	make, makefile, mic
Function:	maintains computer program groups.
Syntax:	**Make** [-*Switch*[-*value*] [*TargetFileName*]

Description: This program is Spice's version of Unix Make. It is a program to help create or maintain computer programs that are comprised of many separate modules.

Make executes commands in the *makefile* to update one target name. The target name typically refers to a program. If no -f switch is present, the file **MakeFile** is used. More than one -f switch may appear. If *TargetFileName* is not specified, the first target name that appears in the *makefile* is used.

Make updates a target if it depends on prerequisite files that have been modified since the target was last modified, or if the target does not exist.

The *makefile* contains a sequence of entries that specify dependencies. The first line of an entry is a single target, then a colon, then a list of prerequisite files. All the following lines that begin with spaces, are shell commands to be executed to update the target. Each shell command must appear on a separate line. These commands are printed as they are executed unless the switch -s is used. If a name appears on the left of more than one 'colon' line, then it depends on all of the names on the right of the colon on those lines, but only one command sequence may be specified for it.

Sharp ('#') and newline surround comments.

The following *makefile* says that 'pgm.run' depends on two files 'a.seg' and 'b.seg', and that they in turn depend on '.pas' files and a common file 'incl.pas'.

```
pgm.run: a.seg b.seg
         link pgm
a.seg: incl.pas a.pas
         comp a
b.seg: incl.pas b.pas
         comp b
```

Currently, only one target may be listed on a line at a time, though it is possible to have multiple targets if a dummy name is given with no succeeding commands, e.g.

```
allfiles: file1 file2 file3
file1: ...
     commands for file1
file2: ...
     etc.
```

Switches:	**-f** = *MakeFileName*	use the named file as the input *makefile*.
	-n	Trace and print, but do not execute the commands needed to update the targets.
	-s	Supresses the printing of each shell command line as it is executed.

Notes: **Make** accepts lines of up to 255 characters. After that it prints a warning and truncates the line.

See also: S. I. Feldman *Make - A Program for Maintaining Computer Programs*

MakeDir

KeyWords:	makedir, mkdir, create, built, create-directory, new-directory, catalog
Function:	create a directory.
Syntax:	**MakeDir** *DirectoryName* [*-switch*]

Description:

MakeDir creates a new empty directory. If you do not specify a directory name, you will be prompted for one. **MakeDir** will not accept as its parameter the name of any existing directory name; if you pass it such a name it will print an error message.

DirectoryName may be a pathname, in which case all the directories, except the final one, must already exist. If *DirectoryName* is a relative pathname (i.e. does not start with a '/'), the directory is created in the **current:** directory.

Switch:	**-Help**	prints a brief help message

Matchmaker

KeyWords:	matchmaker, interface, ipc
Function:	Accent IPC interface generator.
Syntax:	**Matchmaker** *DefFile* {*-switch* [= *value*]}

Description:

Matchmaker generates programs in a target language that allow client processes to invoke operations on server processes by remote procedure calls rather than having to directly use the Accent IPC mechanism to transmit parameters and return results. Matchmaker hides most of the details of Accent IPC from the programmer by packing parameters into messages, sending the messages to server processes, waiting for a reply, unpacking the reply message, and finally returning values to the caller.

For the server side of the interface, Matchmaker generates a procedure that takes a pointer to a message, unpacks its parameters, and calls the corresponding function. Results are then packed into a reply message.

The argument *DefFile* must define the desired remote procedure call interface, as well as some details about the types to be passed in the messages and the style of error handling desired. Running **Matchmaker** produces two (or three in the case of C) program source files. For a complete description of the *DefFile* input see *Matchmaker: A Remote Procedure Call Generator* in the *Spice Programmers' Manual*.

Switches:

-AccentLisp = *opt*	Generate code in Accent Lisp.
-C = *opt*	Generate code in C.
-Pascal = *opt*	Generate code in Pascal.

opt values are:

All	Generate all files in the specified language.
Defs	Generate the Defs file in the specified language.
User	Generate the User file in the specified language.
Server	Generate the Server file in the specified language.
NoUser	Generate all but the user file in the specified language.

NoServer	Generate all but the Server file in the specified language.
-Help	Prints a brief help message, giving usage and what languages are currently available. Watch this message for news of LISP code generation.
-Quiet	Does not display progress information about Matchmaker.
-Verbose	Displays progress information about Matchmaker.

Note: Matchmaker is now written in Lisp - so must be loaded as a lisp function.

Mercury

KeyWords:	mercury, mail, hg, send, rdmail, post, post-office
Function:	mail program.
Syntax:	**Mercury** [*MessageFileName*].

Description: Mercury is a program for handling electronic mail. It includes facilities for maintaining mailboxes, generating replies, forwarding messages, and classifying old messages.

If no *MessageFileName* (i.e. mail box) is specified, **Mercury** uses **default:Mail.Hg.**

Mercury is very similar to RdMail, the mail facility running on CMU-CS-A. For additional information, see the *Mercury User's Manual*.

Note: Strictly speaking, **Mercury** only handles your local mail box, it allows you to examine, classify, and delete old messages; it also reads messages that have arrived to your "in-box", and puts your replies and outgoing messages in your "out-box". Loading your in-box from a remote post office and outloading your out-box into a remote post office is done by a separate process, **Mailman.**

See Also: Mailman

Mint

KeyWords:	mint, scribe, runoff, xoff, pub, troff, tex, ms, mm, mroff
Function:	Document Formatter.
Syntax:	**Mint**
Description:	**Mint** formats documents for the PERQ screen or the Dover printer. It interprets a Scribe-like language, as well as Plot, and DP files. **Mint** prompts the user for the information it needs. See *User Manual for Mint - The Spice Document Preparation System* and *Mint Reference Manual* for more information.
Note:	Plot is a vector drawing program written by Ivor Durham. In executes on the TOPS-10 systems.
See Also:	DP

Monitor

KeyWords:	monitor, performance, systat, partitions, system
Function:	Dynamically displays system-related quantities.
Syntax:	**Monitor** {*-Switch*[= *Value*]}

Description:

This program provides dynamic monitoring of several system-related quantities, such as page usage, disk partitions status, number of basic kernel operations, and process information.

The information is displayed as bargraphs or numbers, and the rate of redisplay is selectable by the user. Options fall into two categories, named *fast* and *slow*; the redisplay rate for the two categories is individually selectable.

The following types of information are available through Monitor:

Boot:	static, boot-time information about the system configuration. Displayed as text, it never changes during execution. Default is OFF.
Pages:	percentages of physical memory pages usage. A page can be marked as *Locked* (non-pageable; for instance, parts of the Accent kernel and the display memory are Locked); *Dirty*; *Used*; and *Mapped To Disk*. The *User Time* (percentage of time spent in user processes) is also displayed. This group is displayed as bargraphs with numeric percentages; the redisplay rate is controlled by *-fastsleep*; default is ON.
Partition:	names of the disk partitions and number of free blocks in each partition. A bargraph indicates the percentage of free blocks and a number indicates the number of free blocks in each partition. Redisplay: *-slowsleep*; default is ON.
System:	number of basic Accent kernel operations per second; for instance, number of Procedure Calls or Read Faults per second. Also, the number of pages of virtual memory and the total number of active processes is displayed. The information is shown as numeric values. Redisplay: *-fastsleep*; default is ON.
Processes:	Status, name, priority, and runtime of the active user processes. The fields have the same meaning as in the *-systat* option of the **Details** program; a bargraph indicates the percentage of time taken by the process in the previous time slot. Redisplay: *-slowsleep*; default is OFF.

	Map:	Map of physical memory pages. Each page is shown as three bits aligned vertically, so that the map is composed of three parallel horizontal tracks. The topmost bit is black if the page is Locked; the middle bit is black if the page is Dirty; and the bottom bit is black if the page is Used. Redisplay: *-slowsleep*; default is OFF.

Switches:

-map Displays a map of physical memory pages.

-boot Displays boot-time information. This is static information about the system configuration, as recorded in the disk Boot Information Block. The following information is printed:

- WCS: size of the Perq Writable Control Store (typically 16 KWords).
- Mem: size of the physical memory, in pages.
- VMem: size of the virtual memory, in pages.
- Type of Z80 firmware.
- Type of IO board.
- Current boot character.

Warning: some of these fields are only hints for the system at boot time, and may thus not be completely accurate on some machines. For instance, the size of the Paging partition may actually be bigger than shown in the VMem field.

-fastsleep = *seconds*
Sets the time to wait between redisplays of "fast" options. Default value is 3 sec.

-slowsleep = *seconds*
Sets the time to wait between redisplays of "slow" options. Default value is 30 sec.

-nopart Do not display Partitions information, which is on by default.

-nopages Do not display Pages information.

-nosystem Do not display System information.

-process Display Process information.

-font = *filename* Use a non-standard font. The default font is *gacha6.kst*.

-Help Prints a help message.

Notes: Monitor takes a non-trivial amount of CPU time. Options such as *-process* should not be made faster than 5 seconds, or they would soak up all the spare machine cycles.

See Also:	Details (*-systat* and *-partition options*).
Bugs:	The *-partition* option does not work for machines with more than 10 disk partitions, because of a system bug.
	Redisplay when the Monitor window is uncovered is rather surprising; the information is temporarily inconsistent.

Mount

KeyWords:	mount, disk, install, load
Function:	attach a device to the file system
Syntax:	**Mount**
Description:	**Mount** is used to attach devices to the filesystem. After you invoke Mount, you will be prompted for the interface on which the device is attached (cio for PERQ1 and eio for the PERQ2). Then you will be prompted for the unit number. If your workstation has only one disk, type 0. If it has more than one disk, type the unit number of the disk to be mounted. The device from which the operating system was booted is mounted automatically by the system. A device may be mounted more than once.
Note:	The Mount and Dismount facilites are not related.

Oil

KeyWords:	oil, ed, vi, visual, sos, lined, teco, flash, editor, change
Function:	text editor.
Syntax:	Edit [[=]*FileName*]{*-Switch*}

Description:

Oil is a screen editor, in the spirit of Emacs. It can be invoked with or without a file specification. If you do not provide a file name, Oil will prompt you for one. By default, Oil is released under the name Editor.Run and can be invoked with the **Edit** command.

See *Oil: The Spice Ascii Editor*, in the Spice Users Manual for more detailed information albeit, not always up-to-date . The file **<boot>oil.hlp** is rather terse but always current.

Oil keeps a transcript of the editing session under the name *FileName*. **+**. The transcript is a copy of all the commands you typed during an editing session. By replaying it (i.e. by having **Oil** go through the same sequence, on the original file) you can save having to redo the changes yourself, should the machine crash while using **Oil**. Of course, this only works on the original file! (**Oil** saves your original file under the name *FileName*$. You can copy or rename *FileName*$ over the original file if it was damaged by the crash.) To replay a transcript file, invoke **Oil** as follows:

Edit = *FileName*

Switches

-help	prints help information.
-log	retain a transcript file (*FileName*. **+**). This switch is asumed by default.
-nolog	does not retain a transcript file.
-replay	replays *FileName*. **+**

Files:	<boot>oil.hlp, <boot>oil.keytran, <boot>oil.picture, <boot>oil.bugs
See Also:	Edit
Bugs:	Oil does not always exit correctly when the transcripting feature is on. Use the **-nolog** switch to avoid making a transcript.

On

KeyWords:	On, remote, execution
Function:	Execute a command on another machine.
Syntax:	**On** <MachineName> <CommandString> Description: If no parameters are specified, **On** prompts for them. In CommandString you cannot abbreviate the command name or use an alias. Many simple commands are handled directly by the shell and cannot be used in CommandString, e.g. version, details, ... There are no switches for the **On** facility itself but the CommandString can contain switches. Note: The remote server must be running on the machine you wish to contact.
See Also:	Remote.

Patch

KeyWords: Patch,debug,sdb

Function: Examine or modifiy any block in a file.

Syntax: **Patch** [*FileName*]

Description: This command is for use by system maintainers or programmers in making low-level changes to binary files. It allows you to see and modify every byte in the file.

If you do not specify a filename or if the specified filename does not exist, **Patch** prompts for the filename. When it has a valid file it prompts with

```
Read Block [0]?
```

Press **CR** to look at the block number in the brackets or type a new block number. You can also type **help** for online documentation that describes the commands you can use.

When you ask to read a block, **Patch** displays it byte by byte or word by word on your screen in 32-rows. You can reference each byte with the indices 0 to 511. **Patch** permits you to make temporary or permanent changes to your file.

To access all hard disk blocks, you can patch the /sys/ file.

Pascal

KeyWords:	pascal, compile, pp
Function:	Pascal compiler.
Syntax:	**Pascal** [*SourceFileName*] [~ *SegmentProgramName*] {*Switch*[= *Value*]}

Description: **Pascal** translates Pascal source files into segment (**.seg**) files that can be linked and run. For a description of the extensions to the Pascal language that are implemented by this compiler, see the *Perq Pascal Extensions* document in the *Spice Programmers' Manual*.

You can specify any number of switches. Note that if you specify a switch multiple times, the last occurrence is used. If you specify the **-Help** switch, the compiler ignores other information on the command line and displays a Help message.

Switches:

-Help The **-Help** switch provides general information and overrides all other switches.

-Range,-NoRange
enable or disable generation of range checking code (enabled by default.)

-Verbose, **-Quiet** enable or disable display of procedure and function names as they are compiled (enabled by default).

-Auto, **-NoAuto** enable or disable automatic initialization of default input/output files, i.e., generate code to call **reset(Input)** and **rewrite(Output)** at the start of the program (disabled by default). **LibPascalInit** takes care of initializing the input/output on behalf of the user's program. **-Auto** is useful if **LibPascalInit** is not linked in.

-Scrounge[= *FileName*], **-NoScrounge**
enable or disable generation of symbol-table (*FileName*.**sym**) and Q-code map (*FileName*.**qmap**) files for use with the Kraut debugger (enabled by default.) *FileName* defaults to *SourceFileName*.

-List[= *FileName*] The **-LIST** switch controls whether or not the compiler generates a program listing of the source text. The default is to not generate a list file. If the **-List** switch is given, the compiler prints with each source line the line number, segment number, and procedure number. The compiler appends the extension .Lst to filename if it is not already present. If you omit filename, the compiler uses the source file name. If the .Pas extension is present, it is replaced with the .Lst extension. If the .Pas extension is not present, the .Lst extension is appended.

-ErrorFile[= *FileName*]
This switch allows compilations to be left unattended. Normally when the compiler detects an error in a program, it displays error information (file, error number, and the last two lines where the error occurred) on the screen and then requests whether or not to continue. The **-Errorfile** switch overrides this action. When you specify the switch and the compiler detects an error, the error information is displayed and written to a file and there is no query of the user. Lastly, the compiler does display the total number of errors encountered on the screen. The compiler appends the extension .Err if it is not already present. If you do not specify a filename, the compiler uses the source file name. If the .Pas extension is present, it is replaced with the .Err extension. If the .Pas extension is not present,the compiler appends the .Err extension.

The error file exists after a compilation if and only if you specify the **-Errorfile** switch and an error is encountered. If the file filename.Err already exists from a previous compilation, it is rewritten, or deleted in the case of no compilation errors.

-Version = *String* The **-Version** switch permits the inclusion of a version string in the first block of the .Seg file. This string has a maximum length of 80 characters. Because a space or a hyphen (-) terminates the version string, you must enclose the string in double quotes if it contains more than one word or a hyphen. Currently this string is not used by any other PERQ software, but it may be accessed by user programs to identify .Seg files.

The version string is terminated by either the end of the command line, the occurrence of a '-' character, or a space. Therefore you must enclose the string in double quotes if it contains more than one word or a hyphen.

-Nomixedmodepermitted = *String*
Prohibits the useage of mixed-mode expressions.

-Mixedmodepermitted = *String*
Permits the useage of mixed-mode expressions (default).

-Peepopt = *String* Disables Peep-Hole code improvement of output code (default).

-Nopeepopt = *String*

Disables Peep-Hole code improvement of output code (default).

-Disassemble = *String*
Enables printing of disassembly listing on the listing file.

-Nodisassemble = *String*
Disables printing of the disassembly listing on the listing file (default).

-Map = *String* Enables printing of variable allocation map on the listing file.

Nomap = *String* Disables printing of variable allocation map on the listing file (default).

-Comment = *String*
The **-Comment** switch permits the inclusion of arbitrary text in the first block of the .Seg file. This string has a maximum length of 80 characters. Because a space or a hyphen (-) terminates the version string, you must enclose the string in double quotes if it contains more than one word or a hyphen. The switch is particularly useful for including copyright notices in .Seg files.

-GlobalInOut, -NoGlobalInOut
-GlobalInOut treats the standard input/output file descriptors as global (i.e. external), even when compiling a program (as opposed to a module.) **NoGlobalInOut** treats standard input/output file descriptors as local when compiling a program (produces slightly faster code in the program); **NoGlobalInOut** is the default.

Examples:

pascal myprog compiles **myprog.pas** and creates the files **myprog.SEG**, **myprog.SYM**, and **myprog.QMAP**

link myprog links the program **myprog** and creates a **myprog.run** file than is executed by typing

myprog executes myprog.run

Note: Invoking the PERQ Pascal compilation facility through the use of the Pascal command by-passes any use of the default file remembered by the shell.

See Also: Compile, Link, Debug, Kraut Documentation

PasMac

KeyWords:	pasmac, preprocessor, macro-processor, #
Function:	macro processor for Pascal.
Syntax:	**PasMac**[*Inputfile*[*Outputfile*]]
Description:	PasMac allows programmers to declare and use macros in their Pascal source code. These macros are expanded by the PasMac preprocessor before compilation. When run with no arguments, PasMac prompts for arguments it needs, such as the *InputFile*. See the file **/usr/spice/doc/pasmac.doc** on CMU-CS-Spice for more information.

Path

KeyWords:	path, connect, cd, current directory, directory, pwd, print working directory, working directory
Function:	change current directory.
Syntax:	**Path** [*PathName*]
Description:	Your path is your current directory, alternately known as your working directory on some computer systems. This is the directory you are "currently using". The current directory name is maintained for program use as the value of the *environment variable*, **current**. With an argument, **Path** changes the current directory to *PathName*. If **Path** is called without an argument, it prints the current path. A new path can then be typed or a (return) can be used to exit without changing the current path. An error message will be printed if you try to set your path to a nonexistent directory.
Notes:	"Path *Pathname*" is equivalent to "define current *Pathname*".
See Also:	SetSearch, Define.

Pause

KeyWords:	pause, wait,
Function:	suspend execution of a command file.
Syntax:	**Pause** *Text*

Description: **Pause** prints *Text* on the console. and then types the message:

```
Type <return> to continue.
```

The shell raises the *Sapphire ICON attention flag* ("!") and waits for a line to be typed before continuing. This command is most useful in command files when some user action is required before proceeding, *e.g.*, changing floppies.

PCC

KeyWords: pcc, cc, c

Function: Compile a C program to produce an assembly file.

Syntax: **PCC** <filename.i> <filename.s> Description:
PCC is the PERQ C compiler. After you have used the Cpp (preporcessor) facility, running **Pcc** is the next step in preparing a C program to run on the PERQ workstation. The first argument is the intermediate file which was created by the Cpp facility and the second argument is the file created by **Pcc** (which will contain the assembly code).

The output file from **Pcc** is then assembled with the Asm facility.

Notes: In a future release Cpp and Pcc will be replaced by a CC (C compiler) facility similar to the UNIX software cc facility.

Print

KeyWords:	Print, press, dover, cz, fonts, spool
Function:	'pressify' files and send them to dover printers
Syntax:	**Print** [[*files*]{-*Switch*[= *Value*]}

Description:

Print is the PERQ version of the Vax/Unix program czarina. Print reads in text files, converts them to press format and spools them to the Dover (using Spool).

At some point in future the environment variable PRINT will be used to specify defaults. The value of PRINT is parsed as a string of arguments before the arguments that appear on the command line. For example "PRINT='-f TimesRoman8'" sets your default body font to 8 point Times Roman.

If a file is already in press format Print will discover this fact by examination. Such files will be shipped as-is with minor changes to the document directory to make the cover sheet of the final output agree with the selected options.

A font name has three parts: A family name (from the set TimesRoman, Helvetica, Gacha, and Sail; Gacha and Sail are fixed width), an optional point size (1 point = 1/72 inch -- 8 point is a good small font), and an optional facing (b = bold, i = italic, nothing = roman). So Gacha8b is 8 point bold Gacha, Helvetica12i is 12 point italic Helvetica.

Print will bind illustration files into press files a la PressEdit. If a parent fiducial mark consisting of the string "<= =<filename<" occurs in a press file it will look for the file on the standard search path and include it in the document. The included file must be a one page press file and have the child fiducial mark "<= =<<" somewhere in the file. Print will register the heads of the parent and child fiducial marks.

Parent fiducial marks are typically inserted by the PressPicture macro in Scribe, while child marks are typically inserted by the user in DP. Certain other programs automatically produce fiducial marks, for instance poof (the Unix plotting program), plot (the DEC 10/20 plotting program) or redraw (the Alto program for converting draw files to press files).

Switches:

-1	Sets one column mode (default).
-2	Sets two column mode.
-a	Use absolute pathnames in the page headers (default, NYI).
-A	Don't use absolute pathnames.

-b banner	Uses banner to label the output. It will appear on the cover page on the line labeled "File:".
-c n	Causes n copies of the output to be produced. The default is one.
-d destination	Says which printer to send the output to. The destination parameter is a comma separated list of locations or server names, for example, "Ruby,Calais" or "3", and is passed to Spool with the -w flag.
-f font	Sets the font to be used for the body of each page. Defaults to Gacha10 , unless two column rotated mode is used, in which case it defaults to Gacha8.
-F font	Sets the font to be used for page headings. Defaults to Helvetica12.
-g	Causes Print to ignore characters which look like garbage, which usually occurs when you try to use Print on a non-text file.
-h header	Sets the string to be used for page headings to header. The default header is constructed from the file name, its last modification date, and a page number.
-H[im]	Causes Print to compute and print the height of each page in a given press file. Default output units are millimeters (-H or -Hm), while -Hi produces output in inches. This flag is especially useful for handling illus- tration files.
-l	Causes line printer simulation mode to be used: pages will be 66 lines long and headers will be omitted.
-o	Causes Print to list the value of each character omitted because of incomplete fonts.
-p name	Causes the press file to be written to the named file rather than being shipped to the Dover.
-q	Quiet mode -- Print won't give the "general information messages" about the number of lines wrapped, pages printed, and so on. Error messages are still sent to stderr.
-r	Causes the output to be rotated 90 degrees on the page (landscape mode). This is good for output that requires a wide page or for pro- gram listings when used in conjunction with two column mode. Keep in mind that this only works for text

files and does not work for existing press files, such as those produced by Scribe. "Print -2r files" is the 'approved' way to get program listings on the Dover.

-R	Causes the output to not be rotated 90 degrees (portrait mode). -R is useful if your default sets -r.
-s pages	Selects pages to be printed. Pages may be a single page specification (eg. "5"), a range of pages ("5-10" or "5:10"), or a list of page specifications (eg. "3,11-13"). The last page may be represented symbolically by '$' or '*'.
-S	Toggle output byte order. Possible byte orders are PDP-11 and ALTO. Use -S to change the byte order from the default.
-t	Causes page titles to be omitted.
-w	Causes long text lines to be wrapped to the next line at the margin, as opposed to being truncated. Wrapped lines are marked with a " right margin. The margin is defined to be 3/8 inches from the right side of the page (or the start of the second column if you are in two column mode and in the first column). This is the default setting.
-W	Causes long lines to be truncated instead of wrapping them.

FILES /usr/cmu/lib/fonts.width describes all the available fonts.

Notes: Print is a Native C program taken for the most part directly from the cz sources. It is still being developed and some of the above switches may not work. It is dependent on fonts.width (usually in the boot area) for its font information.

See Also: Spool

Priority

KeyWords: priority, nice, run queue

Function: set execution priority.

Syntax: **Priority** [*Process*] [*Level*]

Description Priority sets the execution priority of the specified process to *level*. Priority levels range from 15 to 0 with 15 the highest.

For *Process*, you may use the process name, the name in the Icon window, or the process number. The process name is usually the name used to invoke the program; any unique prefix will suffice. In specifying an Icon window name, all processes controlled by that window will be affected. The only way to affect a specific process, when its process name is not unique or the Icon window influences several processes, is to use the unique process number. The command "details - systat" shows the process name, unique process number, and controlling window name for every process.

If Priority is invoked without an argument. the user will be prompted for a process.

See Also: Kill, Resume, Suspend.

Examples: The following priorities are recommended for accent and should be placed in your InitialShell.Cmd file:

```
priority 3mhznetserver 13
priority 3mhzmsgserver 13
priority Sapphire 12
priority Typescript 11
priority Process  10
```

Description:

For *Processes*, you may use the process name, the name in the Icon window, or the process number. The process name is usually the name used to invoke the program; any unique prefix will suffice. In specifying an Icon window name, all processes controlled by that window will be affected. The only way to affect a specific process, when its process name is not unique or the Icon window influences several processes, is to use the unique process number. The command "details - systat" shows the process name, unique process number, and controlling window name for every process.

If is invoked without an argument. the user will be prompted for a process.

PrqMic

KeyWords:	PRQMIC, microcode
Function:	Processes PERQ microcode
Syntax:	**PRQMIC** <Input file> [-Help] Description: PRQMIC is the PERQ microcode assembler. It creates files to be used by the placer (PrqPlace) to make the binary files. The input to the microcode assembler is a file of microcode source. For information on microcode syntax and other information, refer to the document "Microprogrammer's Reference" in the Spice Programmers Microprgramming Manual. If the input filename is not supplied, PRQMic prompts for it.
See Also:	PrqPlace

PrqPlace

KeyWords:	PrqPlace, microcode
Function:	Create binary files from the output of the PERQ microcode assembler PrqMic.
Syntax:	**PRQPLACE** <RootFile> <ListingFile> {-switch}
Description:	RootFile is the name of the microcode source file which has been assembled (without the .MICRO suffix). ListingFile is the file to which the microcode listing is to be written. This facility is run after the source microcode has been assembled by PrqMic. For information on microcode syntax and other information, see the document "Microprogrammer's Reference" in the Spice Programmers Manual.

Switches:

-DELETE	Deletes all intermediate files (this is the default)
-NODELETE	Saves the intermediate files.
-HELP	Displays information about PrqPlace.

See Also: PrqMic

QDis

KeyWords: qdis, qcode, disassembler, pretty-print, statistics, seg file, code file

Function: Q-Code disassembler.

Syntax: [*SegFile*] [~ *ListFile*] {*-Switch*}

Description: **QDis** is a disassembler for Pascal Q-Code. It decodes a **.seg** file into Q-Code lists various pieces of information, depending on the switches used. It is normally used by compiler maintainers to diagnose code generation errors. However, users can benefit from it, by using **QDis**'s statistics gathering facilities. The program can collect statistics about Q-Code (and Q-Code sequence) usage patterns. These can be used to fine tune time-critical application programs.

You must specify the name of a segment (.Seg) file that contains the Q-Codes to disassemble. The .Seg file you specify can contain wildcards. If QDis does not find the specified .Seg file, it appends the extension .Seg and tries again.

The listfile specifies a file to contain the QDis output. If you omit a listfile, QDis outputs to the console.

When you initiate QDis, it identifies itself as the QCode Disassembler and displays switch settings. QDis then displays the following information (Depending on switch settings):

Name of program or module
Name of source file from which generated
QCode version number
Size of global data block
Length of identifiers
Routine descriptor block number, if one exists
(this information pertains only to FORTRAN generated .Seg files)
Version string
Comment string
Language
Number of imported segments
Import list block number
Diagnostic block number, if one exists
(this information pertains only to FORTRAN generated .Seg files)
Unresolved references block number, if exists
(this information pertains only to FORTEAN generated .Seg files)

If you did not specify a routine name or number to disassemble, QDIS displays a list of the program's routines and the following information about each:

Routine name

See Also: Copy and Directory for a list of wildcard characters.

Resume

KeyWords: resume, bg, fg

Function: resume a process.

Syntax: **Resume** [*Process*]

Description: Resume causes a suspended process to be resumed.

For *Process*, you may use the process name, the name in the Icon window, or the process number. The process name is usually the name used to invoke the program; any unique prefix will suffice. In specifying an Icon window name, all processes controlled by that window will be affected. The only way to affect a specific process, when its process name is not unique or the Icon window influences several processes, is to use the unique process number. The command "details - systat" shows the process name, unique process number, and controlling window name for every process.

If Resume is invoked without an argument. the user will be prompted for a process.

See Also: Kill, Priority, Suspend, Systat, Sapphire documentation.

Run

KeyWords:	run, call
Function:	execute a program.
Syntax:	**Run** [*Program*]
Description:	**Run** is a way to invoke *Program*. The shell looks for *Program.run* and *Program.exe* by following the shell searchlist, **<run>**. CTRL ? provides a form of command name completion, but CTRL ? uses **default** rather than **run** for its search list. **Run** aslo may use or set the default file remembered by the shell. If you have been editing and compiling you have created a default file because of the *Alias* definitions for **Edit** and **Compile**. Typing **Run** without an argument will use the default file.
Note:	**Run** is the *Alias* entry with definition "run" -setdefault -usedefault".
See Also:	?, Alias, Compile, Edit, UnAlias

SetProtect

KeyWords:	SetProtect, chmod,access, rights
Function:	Changes the access rights on files.
Syntax:	**SetProtect** [*PathName*]{-*Switch*}

Description: The SetProtect command sets access privileges on files or directories. A file is owned by the user who was logged in when the file was created. The owner determines read privileges (permission to look at the file) and write privileges (permission to change the file). Read privileges including typing, copying, and appending (but now appending to the file). Write privileges include editing and deleting. These privileges are set separately for other users and yourself (for example, you might want to prevent yourself from accidentally deleting a very important file).

Switches:

-OwnerRead (default)
-OwnerWrite (default)
-WorldRead (default)
-WorldNoWrite (default)
-OwnerNoRead
-OwnerNoWrite
-WorldNoRead
-WorldWrite
-Help (Prints a help message.)

You may use wildcards in the terminal component of *PathName*

The access privileges you set for a directory apply to all the files in that directory. For example, suppose in the User partition you have several nested directories under the directory Reports and the path to one of the files is as follows:

```
/sys/user/reports/prod/inv
```

Read privileges are checked at every step in the path, and if write access is required for the requested operation, it must be present on both the file and its parent directory. Thus removing read access from the Reports directory, will eliminate any reading or writing of all the files in Prod, while removing write access from Prod, will eliminate writing on the Inv - or any other file in Prod.

To find out what access privileges are in effect for any directory, give the Direct command with the **-all** or **-prot** switch **dir -all** or **dir -prot.**

See Also: ChangeOwner

SetSearch

KeyWords:	setsearch, path, search rules, set, setpath, path
Function:	modify search paths.
Syntax:	**SetSearch** [DirName]{,*DirName* } {-*Switch* }

Description:

SetSearch allows you to modify your search list (**default:**). Some programs that interpret a relative pathname, i.e. a filename that does not begin with a '/', will look in all the directories listed in **default:** to find the file. The first directory that contains the file is chosen.

If no argument is specified in the command line, the current search list is printed and you will be given three choices:

- Name to push:
  If you specifiy a Dirname, **SetSearch** pushes that name onto the search list and retypes the list.

- **-pop** to remove an item
  The **-pop** switch remove the first item (excluding the current directory) off the list.

- CR to exit
  Typing just "*return*" will exit without changing the search list.

DirName in the command argument or in the above case must be a valid *Directory*.

Pushing and popping are actually done one past the beginning of the search list so that **current:** *remains at the head of the list.*

Switches:

-pop[= *n*]	Removes the *n* items below the first item from the search list. The default for *n* is 1.
-Reset	Pops all but the first (which is usually **boot:**) directory from the search list.
-Help	prints a help message

Note:

SetSearch affects only the **default:** search list. Setsearch is a shortcut for

Define <default> -after = 1 { *DirName* [,] } $_{\text{reversed}}$

Note: The directories are pushed onto **default:** by **SetSearch** whereas **Define** splices its argument onto **default:** in the order the elements are presented.

The shell uses the search list, <run>, *to find commands to execute. This list can only be updated using* **Define.**

The only way to not have **current:** at the head of the search list is to explicitly set **default:** using **Define**.

See Also: Define.

Examples:

Setsearch /sys/user/guest
is equivalent to:

```
define default: /sys/user/guest/,default:
```

SetSearch -pop is equivalent to:

```
define default: -after = 1 -replace = 1
```

(note: -after = 0 would refer to the top of the stack - which is the **current:** directory).

SetSearch -pop /sys/user/guest
is equivalent to

```
define default: -after = 1 -replace = /sys/user/guest/
```

Shell

KeyWords:	shell, monitor, cshell, interpreter, command-processor
Function:	Spice system command interpreter.
Syntax:	**Shell [-quit]** [*command*] **[-Help**
Description:	The **Shell** is the system command interpreter. See the *Introduction to the Spice Users' Manual* for a description of the shell and its use. Typing the Shell command tells the system that you want to open a new window for a new shell. After you give the Shell command, you will prompted to specify the window corners. If a command is specified, that command will be executed as soon as the new window is opened. If the -quit switch is specified, the new window will be eradicated as soon as the specified command has been executed. The -quit switch must precede the command.
See Also:	**?, CTRL ?**

Show

KeyWords:	show, env, environment, set, variables
Function:	examine environment variables and search lists.
Syntax:	**Show** [*Name*] { *Switches* }

Description: **Show** is used to display *environment variables*. If **show** is invoked without any argument, all the *environment variables* are displayed according to the *Switches*.

Environment variables are used to communicate information between processes, typically between the shell and programs it runs. The *Environment Manager* serves as a repository for the *environment variables*.

There are two types of *environment variables*:

strings	for simple information.
search lists	for ordered lists of directories. These search lists indicated by a name with a trailing ":" in shell command lines that refer to them, viz. **current**, **default**, **run**.

Search lists are used to interpret filenames which are not specified as absolute pathnames, i.e. names not starting with '/'. In general, programs will either interpret the pathname relative to **current** or look for the pathname in each of the directories in **default** in turn until a file is found. Note: The interpretation of relative pathnames is handled by the individual programs and some variation from this model is possible. Note: **current** is a single element search list.

An *Environment variables* scope effects how variable is found and how the variable is shared. The scope values are:

Global	The variable is shared for read and write operations with all processes referencing the variable.
Local	All access to the variable is restricted to the process that created the variable. A child process can inherit a copy of all of its parent's local variables at the time that it is *spawned*. For example, when a shell creates a subshell, the subshell copies the parent's local variables. Write operations do not effect the parent's copy.

When a variable is created it must be given *Global* or *Local* scope. When the variable is referenced a third option is available: *Normal* "scope". This looks for a *Local* variable of the given name and iff that fails looks for a *Global* variable.

The evaluation of a *search list* produces a list of strings. These strings may contain imbedded *search lists*. If the variable is *resolved*, any imbedded *search lists* will be recursively substituted for.

Switches:	**-Local**	lookup the local variable.
	-Global	lookup the global variable.
	-Normal	try to find a local variable and if it does not exist look for a global variable.
	-Resolve	fully evaluate the variable replacing imbedded *search lists* recursively and display all elements of the resultant evaluation.
	-Full	fully evaluate the variable replacing imbedded *search lists* recursively and return all elements of the resultant evaluation.
	-Help	prints a help message.

Examples:

Show <default> shows the default search list

Show <default> -Resolve
shows the default search list in terms of real directories.

Show shows everything

Note: "Show options" is equivalent to "define options -show"

See Also: Define, Details, SetSearch, Path

Speak

KeyWords:	speak, listen, send, rsend, finger, who, wall, broadcast
Function:	Program for sending Perq users short messages
Syntax:	**Speak** [*UserName*]{*-Switch*}

Description: This program allows the user to send short messages to another user's listener. To send a message to someone, prefix the message with the name of the destination listener enclosed in slashes, like so:

```
/ets/ hi there
```

From then on until you change the destination (by prefixing a message with another destination) your destination will be that listener and any messages you type will go to him. If for any reason **Speak** cannot find the destination, an error message will be printed and your destination will be changed to your own listener name. **Speak** indicates the current destination in its prompt and on its title line.

Sending a message to the imaginary user "who" will get you a list of all users on the network currently running listeners.

Note that the name of your listener and your speaker should be the same name! If you do not include a *<UserName>* when you start **Speak**, your name will default to the name of your machine.

Switches: **-Help** Prints a help message.

Notes: A user must be running the **Listen** program before you can **Speak** to him.
You cannot call yourself by the name "who".

See Also: Listen

Bugs: The "who" service seems to be extremely unreliable. This is being worked on. It seems to be related (not surprisingly) to the number of crashes and restarts of listeners on the network. Word will be posted when this is either fixed or made more robust.

Spoonix

KeyWords:	spoonix, unix, c, lnk
Function:	Spoonix is a server process, called by the Spoonix C library, to implement a collection of frequently used Unix system calls. Because it is new software, it should be run with a (very) small window to show diagnostic error messages. Spoonix must be executing before the Spoonix C library can function.
Syntax:	Spoonix
Notes:	Spoonix and the appropriate C compiler, assembler, linker, and libraries must be retrieved and installed by using the appropriate updates and command files. At the time of this printing, current information will be found on CMU-CS-SPICE in /usr/spoonix/doc/GetSpoon.press. Additional information can be found looking at the Berkeley 4.1 Unix documentation of the C library.
See Also:	See also Assembler, C compiler, and Linker, and the above-mentioned documents.
Bugs:	There are inconsistencies between the functions of Spoonix and Unix. Spoonix is not sensitive to filename case, and it does not simulate Unix (hard) links perfectly. At the time of this writing, Spoonix is still evolving.

Statistics

KeyWords:	statistics, run time
Function:	enable system performance statistics.
Syntax:	**Statistics [yes | no] {-Switch}**

Description: The **Statistics** command enables (yes) or disables (no) the printing of performance statistics by the Process Manager. These statistics are printed for each process after the process dies. The operating system constantly collects statistice about the performance of the swapper. The Statistics command permits you to enable or disable the display of information after each process executes.

Format: **Statistics [Yes|No] [-switch]** The option is "Yes" to display the statistics after each process executes, or "No" to end the displaying of statistics.

The only switch is **-Help** to obtain information about the Statistics facility.

After each process executes, the shell displays statistics for that program in the following format:

```
Elapsed
41.0 secs
Load     0.000 secs
Execute
0.0718960 secs
```

"Elapsed" is the total time since between the Start and finish of the program.

"Load" is the time spent loading the program and its modules into memory.

"Execute" is the total time spent executing including IO, Swap and Move times.

Switches: **-Help** Prints a help message.

Bugs: The Load time is always set to 0.

Stut

KeyWords:	Stut, tape, streamer
Function:	Saves and retrieves files on tape using the Streamer.
Syntax:	**Stut** [-*Switch*]
Description:	STUT is the STreamer UTility program used to save and retrieve file on tape using the streamer. The use of this facility is described in the *Tape Streamer User's Guide for the Accent Operating System*.
Switches:	**-Help** Prints a help message.

Suspend

KeyWords: suspend, CTRL Z, ↑Z, control Z, wait

Function: suspend a process.

Syntax: **Suspend** [*Process* -Help]

Description Suspend halts the execution the specified process. The process may later be Resumed.

Sapphire will suspend the listener process when CTRL DEL is typed. The Sapphire symbol is displayed while Sapphire waits for a command. If a space is typed rather than a "real" Sapphire command, Sapphire resumes the process.

For *Process*, you may use the process name, the name in the Icon window, or the process number. The process name is usually the name used to invoke the program; any unique prefix will suffice. In specifying an Icon window name, all processes controlled by that window will be affected. The only way to affect a specific process, when its process name is not unique or the Icon window influences several processes, is to use the unique process number. The command "details - systat" shows the process name, unique process number, and controlling window name for every process.

If Suspend is invoked without an argument. the user will be prompted for a process.

See Also:

Kill, Priority, Resume, Systat, Sapphire documentation.

Type

KeyWords: type, cat, more, list, print, display, show, file

Function: display a file in a window.

Syntax: **Type** *Filename* {*-Switch*}.

Description: **Type** displays a file on the screen. If no *Filename* is specified then the name remembered from the last **Edit** or **Compile** command is used. Type is an alias for **run type -usedefault.**

Type does extension completion on the file name you specify. If the file to print is FOO.PAS, you only need type FOO. The extension that **Type** knows about, in order tried, are: .Pas, .For, .Micro, .Cmd, .Dfs, and .Doc.

Switches:

-Wait When a form-feed character is encountered in the file being displayed, the message

```
** type a <return> to continue:
```

will be displayed and the program will wait for the user to type RETURN indicating that he is ready to continue.

-NoWait Do no special processing for form-feed characters.

-Help Type a short message describing the **Type** command.

Note: To keep text from scrolling out of a window before you have read it,type CTRL LF to the shell for that window. Once you have thus set more-mode for the window, whenever output from a single command is about to overflow the window, output will pause and a black bar will appear at the bottom of the window. At this time you should type LF to allow scrolling to continue. If you wish to get out of more-mode, type CTRL \.

See Also: Alias, Shell, and *Introduction to the Spice User's Manual* section on Scroll Control.

UnAlias

KeyWords:	unalias, alias, abbreviations, commands, shell
Function:	remove an *Alias* entry.
Syntax:	**Unalias** [*Command*]

Description: **Unalias** removes *Command* from the shell's *Alias* table. *Command* must be typed exactly as it was established by **Alias** or appeared in ?; no abbreviation is allowed.

After "**UnAlias** *Command*", *Command* will be sought following the shell searchlist, **<run>** and *Command* may no longer be abbreviated. CTRL ? provides a form of command name completion, but, CTRL ? uses **default** rather than **run** for its search list.

If **Unalias** is given no argument, a usage reminder is printed.

Bugs: Some of the initial shell *Alias* table entries will not go away even though an *Unalias* is performed on them and they no longer appear in the ? *Alias* table listing. For example, ?, **Help** ... can not be removed, thoughthey will disappear from the table. They can, of course, be *Alias*ed to a different command.

See Also: ?, Alias, Shell

Update

KeyWords: update, transfer, move, save, archive, store, install, restore, retrieve, backup, dump

Function: move a collection of files from or to a Vax host.

Syntax: **Update** *FileGroupName*{-*Switch*[= *Value*]}

Description: **Update** is used to transfer files between the PERQs and the Vax Systems. In addition to just storing and retrieving the files, it checks the dates of the files, and if they do not need to be transfered no transfer is done.

A set of files can be identified by a logical name (*FileGroupName*) and can be stored or retrieved as a group, by specifying the name of the group, followed by any relevant switches. It is not strictly necessary to have a *FileGroupName*, a Vax directory name can be used, in which case you must use a different version of the command line (See the reference manual).

This program offers a rather large number of switches, options, and special commands. For complete details, see *Update: A Version Control/File Transfer Facility*. The following are a just a sample of the switches users are more likely to need in order to retrieve Spice software:

Switches: Note,this is only a partial list.

-List The program prompts for a group name specification and lists all registered file group names which match the specification. It accepts wild cards. Typing RETURN to the prompt is equivalent to typing "*" and results in a listing of ALL file group names currently known to this program.

-Retrieve Retrieve a group of files (default).

-Host = *HostName* Specify the remote host. For retrieving, the default is **CMU-CS-CFS**.

-Login = *Account* The account to be used for login in on the remote host. Normally not needed, unless retrieving from a protected directory.

-Test, -Current, -Old, -Version = *VersionNumber*
Specify the version to retrieve. The default is **Current**.

-Check Simulate the operation and list the files that would be transferred but don't do any transfers.

-Supersede, NoSupersede
These two are mutually exclusive. The former specifies that if a file being transferred is older than the one already there, the "newer" file must be overwritten anyway. The latter specifies that an "older" file should never be overwritten with a "newer" file. If neither **-Supersede** nor **-NoSupersede** are specified the user is prompted.

-Help Prints a complete list of switches and terminates.

-Document Types the contents of **<Default>Update.Doc**. This file is distributed with the program and describes **Update** is some detail.

For the specification of additional switches and the format of the store command files, please read the Update documentation.

See Also: CMUFTP

Verbose

KeyWords:	verbose, command file, echo, printing
Function:	control printing of *shell* command lines before execution.
Syntax:	**Verbose** [*Boolean*] [**-Help**]

Description:

Verbose sets the flag which controls whether the shell command line is printed out before it is executed. The flag applies only to shell lines that are read in command files -- not those typed at the console.

If **Verbose** is set to *True*, each command line will be printed before it is executed. If **Verbose** is set to *False*, command lines are merely executed.

If **Verbose** is invoked with no argument, it will print out the value of the flag.

The following argument values are allowed:

True, Yes	represent the boolean *True*.
False, No	represent the boolean *False*.

Switches

-Help	prints a reminder of the syntax.

Note: Only the first character of the argument is tested, ignoring the case.

Bugs: The flag value is not maintained using a stack discipline; thus setting and clearing the flag in a command file, will not work as expected.

Version

KeyWords:	version
Function:	show shell version number.
Syntax:	**Version**
Description:	**Version** displays the shell version number followed by a short list of the changes in this version of the shell.

Index