I spy

spy is a Navigation utility for quickly browsing the Unix file system.

I INTRODUCTION

I.1 WHY USE SPY?

If you are using a UNIX shell, *spy* is a fast text-based file-navigation utility for organizing your files. It provides greater speed in manipulating files than a GUI, while also providing direct access to scripts and UNIX commands, and being easier to use than the UNIX C shell.

I.2 SPY – INTERFACE TO UNIX

spy runs in a window shell, and provides an efficient interface to the UNIX file system via its single-keystroke commands. Anything you can normally do in a c-shell, you can do in *spy*, but more quickly. Using single keystroke commands, *spy* quickly navigates between directories, and selects files in order to perform actions on them. Once files are selected, you can use all the standard UNIX commands with them through the use of a UNIX meta-key.

The most commonly used single-keystroke commands are listed in the following *Quick Help Reference*. These commands are covered in greater detail in the following pages.

2 SPY – QUICK HELP REFERENCE

2.1 SPY COMMANDS

J or K	Move cursor down / up
H or L	Move cursor left / right
U	Up a directory
D	Down a directory or Display a file (view only) using *less*
Q Enter	Quit *less* (if you entered file with D)
E or V	Both of these will Edit a file (with *vi*) or Enter directory
Ctrl B or Ctrl F	Back or Forward a page
T	Tag file (on/off toggle)
Shift T	Pattern tag (enter *string* to select all files that match)
Ctrl T	Tag all files in directory (on/off toggle)
Y	Copy (yank) file(s) to the copy buffer
I	Inventory of files in the copy buffer
P	Paste a file from the copy buffer (and clears the buffer)
Z	Clear the copy buffer
Shift R	Remove (delete) tagged files or current file if none tagged
C	Copy highlighted or tagged files to user specified directory
M	Move tagged files to a new Name or Directory
Shift J	Jump to user specified directory
Shift H	Jump to Home Directory
Shift L	File info (on highlighted or tagged files)
Ctrl L	Redraw the screen
!	Enter UNIX command (use % to refer to tagged files)
/ *string*	Search (move to) file whose name begins with *string*

2.2 VI COMMANDS

J or K	Move cursor down / up
H or L	Move cursor left / right
A	Add text (use Esc to exit back to command mode)
I	Insert text (use Esc to exit back to command mode)
X	Delete a character
D D	Delete a line
Y Y	Copy current line to Clipboard
P	Paste after current line (deleted text, or copied with Y Y)
: W Enter	Save (Write) the file without quitting
: W Q Enter	Save (Write) the file and then Quit
: Q ! Enter	Quit without saving
Alt E	Launch *vi* to edit the contents of a Houdini edit field
/ *string*	Search (move to) *string* within file
/ Enter or N	Go to next occurrence of *searchString*

3 STARTING SPY

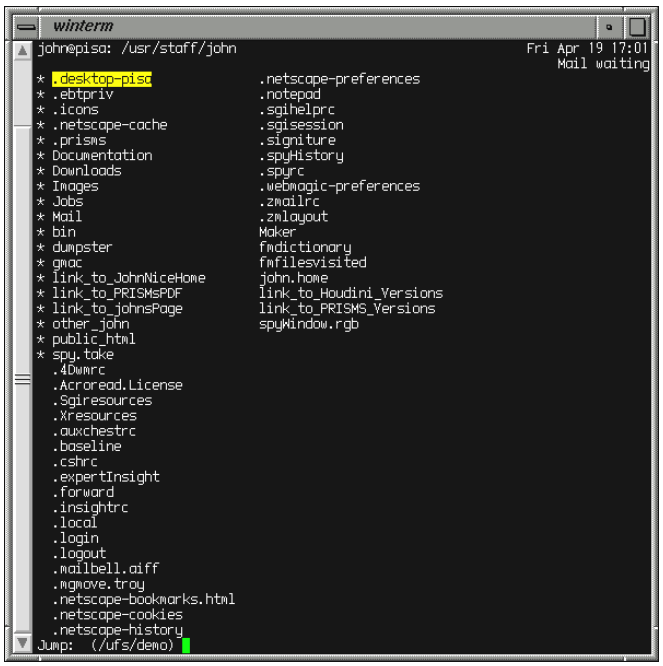
spy starts automatically if you login as *animate* or any account in which *Houdini* has been setup. Otherwise, when you see a UNIX *csh* prompt (`%`) type:

```
% spy
```

or

```
% $HFS/bin/spy
```

A window appears similar to the one below:

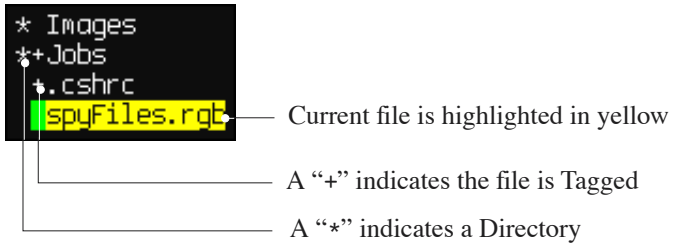


Displayed on the top line are your name, the machine you are logged into, the current directory, the current date / time, and a UNIX mail status indicator.

Below this are listed all the files in the current directory:

- Files are listed in columns starting on the left.
- Files with a * to the left of the name are directories.

3.1 CURRENTLY HIGHLIGHTED FILE



4 NAVIGATING WITH SPY

4.1 MOVING THE CURSOR – NAVIGATION CONTROLS

The current directory is always displayed at the top of the *spy* window.

The current file is always highlighted in yellow within the *spy* window.

The yellow high-light is also known as the *cursor*. You can move the cursor within the *spy* window with the following keys:

MOVING DOWN / UP

J / K Moves the cursor down or up a line.

MOVING LEFT / RIGHT

H / L Moves the cursor one column to the left or right.

4.2 SEARCHING WITHIN A DIRECTORY

MOVE TO THE NEXT FILE THAT BEGINS WITH...

The *search* command moves the cursor move to a file that begins with the letters you specify.

/ `string` Enter

Typing / and a string selects the file that begins with *string*. You can use standard UNIX wildcard characters to perform pattern searches. For example, entering:

/ `Scene1*.pic` Enter

we are telling the search to find "any name starting with *Scene1* and ending with *.pic*. This works because the * character is a wildcard character meaning any number or letters.

Similarly, we could tell the search to find **.txt*, in which case *spy* would search for any file that ends with the letters *.txt*.

There is a *spy* option to control search behaviour. See *relaxSearch* p. 308.

Tip: To find files in multiple directories, see the UNIX *man pages* for the UNIX *find* command. For a brief example of the *find* command, see *Searching Multiple Directories* p. 298.

MOVE TO THE NEXT OCCURRENCE OF...

Typing N moves the cursor to the next occurrence of a file or directory whose name starts with the *string* specified in the last search.

4.3 PAGE FORWARD OR BACKWARD

If there are more files in a directory than can be displayed in the *spy* window, *spy* indicates it is displaying *Page x of y*. To flip between these pages, use the keys:

Ctrl F Displays the next page of files (forward).
Ctrl B Displays the previous page of files (backward).

4.4 MOVING BETWEEN DIRECTORIES

Each directory can contain files and other directories. Use the following keys to move down and up between directories.

MOVING DOWN A DIRECTORY

D If the file that is currently highlighted with the cursor is a directory, typing D moves down into that directory. e.g. if you were in *$HFS/houdini* and the cursor was on the file *sbin*, you will then be in *$HFS/houdini/sbin* after typing D.

MOVING UP A DIRECTORY

U Moves *up* one directory level, and enters the *enclosing directory*. e.g. if you were in *$HFS/houdini/sbin*, you will be in *$HFS/houdini* after typing U.

4.5 JUMPING TO ANOTHER DIRECTORY

Shift J Prompts you to jump to another directory. *spy* displays a default directory name, which is the previous directory you jumped *from*. There are now several choices. Do one of the following:

- Typing Enter goes to the displayed default directory.
- Typing a directory name followed by the Enter key jumps to that directory. For example, typing: *$HFS/demo* Enter, jumps to the *$HFS/demo* directory.
- Typing Esc re-displays the default directory, and allows you to use the J and K keys to scroll through a list of directories from which you have previously jumped. Type Enter to jump to the displayed directory.
- Typing a question mark (Shift /) displays a list of directories from which you have previously jumped. Use the J and K keys to move down and up the list. Type Enter to jump to the highlighted directory.
- Type Bksp at any time to abort the jump.

Tip: A useful trick is to use the *XWsh* pop-up menu of the *spy* window (click in the window with) to clone your current window. You can then navigate in the second window to another location, highlight the directory name in the second window with the mouse, and click in the first window to paste the directory name in the *Copy* or *Jump directoryName* requested by *spy*. This is also useful for Copy/Paste.

Jump not only allows jumping to a directory, but also to filenames within the current directory or in another directory (the cursor is moved to the file). Names differ from directories in that directory names are preceded with a slash (/).

JUMPING WITH ENVIRONMENT VARIABLES

Environment variables can be used in pathnames, you may want to set up environment variables to lessen the amount of typing associated with long directory names. Instead of a directory path, simply type the variable name. For example when performing a jump (Shift J), typing:

```
$HFS/demo Enter
```

will jump to the directory containing the Houdini demo files. Names of directories stored in environment variables can be obtained by typing:

```
!printenv Enter
```

To create your own environment variables, see *Setting Your Own Environment Variables* p. 333.

4.6 GOING HOME

In UNIX, each user has a special directory to store files called the *home directory.* This is the directory which is usually displayed when you first log in to your machine and start *spy.*

Typing Shift H takes you back to your home directory.

Note: In UNIX, anywhere a pathname is requested, typing "~" will specify your home directory. This can be used to rapidly get to any file or subdirectory within your home directory. For example:

```
~/MyFiles
```

enters the directory */MyFiles* which is in your home directory. This can be used in conjunction with the *jump* command (Shift J) to jump back to your home directory.

4.7 SEARCHING MULTIPLE DIRECTORIES

To search multiple directories for a file, you need to use the UNIX *find* command. For example, typing:

```
!find /usr/staff/john -name "Jo*" -print
```

displays all the files beginning with *Jo* that are contained within any subdirectory of */usr/staff/john*. It is important to note that the *find* is case sensitive. See the UNIX *man* pages for more information.

For an easier way, you may want to try the *Find > Search For > Files* command available from the IRIX desktop menu. See *Finding Files* p. 342 for details.

5 WORKING WITH FILES

5.1 SELECTING FILES

Within a directory, you can tag a number of files to operate on. Common operations include deleting files, and displaying images on the screen. The files that are tagged will have a **+** to the left of the name. This section describes how to tag and untag files. Later you will see how you can use the `%` character in UNIX commands as a shorthand notation for the tagged files.

SELECTING A FILE

T Toggles the tag on the current file on/off. The second line of the *spy* window shows the number of selected file in brackets:

```
john@pisa: /n/berne/Documentation/Images/
Page 2/2 (+9)
```

SELECTING ALL THE FILES

Ctrl T Tags or untags all files in the current directory.

SELECTING SPECIFIC FILES

Shift T Prompts you with a file name pattern used to toggle the tags on the files which match the pattern:

`g*` Enter	tags all files starting with the letter *g*
`*.pic` Enter	tags all files names ending with *.pic*
`??.pic` Enter	tags all files starting with any two characters and ending with *.pic*, such as *21.pic* but not *121.pic*.

5.2 COPY, RENAME, AND DELETE FILES

Following are the *spy* commands for basic file management. For an alternative way to perform these functions, see the section *IRIX File Management* p. 337.

COPYING FILES

C Copies the tagged files to a new name or a new directory. When invoked, it asks you for place to copy the selected file to. If you specify a UNIX directory path, it will copy the file there. If a filename is entered (i.e. you didn't have any "/" characters in what you typed), then the copy is named to whatever you entered.

RENAMING AND MOVING FILES

M Moves the tagged files to a new name or a new directory. When invoked, it asks you for place to move the selected file to. If you specify a UNIX directory path, it will move the file there. If a filename is entered, the file is renamed.

DELETING FILES

Shift R Removes the tagged files after prompting you for a confirmation. To delete *all* the contents of a directory, use the UNIX command: *rm*.

5.3 COPYING AND PASTING FILES BETWEEN DIRECTORIES

There is a directory in your home directory called *spy.take*. It is used to store copies of files which you later put elsewhere. It is known as the *copy buffer*.

COPYING FILES TO THE COPY BUFFER

Y Copies (*y* stands for the archaic term "yank") tagged files into the copy buffer *(spy.take)*. You can copy as many files, on as many occasions as you want into the copy buffer. Files in the copy buffer remain there, even between sessions.

PASTING FILES FROM THE COPY BUFFER

P *Pastes* all the files from the copy buffer *(spy.take)* into the current directory, and empties the copy buffer.

INVENTORY OF FILES IN THE COPY BUFFER

I Displays an Inventory of files currently in the copy buffer.

CLEARING THE COPY BUFFER

Z *Empties* the copy buffer (the *spy.take* directory in your home directory).

Tip: A useful trick is to use the *XWsh* pop-up menu of the *spy* window (click in the window with) to clone your current window. You can then *Copy* (Y) your files from the first window, and then *Put* (P) them into the directory displayed in the second window.

5.4 CREATING DIRECTORIES

CREATING NEW DIRECTORIES

`mkdir` *`directoryName`* Makes a new directory with the given name. You can put several names separated by spaces on the same line. For example:

! mkdir myDirectory anotherDirectory Enter

DELETING DIRECTORIES

You can delete the directories with the same *spy* command (Shift R) that you use to remove files. You will need to ensure that you have deleted all files from the directory before you do this. You can also use the UNIX command *rm* with the *-r* option.

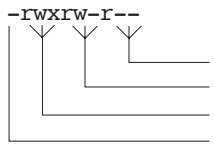
5.5 LISTING FILES WITH DETAILED INFORMATION

Shift L Lists all tagged files. If none are tagged, this command lists information for the current file. The list shows detailed information about the files.

EXAMPLE

```
-rwxrw-r-- john prisms 265535 12:29 Mar 21 96 testFile
```

The permissions of a file are displayed in the first ten characters. They provide the following information:



Permissions	Type
- no permission	- file
r readable	d directory
w writable	l link
x executable	

The first character tells you if it is a file, directory, or symbolic link. Then come three sets of three characters. These show the Read/Write/Executable (rwx) permissions of the file for the owner, the group to which the file belongs, and for all other users.

If a person or group has permissions to read a file, they will be able to open, read, and make copies of the file. If a person or group has permissions to write to a file, they can also modify and delete the file. If they if have permissions to execute a file, and it is an application that can be run, they will be able to run the application. If it is a directory, they will be able to go into it.

After the first ten characters, the following information is displayed:

- The name of the owner of the file (the person who created it).
- The name of the group to which the file belongs.
- The size of the file in bytes.
- The time and date on which it was last modified.
- The name of the file itself.

If the file is a symbolic link, in place of the name of the file, it will show the pathname of the file or directory to which the link points.

WHAT IS A SYMBOLIC LINK?

A symbolic link is simply a pointer to another file or directory. Opening the link is the same as opening the original file or directory, except you don't have to navigate to where it actually resides in order to open it. This saves a great deal of time jumping back and forth between directories.

You can create a symbolic link with the *ln* (link) command. For example:

```
ln -s /usr/staff/john/Papers link_to_Papers
```

creates a link in the current directory (called "link_to_Papers") which automatically opens the directory */Papers* whenever you open the link.

If a link is deleted, the original remains intact.

5.6 DETERMINING THE CONTENTS OF A FILE

F This command tries to determine the type or contents of the file, such as: ASCII, binary, etc.

5.7 MAKING A FILE WRITABLE

Ctrl W makes the currently highlighted file writable. You can only do this if you have owner or group permissions to write (make changes) to the file.

5.8 MAKING A FILE EXECUTABLE

Ctrl X makes the currently highlighted file executable. You can only do this if you have owner or group permissions to write (make changes) to the file.

6 DISPLAYING FILES USING LESS

6.1 DISPLAYING THE CONTENTS OF A TEXT FILE

FROM SPY

If the current file is not a directory, typing D displays the contents of a text file using a program called *less*.

FROM UNIX

Typing the command:

```
less filename
```

displays the contents of a text file called *filename* using *less*.

6.2 LESS COMMANDS

When *less* displays a file, you can use the following single-key commands, which are similar to *spy* and *vi*:

J or Enter Scroll down one line.

K Scroll up one line.

F or Space Scroll forward one page.

B Scroll backward one page.

Shift G Goes to the end of the file.

1 G Goes to the beginning of the file (*Line#* then G goes to that line).

/`string` Search for *string* in the file, and display that line.

N Go to next occurrence of *string*.

H Display help, including the commands listed here.

Q Enter Quit *less* and return to *spy* or UNIX

E or V Edit the current file using *vi*. If you do not know how to use *vi*, you can get out of it by typing : Q ! Enter. Then read the chapter *vi - Text Editor* p. 313.

Note: Once you are done editing in *vi* (i.e. you exited *vi* with the : W Q Enter command) you are returned to *less*, which still displays the file. If you want to get back to *spy*, you will still need to quit *less* by typing Q Enter.

7 UNIX COMMANDS FROM SPY

7.1 THE UNIX KEY (!)

Although many things can be done from within *spy*, starting Houdini or other programs is often done with UNIX commands.

While you are in *spy*, any time you need to execute a UNIX command, simply precede the command with the ! key, and then type the command. For example:

!`df -k` Enter

displays the amount of free disk space in kilobytes, using the UNIX command *df*.

7.2 THE TAG CHARACTER (%)

UNIX commands often require the name of a file(s) as arguments. We can provide these filenames from *spy* by using the `%` character within our UNIX command. Anywhere this character is found, it will insert the name(s) of the currently tagged files.

For example, if the files *1.pic*, *2.pic* and *3.pic* are tagged, then the command:

!`iplay %` Enter

translates into:

!`iplay 1.pic 2.pic 3.pic` Enter

Note: If no file(s) are tagged, a command using the % character will act upon the currently highlighted file only.

For more information on UNIX commands, see the chapter *UNIX* p. 319.

8 COMMAND HISTORY

8.1 EXECUTING A UNIX COMMAND

!*cmd* Enter The command *cmd* is executed.

For example:

```
!df Enter
```

displays the amount of free disk space.
For more information on UNIX commands, see the chapter *UNIX* p. 319.

8.2 REDO THE LAST COMMAND

!! Enter Re-executes the last UNIX command.

8.3 REDO THE LAST COMMAND STARTING WITH...

!!*string* Enter Re-executes the last command which started with the string *string*. For example, if a recent command was:

```
!imgview test.pic Enter
```

then the command

```
!!im Enter
```

re-executes it.

8.4 COMMAND HISTORY DISPLAY

!? Displays the command history – a list of the most recent UNIX commands you've executed. As in the jump history, you can move down and up the list with J and K. Movement forward and backward along one line is accomplished with the Space and Bksp keys.

EDITING THE COMMAND HISTORY

Once the list of command history is displayed, you can edit the line with the following commands, similar to the *vi* editor:

Enter Typing the *Enter* key executes the current command as-is.

/*string* Enter Moves the cursor to the next command starting with *string*.

A Adds text after the location of the cursor until you type Esc or Enter.

I Inserts text before the location of the cursor until you type the Esc or Enter key.

X Deletes the character in front of the cursor.

Shift C Replaces text after the cursor with text you enter, until you type Esc or Enter.

D W Deletes the current word.

C W Replaces current word with text you enter until you type the Esc or Enter keys.

Ctrl C You can abort the jump at any time by pressing Ctrl C or Del depending on your setup.

! Esc Displays the most recent command and allows the above editing commands to be used. Then, when you press Enter, the command is executed. As above, Ctrl C or Del aborts the command.

9 HELP & OTHER SPY COMMANDS

COMMAND HELP

`?` Lists all single keystroke commands, as described in this chapter.

REDRAW

Ctrl L Redraws the screen if it has been corrupted by other programs sending text to the same window shell. If you are ever unsure that what you see in the *spy* window is correct information, this command ensures that it is up-to-date.

LIST CURRENT PROCESSES

Ctrl P Lists the processes you are running. You can use the UNIX *kill* (see *Cancel a Process* p. 329) command to kill these processes.

QUITTING SPY

Shift Q Quits *spy* and leaves the jump history and the command history in the file *.spyHistory* file your home directory.

9.1 SPY NUMBER KEY SHORTCUTS

spy is set up with several default key mappings, as follows:

1 Equivalent to: `hview % Enter` (where % is the highlighted file)

2 Equivalent to: `houdini % Enter`

3 Show CPU system activity via *gr_osview*.

4 Show most active CPU process via *gr_top*.

Shift M Read mail using current mail program (usually *elm* or *MediaMail*).

10 CUSTOMIZING SPY

10.1 THE .SPYRC FILE

A *.spyrc* file in your home directory allows you to customise *spy* with a list of key-mappings. Each entry assigns a commonly used command to a single keystroke. For example, the default *.spyrc* file assigns *hview* to the 1 key, and *launch Houdini* is assigned to the 2 key.

Edit the *.spyrc* file with a text editor (*vi* or *jot*) to add your own commonly used commands. Do this with the *map* command. For example:

`map 9 unix jot %`	Open selected file with *jot* when you type the 9 key.
`map 8 jump =$HFS`	Jump to directory held by variable "$HD".
`map > jump :?`	Show jump history.
`map ? unix_cmd :?`	Show UNIX command history.

Be sure to map only those keys which *spy* does not already use, or you will have trouble executing certain *spy* commands.

Use Ctrl R to reload the *.spyrc* file after you have saved your changes.

Note: Keywords (i.e. *jump*, *unix_cmd*) must be in lowercase, and must have a space after them in order to function.

RELAXPROMPT

Including the command:

```
relaxprompt
```

in your *.spyrc* file stops the *Press Enter to Continue* messages from appearing. It also lets the *spy* "Continue:" prompt accept any *spy* command immediately after the previous command finishes without having to type Enter first.

RELAXSEARCH

There is a *.spyrc* command to determine the behaviour of directory searches performed with the / key. By including the command:

```
relaxsearch
```

in your *.spyrc* file, *spy* will find patterns in the middle of filenames as well as at the beginning of the filenames. For example, if you search for "12" and the directory contains:

123.rc *flame123.pic* *flame212.pic*

then each of these files would be found in turn.

Without *relaxsearch*, you would have to search for "*12" to get the same behaviour.

If you set the relaxsearch mode and want to find only filenames beginning with "12" then search for "^12". This works because the leading '^' is a standard search pattern character meaning "the beginning of a line".

COLOR

You can customise the display of the file listings such that files are displayed in different colors. This only works if the terminal you are running *spy* from is an ANSI terminal. To check if your terminal is an ANSI terminal, use the following UNIX command:

```
echo $TERM
```

To customise display of colors with your *.spyrc* file, use this command:

```
color filenamePattern color
```

where *filenamePattern* is the string to match (you can use standard *, [] and ? as wild-card characters), and *color* is one of the colors in the list below.

For *filenamePattern* , you can also specify certain file types:
-dir for directory, *-x* for executable, *-tagged* for selected files.

allowable color names

The *color* parameter must be one of :

- white — reverse_white
- red — reverse_red
- yellow — reverse_yellow
- green — reverse_green
- cyan — reverse_cyan
- blue — reverse_blue
- purple — reverse_purple
- black — reverse_black

example

If you added something like:

```
color *.hip purple
color -dir yellow
color core reverse_red
color -tagged reverse_blue
color -x green
```

to your *.spyrc* file, it would color all .hip files purple, all directories yellow; all "core" files in reverse red; all tagged (selected) files in blue; and all executable files (as in UNIX permissions) green.

COLORTOGGLE

There is also a command *colortoggle* which toggles the display of color. You can map this to another key, such as uppercase "C", in your *.spyrc* file by using:

```
map C colortoggle
```

in your *.spyrc* file. The Shift C key combination is the default for turning the display of color on and off.

10.2 IGNORE MASKS

IGNOREMASK

Sometimes it is desirable to exclude the display of certain file types. This can be accomplished using ignore masks. To set up an ignore mask, use the .spyrc command *ignoremask*. The syntax for this command is:

```
ignoremask [mask] [group]
```

The mask is a pattern string which follows the same rules that spy uses when searching for files within a directory (See *Searching Within a Directory* p. 296). The group number is optional. If no group number is specified, group 0 is used.

examples

Here is an example of an ignore mask that tells *spy* not to display all files starting with a period (" . ").

```
ignoremask .*
```

A maximum of 256 ignore masks can be specified in the *.spyrc* file. Each ignore mask command only accepts a single mask string. However, masks can be accumulated by using the same group number. For example:

```
ignoremask Makedepend 1
ignoremask *.o 1
```

This means that when group 1 is in use, all files matching the patterns *.o and Makedepend will not be shown. *spy* has a limit of 64 groups numbered from 0 to 63.

IGNORETOGGLE

The use of the ignore groups can be toggled using the key command *ignoretoggle*. The syntax for this command is:

```
ignoretoggle [group]
```

It is possible to map a key to toggle the display of a particular group by adding the following to the *.spyrc* file.

```
map [key] ignoretoggle =[group]
```

Here is an example which maps the o key to toggle the display of all .o files in a directory:

```
ignoremask *.o 1
map o ignoretoggle =1
```

IGNOREDEFAULT

Finally, it is possible to specify whether a group will be in use on starting *spy*.

```
ignoredefault [group] [use]
```

If this command is not present in the *.spyrc* file, all groups will be used. The *use* parameter should be set to 0 if this group should not be used on startup and 1 otherwise.

10.3 REMAPPING SPY KEY FUNCTIONS

In general, you should never need to remap the *spy* key functions, but if for some reason you would like to, you can use the *map* command to reassign the keys *spy* uses with the following list of Key Functions.

For example, if you were using a Dvorak keyboard layout, but wanted to maintain the six directory navigation keys in the same physical location, you could add the following mappings to your *.spyrc* file:

```
map d left
map h down
map t up
map n right
map g climb
map e display
```

LIST OF SPECIAL KEY NAMES

`<left>`	←
`<right>`	→
`<up>`	↑
`<down>`	↓
`<s-left>`	Shift ←
`<s-right>`	Shift →
`<home>`	Home
`<end>`	End
`<pagedown>`	PgDn
`<pageup>`	PgUp
`<insert>`	Ins
`<f1..f12>`	F1 - F12

example

To get *spy* to use the arrow keys instead of J K H L, add this into your *.spyrc* :

```
map <left>   left
map <right>  right
map <up>     up
map <down>   down
map <enter>  display
map <s-left> climb
```

LIST OF SPY KEY FUNCTIONS (ASSIGN TO A KEY)

Key Function	Description
climb	Move to parent directory
colortoggle	toggles the display of color
command	Execute *spy* command
copy	Copy file(s)
date	Show current date and time
display	Show file or enter directory
down	Move cursor down

drop	Drop taken file(s) (Paste)
empty	Empty taken files(s)
enter	Edit file or enter directory
experience	Toggle user experience
file	Determine type of file(s)
help	Shows Help. List commands mapped in *.spyrc*
helpdir	Toggle help directory
home	Go to home directory
ignoredefault	default mask to use
ignoremask	Set mask for filename patterns to ignore
ignoretoggle	Toggle ignore mask group
invalid	Undefined key
inventory	List taken file(s) (show Clipboard)
jump	Jump to file or directory
keys	List key assignments
left	Move cursor left
loadrc	Re-load *.spyrc* file
longlist	Long listing of file(s)
move	Move file(s)
next	Repeat previous name search
nextfile	Move cursor to next file
pagedown	Go to next page
pageup	Go to previous page
patternpick	Select (pick) files by pattern match
pick	Select (pick) or unselect file (toggle)
previous	Move cursor to previous file
quit	Exit *spy*
redraw	Re-draw the screen
remove	Remove selected file(s)
right	Move cursor right
search	Search directory for name
setenv	Add a new Environment variable
showdetail	Toggle directory detail
showmemory	Show memory usage
startshell	Start another shell
take	Take file(s) (Copy)
unix_cmd	Execute UNIX command
unpick	Turn all selection (pick) flags off
up	Move cursor up
version	Display *spy* Version Number
visibility	Toggle visibility
who	Lists users that are currently logged in

2 vi - Text Editor

1 INTRODUCTION TO VI

The IRIX OS comes with two main editors: *vi*, and *jot*. Jot is a GUI-based editor you should use if you use the IRIX Magic Desktop. *vi* provides tighter integration with UNIX commands and *spy*. It also provides a powerful lowest-common-denominator.

There are two other popular editors you should know about: *nedit* and *vim*. *nedit* combines the power of *vi* with a friendly graphical interface. The caveat is that it only comes with IRIX 6.3 and higher. If you have IRIX 6.3 – this is the editor of choice. The other editor – *vim* is an excellent shareware text editor which is a super-set of *vi* with many extra powerful features. If you like the way *vi* works, it would be well worth your while exploring a few shareware sites to obtain a copy of *vim*.

This section provides condensed descriptions of the complete *vi* command set. For detailed information, see the *SGI_EndUser* online book (available from the *Irix Desktop:Help* menu) for the *IRIS Utilities Guide:Appendix A – vi Text Editor*.

1.1 STARTING VI

You can open a file in *vi* from *spy* by selecting the file and typing V. To launch *vi* explicitly with a UNIX command, type one of the following:

`!vi` *filename* Enter
`!vi +/`*string filename* Enter

If the file *filename* does not exist, a new one is created. If the file exists, it is opened for editing. If there are several names, *vi* edits them in sequence. The + option opens the file at the line which contains *string*.

1.2 QUITTING VI

QUIT VI

:Q Enter Quit *vi* whether you have saved or not.

QUIT VI AND SAVE

:W Q Enter Quit *vi* and save (write) your changes.

QUIT VI GUARANTEED

Esc :Q ! Enter Leaves Input mode, and quits without saving modifications.

2 VI – COMMAND MODE

vi operates in two modes; *Command mode* and *Input mode*. In Command mode, you can move the cursor around, as well as cut, copy and paste text.

2.1 MOVING AROUND

Use the following key commands to move around in *vi* while in Command mode:

J or Enter	Down
K or -	Up
H or Bksp	Left
L or Space	Right
W	Move to next word
B	Move to previous word
E	Move to end of current word
Shift W	Move to next word after the next space or tab
$	Move to end of line
0 (zero)	Move to beginning of line
Shift H	Move to top of the screen
Shift M	Move to middle of the screen
Shift L	Move to bottom of the screen
Ctrl F	Go forward one screen
Ctrl B	Go backward one screen
5 Ctrl F	Go forward five screens
*35*G	Go to line 35
:*35* Enter	Go to line 35 (alternate method)
*35*J	Move down 35 lines
Shift G	Go to the last line
:$ Enter	Go to the last line (alternate method)
/*string*	Search forward for *string*
Shift /*string*	Search backward for *string*
N or Shift N	Move to next or previous occurrence of *string*

Tip: If the cursor does not move, you may be in Input mode. Type Esc to get out of Input mode and back to Command mode – the cursor keys should then work.

2.2 COPYING AND PASTING TEXT

The following commands can be used to cut, copy, and paste text:

X	Cut the character ahead of the cursor
Shift X	Cut the character behind the cursor
D D	Cut (delete) the current line
12 X	Cut the next 12 characters
12 D D	Cut the current line and the next 11 lines
D Shift G	Cut to the end of the file
D 1 Shift G	Cut to the beginning of the file
D / *hello* Enter	Cut to the next occurrence of the word *hello*
Shift D	Cut from the current character to the end of the line
Y Y	Copy (yank) the current line to the clipboard
Y Shift G	Copy (yank) all lines from the current line to the end of the file into the clipboard
10 Shift Y	Copy (yank) the next ten lines into the clipboard
Y *4* W	Copy (yank) four words
Y / *hello* Enter	Copy (yank) everything up to the next occurrence of the word *hello*
P	Paste the contents of the clipboard after the current line
Shift P	Paste the clipboard before the current line

Tip: Text that is deleted is automatically put into the clipboard. You can subsequently use a paste (P) command to insert this text elsewhere.

2.3 MISCELLANEOUS COMMANDS

U	Undo the last operation
.	Repeat the last command
~	Switch case from upper to lower or vice versa
>>	Indent the current line
<<	Outdent the current line
%	Move to a matching parenthesis or bracket
Shift J	Join the next line with the current line (erases a carriage return)
Ctrl L	Redraw the screen

3 VI – INPUT MODE

Text is entered in Input mode. In input mode you can add text by appending, inserting and replacing. The Esc key returns you to Command mode where you are able to move the cursor, and copy and paste text.

3.1 ADDING AND CHANGING TEXT

Use these keys to add text. Once you are done, use Esc to exit to Command mode.

A	Add text after the current character
Shift A	Add text after the current line
C C	Change the current line
I	Insert text before the current character
R	Replace the current character (Note: no Esc necessary)
12 S	Substitute text in place of the next 12 characters
C F *x*	From the current character to the next occurrence of character *x*, replace it with the typed text
O	Open a new line after the current line and insert text
Shift O	Open a new line before the current line and insert text
Shift J	Join two lines
! ! *date*	Replace current line with output from the UNIX date command
! Shift G *fmt* Enter	Pass all lines from the current line to the end of the file to the UNIX *fmt* command, and replace them with the output of *fmt* (fmt formats it's input into simple paragraphs).

EXAMPLES

A `hello` Esc Add the word *hello* after the current character.

I `hello` Esc Insert the word *hello* before the current character.

1 2 S *hello* Esc Substitute the word *hello* in place of the next 12 characters.

C F X *hello* Esc From the current character to the next occurrence of the character *x*, replace it with the word *hello*.

S *hello* Esc Substitute the word *hello* instead of the rest of the current line.

O *hello* Esc Open a new line after the current line and insert the text *hello*. Before the Esc, you can type multiple lines of text, followed by Enter at the end of each line.

Shift 0 *hello* Esc Open a new line before the current line and insert the text *hello*.

! ! `date` Enter Replaces current line with the output of the UNIX *date* command.

4 READING AND WRITING VI FILES

Use the following commands from Command mode to read and write files:

:R *name* Enter	Read the file *name* so it is appended after the current line
Shift Z Z	Write out the current file and exit
:W Enter	Write out the current file
:W! Enter	Write out the current file, even if it is read-protected
:W *name* Enter	Write out the current file into filename *name*
:N Enter	Edit the next file in the list of files being edited *
:E *name* Enter	Edit the file *name*
:E# Enter	Go back and edit the most recently edited file
:Q! Enter	Exit *vi* without saving anything

* the list of files comes from the command line. See *Starting vi* p. 313.

5 EDITING VERY LARGE VI FILES

If you edit very large files (greater than 2-3 MB), such as an *uuencoded* file sent in email, you may find that *vi*'s default temp directory is too small to adequately hold the temporary files that are created.

This happens because, by default, *vi* uses the */tmp* directory for the temporary edit file, and on some SGI systems, the root file system *(/)* is quite small, and frequently has very little disk space available (to find the one with the most space, use the *df* UNIX command).

You can change the directory that *vi* uses by setting the *directory* variable from within *vi*. Simply type:

```
:set directory=/var/tmp Enter
```

from command mode to set the temp directory to another place. A good place is often the */var* directory, as the */var* partition is usually much larger and has more free space.

This can be set permanently in your *~/.exrc* file so that each time you run *vi* (like when you edit email in *elm*) *vi* will use the bigger directory.

Tip: If editing very large files, or files with very long lines, you may also want to consider using *jot*, which is more efficient at editing those types of files.

3 UNIX

I UNIX INFORMATION

I.1 INTRODUCTION

This section provides a cursory introduction to the UNIX operating system. The key things you need to know about UNIX are:

- how to login
- the UNIX file system
- using the C shell and basic UNIX commands

WHAT TO READ

The following topics are most relevant for Houdini users:

- regular files, directories, links, symbolic links, root
- permissions, owners, groups
- mounted file systems, partitions
- user's home directory

FURTHER REFERENCE

For more information on how to use UNIX, we'd recommend the book:

UNIX for Dummies (2nd Edition)
John R. Levine & Margaret Levine Young,
IDG Books Worldwide, Foster City (California), 1995, ISBN: 1-56884-905-2.

I.2 GETTING A SHELL IN WINDOWS

- The standard install of Houdini gets you a Windows native version of csh (tcsh).
- After selecting: *Start > Houdini > Command Line Tools*, type: *csh* to get a shell. If you don't, it will be a DOS shell.
- However, since you may not have *cygwin* installed, you'll have to type the command: *sourcerc* in the csh in order to get usable subset of the standard commands you usually find in: */bin* like *ls* (!). Use *printrc* to see what *sourcerc* actually does.
- To get a really usable csh, you'll need to download and install cygwin, and set your $PATH variable to use it. You can obtain cygwin from: *http://www.cygwin.com/*

2 UNIX QUICK HELP REFERENCE

ls -l	List files in current directory w/date+size
ls *Jo** *or* ls **.tif*	List files that start with *Jo* or end in *.tif*
ls -l | more	List files, and pause when screen is full
cd *directoryName*	Change Directory (specify full path)
cd *./directoryName*	Change Directory (the name is one listed in the current directory with "ls")
cd ..	Go up one directory level
cd ~	Change to your Home directory
pwd	Display which directory you're currently in (pwd = "Present Working Directory")
cp *fileName directoryName*	Copy file (*cp -r * /dirName* for recursive)
mv *oldName newName*	Rename file
mv *originalName directoryName*	Move a file to a new directory
mkdir *directoryName*	Make a new directory
rm *fileToErase*	Remove (erase) a file
rm -rf *DirectoryToErase*	Erase a directory and everything in it ***Note:*** Use with extreme Caution!!!
ln -s *originalFile linkedName*	Create a symbolic link to open the original file whenever *linkedName* is used.
ln -s */usr/File linkToFile*	Create a symbolic link in the current directory to the file */usr/File*

EXTRA COMMANDS

find */usr* -name "*Jo**" -print	Find files starting with "Jo*" contained anywhere in the */usr* directory
less *TextFileToView*	View a text file (type Space to move forward a page; B to go back a page; and Q to return to the UNIX shell)
vi *TextFileToEdit*	Edit a text file with *vi*
jot *TextFileToEdit*	Edit a text file with *jot*
iplay *ImageFileToDisplay*	View an image (SESI viewer (.pic okay))
imgview *ImageFileToDisplay*	View an image (SGI's viewer (no .pic))
rlogin *bern* -l *john*	Remote login to "bern" (Ctrl D to close)
man *commandName*	Read the man pages about a command
df -k	Display Free disk space in kilobytes
du -k	Display disk Usage in kilobytes
gr_osview	Show graphic overview of sys. activity
gr_top -p 7	Display top processes using system CPU
ps -ef | grep "*commandName*"	Display pid number of a process
kill -9 pid#	Kill a currently active process (Caution!)
Ctrl C	Cancel a command
Ctrl S	Pause display to screen
Ctrl Q	Resume display to screen

3 THE UNIX FILE SYSTEM

3.1 SPECIAL UNIX DIRECTORIES

There are many special directories within a UNIX file system. Most of these are specific to your site and installation. However, there are several directories common to almost all UNIX installations. These include the following:

`/`	This is the *root* or top level directory.
`/usr`	This is the *user* directory. Often a mounted file system with a large portion of UNIX files.
`/tmp`	This is a temporary directory into which anyone can write. When the computer is rebooted, all files in */tmp* are removed.
`/usr/tmp`	This is a temporary directory on the */usr* file system.
`/dev`	UNIX's way of accessing connections to devices like CDROMs and tape drives is by treating them as a file.
`/etc`	Contains machine configuration, startup setup files.
`/bin`	Contains the core set of UNIX commands.
`/usr/bin`	Contains more UNIX commands.
`/usr/bsd`	Contains the University of Berkely UNIX commands.
`/usr/sbin`	Contains SGI UNIX commands.
`/usr/lbin`	Contains additional SGI UNIX commands.
`/usr/spool/`	Contains news, printer, and network files.

LOCAL OR REMOTE?

A file on your machine seen from your machine might not have the same path as a file on your machine seen from another machine. For example, the file:

```
/usr/people/john/readme.txt
```

might appear as:

```
/n/machineName/people/john/readme.txt
```

from another machine. Items local to your machine use */usr* when you are on the machine itself (or if you rlogin). These same items appear as */n/machineName/* when viewed from elsewhere on the network.

Note: This is only one of many possible ways that your UNIX network may be setup. You should consult your system administrator for details on how your system has been setup.

3.2 FILE PERMISSIONS AND OWNERSHIP

Each file can be assigned to a specific owner or group. The owner or group may have special permission or authority regarding reading, writing and executing the file. Permissions control who can read and modify files. Because it is often desirable for several people to have access to a file, permissions can be set for an individual (the owner), a group, and for everyone else. See the section *Listing Files with Detailed Information* p. 301 for how to list permissions from within *spy.*

DISPLAYING PERMISSIONS

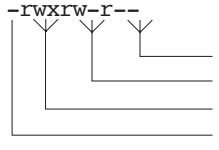
You can use the *ls -l* command to list detailed permissions information for all files in the current directory. Specifying a filename will list permissions only for that file.

example

```
-rw-rw-r-- john prisms    318 May 22 19:59 testFile
```

To also list hidden files (files whose names start with a " . ") you should use *ls -la* .

The permissions of a file are displayed via the first ten characters. They provide the following information:



Permissions	Type
`-` no permission	`-` file
`r` readable	`d` directory
`w` writable	`l` link
`x` executable	

The first character tells if it is a file, directory, or symbolic link. Then come three sets of three characters. These show the Read/Write/Executable (rwx) permissions of the file for the owner, the group to which the file belongs, and of all other users.

If a person or group has permissions to read a file, they will be able to see it when they attempt to display a listing of files, also they will be able to open, read, and make copies of the file. If a person or group has permissions to write to a file, they can also modify and delete the file. If they if have permissions to execute a file, and it is an application that can be run, they will be able to run the application.

After the first ten characters, the following information is displayed:

- The name of the owner of the file (the person who created it).
- The name of the group to which the file belongs.
- The size of the file in bytes.
- The time and date on which it was last modified.
- The name of the file itself.

If the file is a symbolic link, in place of the name of the file, it will show the pathname of the file or directory to which the link points.

what is a symbolic link?

A symbolic link is simply a pointer to another file or directory. Opening the link is the same as opening the original file or directory, except you don't have to navigate to where it actually resides in order to open it. This saves a great deal of time jumping back and forth between directories.

You can create a symbolic link with the *ln* (link) command. For example:

```
ln -s /usr/staff/john/miscellaneous/Papers link_to_Papers
```

creates a link in the current directory (called "link_to_Papers") which automatically opens the directory */Papers* whenever you open the link.

Links are not actually copies of the file or directory. If you delete the link, the original will stay intact.

4 COMMON UNIX COMMANDS

Following, are the UNIX commands most useful to Houdini users. The Silicon Graphics UNIX manuals contain more detailed descriptions of these commands.

Commands with a single stroke equivalent in *spy* are noted with (*spy:* C) where *c* is the *spy* character equivalent. If running UNIX commands from *spy*, remember to type a `!` character before typing the UNIX command.

Most of these commands can also be performed via IRIX. See the section *IRIX File Management* p. 337 for details.

Warning: All UNIX commands and filenames are case-sensitive. That means it's very important to always use uppercase or lowercase letters exactly as shown. Mixing upper and lowercase letters in UNIX will cause many errors.

4.1 FILE MANAGEMENT

The file management commands allow you to manipulate files by moving, copying, deleting and other such operations. For an alternative way to do these functions, see the section *IRIX File Management* p. 337.

LISTING FILES

`ls` List the names files in a directory with details. (*spy:* L)

Example:

```
> ls -l *
-rw-rw-r-- john prisms    318 May 22 19:59 .spyrc
-rw-rw-r-- john prisms    812 Jun 10 21:00 .zmailrc
-rwxrw-r-- john prisms   18860 May 9 14:49 Maker
```

CHANGING DIRECTORY

`cd` Changes the current directory to the one specified. (*spy:* Shift J)

`cd /usr/bin`	changes directory to: */usr/bin*.
`cd ~`	changes current directory to: your home directory.
`cd..`	changes current directory to: the parent directory.

Note In UNIX you can use . and .. at the beginning of a pathname to mean "the current directory" and "the parent of the current directory" respectively.

See *Going Home* p. 298 for an explanation of the home directory.

COPYING FILES

`cp` Copy a file or files to a directory (*spy:* C).

```
cp butterfly1.pic /tmp/test.pic
cp *.pic /tmp
```

The first example copies the file *butterfly1.pic* into the directory */tmp* and names the copy *test.pic*. The second example copies all files ending in *.pic* to */tmp* .

LINKING FILES

`ln` Link a file to another location, without using space for each file. For example:

```
ln –s /usr/staff/john/miscellaneous/Papers link_to_Papers
```

creates a link in the current directory (called "link_to_Papers") which automatically opens the directory */Papers* whenever you open the link. See *What is a symbolic link?* p. 302 for a an explanation.

MOVING / RENAMING FILES

`mv` Move file to a new location in the file system, or rename a file (*spy:* M).

```
mv butterfly1.pic oldButterfly.pic
mv *.pic /tmp
```

The first example renames the files butterfly1.pic to oldButterfly. The second example moves all files ending in *.pic* into the directory called: */tmp*

DETERMINING THE CURRENT DIRECTORY

`pwd` Present Working Directory. Displays name of directory you are currently in.

Example:

```
> pwd
/usr/staff/john
```

MAKING A DIRECTORY

`mkdir` Create new directories.

Example:

```
mkdir MyDirectory AnotherDirectory
```

REMOVE DIRECTORIES

`rm` Remove files or directories. The option *-r* does this recursively. i.e. it will also remove any files and subdirectories contained by the directory (*spy:* Shift R).
The *-rf* option does this recursively without confirming the action.

Example:

```
rm oldButterfly.pic
rm –rf MyDirectory
```

Warning: The *-rf* option forces the removal of all files or directories without a prompt! Everything contained within the directories will be immediately erased!

COPYING LARGE DIRECTORIES OF FILES

To copy large amounts data across a network, it is much faster to use the *tar* command (usually used for making tape backups) in conjunction with a pipe command (|) than to make a straight copy with the *cp* command. This technique also has the added advantage of keeping all file permissions and dates unchanged.

```
tar cvf - source | (cd /destination; tar xvf -)
```

For Example:

```
tar cvf - Images | (cd /n/saba2; tar xvf -)
```

Moves the directory */Images* and all its contents to a new directory in */n/saba2* .

Note: The " - " is important in order to parse *stdin* and *stdout* through the pipe.

EDITING TEXT FILES

`vi` Edit a text file using *vi*. See *vi - Text Editor* p. 313 for details.

Example:
```
vi Readme.txt
```

DISPLAY FILE CONTENTS

`cat` Concatenate files. This displays the contents of the files on the screen. This can be paused and resumed with the Ctrl S and Ctrl Q keys. You can stop it altogether by typing Ctrl C. Alternately, you can redirect the result to a new file with the UNIX redirection (>) symbol.

Example:
```
cat file1.txt file2.txt
cat file1.txt file2.txt > file1-2.txt
```

The first example displays the result of concatenating file1.txt and file2.txt on the screen. The second example concatenates file1.txt and file2.txt and pipes the result into the files *file1-2.txt*.

FILE PERMISSIONS

`chmod` Change the read/write/execute permissions of files.

Example: to make a file writable by all users:
```
chmod +w butterfly1.pic
```

If you make a file read-only, you will not be able to make changes to it, unless you make it Writable again.

FILE OWNERSHIP

`chown` Change the owner of file (only possible if you own the file, or as superuser).

Example:
```
chown animate *
```

Changes the owner of all files in the current directory to *animate*.

GROUP OWNERSHIP

`chgrp` Change the group ownership of files.

Example:
```
chgrp animate *
```

Changes the group of all files in the current directory to *animate*.

4.2 FILE OPERATIONS

File operations help you to view the differences or the contents of a file. The file operations do not alter the contents of the file, however, they will give you valuable information regarding the contents of the directory, size of the file and differences between two files.

DISPLAY CONTENTS OF FILE

`less` Display the contents of a file page by page. (*spy:* D)

Example:
```
less README
```

See *Displaying Files Using less* p. 303 for details.

SEARCH FILES FOR A STRING

`grep` Search file(s) for a specific string and display the lines that contain it.

Example:
```
grep hello Read*
```

DISPLAYING THE DIFFERENCES BETWEEN FILES

`diff` Compare one text file to another and display the differences. This doesn't work well with binary files.

Example:
```
diff Readme.txt Readme2.txt
```

COMPARE FILES

`cmp` Compare two binary files, byte for byte and display the differences.

Example:
```
cmp -l mantra.pic /tmp/test.pic
```

SORTING ALPHABETICALLY OR NUMERICALLY

`sort` Sort the lines in a file alphabetically or numerically.

Example:
```
sort Readme.txt
```

DETERMINING FILE SIZE DIFFERENCES

`sum` Make a checksum of all files. The checksum is useful to determine if two files are the same if they are on different machines or *cmp* is usable.

Example:
```
sum *.pic /tmp/*.pic
```

4.3 UNIX PROCESSES

In UNIX, you can run many applications simultaneously, because UNIX is a multiprocessing operating system in which each application is a separate process.

WHICH PROCESS?

Each process running on a UNIX machine has a process ID number attached to it. Using this command lists out which processes are running, and their ID numbers.

`ps` Display information about which processes are running, who they are being run by, their process ID (useful with *kill*) and for how long the process has been running.

Example:
```
ps -lae
```

You can filter the information to find a specific process. For example, to find a process called *crabs*, you could type:

```
ps -ef | grep "crabs"
```

Assuming a process called *crabs* is running, this will display its process ID number.

CANCEL A PROCESS

`kill` Terminate a process which is currently running.

Example:
```
kill -9 processNumber
```

Where *processNumber* is one of the ID numbers listed by the *ps* command. The *-9* ensures that the process will be killed no matter what.

CPU ACTIVITY GRAPHIC DISPLAY

`gr_osview` Opens an IRIS window to display the IRIS activity: CPU time consumed by users and the UNIX system, I/O, swapping time. Example:

```
gr_osview -a
```

CPU ACTIVITY DISPLAY

`gr_top` Continuous display of all processes that use more than 1% of CPU time. To get a version with smaller text, use: *gr_top -p 7* .

CURRENT LOGIN LIST

`who` Lists who is logged in to the system right now.

Example:
```
> who
fred
rick
```

4.4 DISK MANAGEMENT

CURRENT DISK USAGE

`du` Disk usage: reports how much space a directory and sub-directories use in system blocks (usually 512 bytes). Use the *-k* option to report it in kilobytes (1024 byte blocks). Use the *-s* option to get a summary of tagged files and directories. Example:

```
du -sk *
```

FREE DISK SPACE

`df` Disk free: shows the space available on the disk in blocks (usually 512 bytes). Use the `-k` option to report display it in kilobytes (1024 byte blocks). Example:

```
df -k
```

4.5 FILE COMPRESSION

COMPRESSING FILES

When sending files in e-mail, it is usually desirable to compress the files into a smaller format so that sending charges and times are reduced.

To compress a file(s) for delivery in email, tag the desired files in *spy* and enter:

```
tar cvf sendfile.tar %
compress %
uuencode % % > %.uue
```

UNCOMPRESSING FILES

To decompress a file(s) after receiving them in email, select the file in *spy* and enter:

```
uudecode %
uncompress %
tar xvf %
```

5 TAPE BACKUPS

REMOTE TAPE ARCHIVES

Backups to tape are normally done with the UNIX *tar* (tape archive) command which saves all the files in a directory and its subdirectories onto tape. *tar* is also used to report and extract files from tape. *tar*, by default, uses the tape drive, but can also be used to archive and extract from a single disk file or from standard input and output *(stdin, stdout)*.

- To create a new tape from the hard disk. Ensure the directory is highlighted in *spy*, type ! to get to UNIX, and type:

  ```
  tar cvf iris:/dev/tape %
  ```

  where *iris* is the name of the machine where the tape drive is located (the host), and */dev/tape* is the device you are using (the device). You should substitute the name of the host you will be using in place of *iris*. The device is usually a DAT.

- To view the contents of a tape, type ! to get to UNIX, and type:

  ```
  tar tvf iris:/dev/tape
  ```

  the contents of the tape are listed on your screen.

- To restore an entire tape back onto your hard disk, ensure that you are in the directory to which you want the files restored, type ! to get to UNIX, and type:

  ```
  tar xvf iris:/dev/tape
  ```

LOCAL TAPE ARCHIVES

In the above examples, the *f* parameter (as in *cvf* and *xvf*) specifies that *tar* should use what follows as the machine for where the tape is connected (i.e. *iris:/dev/tape*). We can then see that because of the *f* parameter, the *tar* command looks to the machine called *iris* for a tape drive. If the tape drive is instead connected directly to your own machine, you no longer need the *f* option, nor the machine specifier. This yields a slightly different command for saving from the hard disk to tape when the tape drive is connected directly to your machine:

```
tar cv %
```

This saves the currently highlighted file or directory onto a local DAT tape. Similarly, the command to restore a tape to disk becomes:

```
tar xv
```

For more information on the *tar* command, see the UNIX man pages.

THE NOHUP OPTION

The *nohup* option means "No hangup". Use this to run a UNIX command and the command will continue even after you logout. This is useful to continue a tape backup or a render script after you logout. For example:

```
nohup tar cvf fiji:/dev/tape %
```

THE BADTAR COMMAND

The *tar* command sometimes encounters tape errors which cause it to quit in the middle of reading your files. The *badtar* command attempts to skip files in which errors occur. It is usually run with the *dd* command. For example:

```
dd if=/dev/tape conv=noerror | badtar xvf -
```

attempts to retrieve files from the tape to the current directory, and skips any files which contain errors.

AN EASIER WAY?

There is an easier way to perform tape operations. Refer to the section *Tape Backups* p. 344.

6 ENVIRONMENT VARIABLES

Some UNIX environment variables are provided for users who login with the standard *HOUDINI* setups. You can list the settings of all the variables with the UNIX *printenv* command. In *spy*, type:

```
!printenv Enter
```

Environment variables are used for many purposes, some of which are:

- Modifying the behaviour of programs
- Specifying locations of resources (e.g. a path specifies where executables are)
- Stringing long names into an easier to use variable

For a listing of environment variables used by Houdini, see *Environment Variables* p. 211 in the *Scripting* section of the Reference manual.

SETTING YOUR OWN ENVIRONMENT VARIABLES

You can set your own environment variables with the UNIX *setenv* command, or with the *spy* S command. For example:

in UNIX: `setenv ZOO /usr/staff/john/pic1/jobs/Zoo`
in *spy*: `S ZOO=/usr/staff/john/pic1/jobs/Zoo`

sets the environment variable *$ZOO* to contain the pathname to the file *Zoo* in the subdirectory: */jobs*. For an explanation of jumping, see *Jumping with Environment Variables* p. 298.

You can put your own variables in the *.login* file of your home directory. These can be job specific. If you change the *.login* file, it is not effective until the next time you login, or until you source the *.login* file. You should always source your *.login* file from a UNIX shell (i.e. quit *spy* first). Source the file in UNIX by typing:

```
source .login
```

THE SEARCH PATH VARIABLE

When jumping to a directory or file, the *path* variable is used to help search for the target. For example, say you are currently in the directory */usr/staff/john*, and the *elm* command is located in the directory */usr/bin*. You want to run the *elm* command, so you type *elm* into a C shell window, and it responds with a message that the *elm* command can't be found.

Including the directory in which *elm* can be found in the *path* variable enables it to be found instead of returning an error message. For example, adding the line:

```
set path=( $HFS $HD /usr/bin $path )
```

to your .login file, adds the $HFS, $HD, and /usr/bin directories to the directories that will be searched when looking for a UNIX command. Therefore, the *elm* command would be found, since it is in the */usr/bin* directory.

Note: Although *spy* recognizes the standard csh/tcsh CDPATH variable, on some NT systems, the CD_PATH variable should be used instead to fix problems with incorrect behaviour in shells.

7 SPECIAL KEYBOARD KEYS

There are some special csh characters you should be aware of. The defaults are listed below. They can be viewed or changed with the *stty* command and can have different meanings depending on the program currently executing and receiving the keyboard input.

See *stty* or *termio* for general keyboard and terminal information or the manual page for the program you are running such as *csh*, *vi* or *spy*.

Enter	Carriage return and new line
Bksp	Delete the last character
Ctrl W	Delete the current word
Ctrl U	Delete the current line
Ctrl R	Redraws the current line
Ctrl V	Literal interpretation. Takes next character typed literally. Often used to enter the ASCII control characters.
Ctrl S	Pause display to screen
Ctrl Q	Resume display to screen
Ctrl C	Terminate a command (sometimes also Del)
Ctrl \	Absolute terminate. Often creates a core file
Ctrl Z	Suspend UNIX and return to csh. You can resume the UNIX process by typing F G for foreground processing, or B G for background processing.
Ctrl D	Terminate a login.
~Ctrl Shift Z	Temporary halt to remote login. You can return to the remote login by typing F G Enter in the local *csh*.

8 C SHELL (CSH) SCRIPTING

The C shell is the standard UNIX command interpreter. You may want to become familiar with it, as many scripts are written in *csh* (C shell) notation, and it is possible that you will want to write your own at some point.

Some key points you should know about are listed below. You should obtain a book on UNIX for more information.

8.1 ABOUT SCRIPTS

A *csh* script is a list of C commands. You can type the same lines into a shell, but a script is more flexible because it can be edited and run repeatedly by typing:

```
source scriptName
```

A script intended for execution by the *csh* will most always contain as its first line:

```
%!/bin/csh –f
```

and have the file permissions set to executable. If this is the case, you can simply type the name of the command without using *source*.

Below are some *csh* features that you'll probably want to take advantage of. Please refer to the *csh* UNIX manual for more detail.

8.2 ALIASES

Common commands can be *aliased* to a shorter word as in:

```
alias ttyps ps –ef | grep ttyd1
```

this enters the command: *ps -ef | grep ttyd1* every time you enter *ttyps*.

To remove an alias, type

```
alias –u aliasname
```

8.3 VARIABLES

Variables can be set with:

```
set a = anotherword
```

Once set, they can be used in a command line by preceding it with a *$*, as in:

```
echo $a
```

Arguments to scripts can be accessed via the variables:

```
$argv[1], $argv[2], $argv[3],...
```

8.4 CONTROL CONSTRUCTS

Control constructs are used mainly in scripts and provide a mechanism to execute different commands based on some condition. Keywords:

```
foreach, while, if, switch
```

8.5 JOB CONTROL

Job control allows the suspension of the currently running job. It allows jobs to be run in the background, as well as other process *job* controls. Keywords:

```
^Z,%2, fg, bg, stop, kill
```

8.6 INPUT/OUTPUT REDIRECTION

The input of most commands comes from the keyboard and the output goes to the screen by default. *csh* provides a mechanism for redirecting I/O (input/output) to and from files or other programs. For example:

`ls list`	Put a listing of files into the file called *list*.
`cat < msg`	Write the file called *msg* to the screen.
`who | wc`	Pipe output of the *who* command to the *wc* command.

8.7 STANDARD CSH COMMANDS

HOW TO LEARN ABOUT THE COMMANDS

You can easily find the usage of any command by using the *man* command. For example:

```
man pwd
```

provides information on the *pwd* command.

finding help related to a topic

If you want help on a topic, use *man -k topics* to list all the commands related to that topic. For example:

```
man -k graphics
```

lists all the commands that have the string "graphics" in their description.

If you are completely new to UNIX, you may want to start with *man man*, which supplies help on *man* itself.

If you run across a command and can't find it in the man(ual) page it is most likely a built-in *csh* command and you should refer to the *csh* man page.

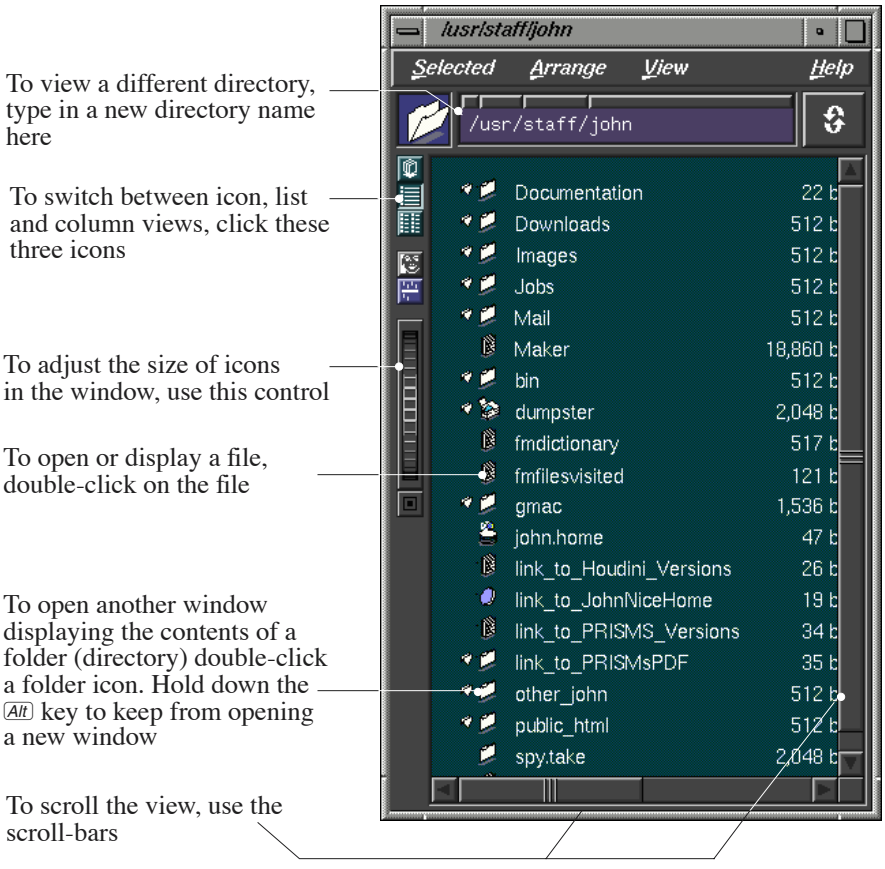
4 IRIX File Management

This is a cursory overview of SGI's IRIX operating system for manipulating UNIX files and directories. For complete documentation, see the *SGI_EndUser* online books available from the *Desktop > Help > Online Books* menu.

1 DISPLAYING DIRECTORIES / HOME DIRECTORY

Your system's administrator should have installed IRIX in such a way that you will see an icon on your desktop whose name is the same as your login name. In the example below, it is called "john".

To see a window displaying the files and folders (subdirectories) in your home directory, double-click this icon. You should see a window somewhat like the one below.



2 SELECTING & RENAMING FILES

2.1 SELECTING FILES

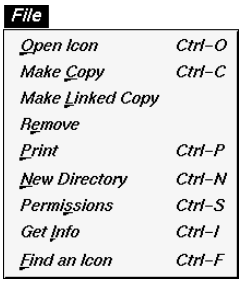
To select a file or folder, click on its icon – it turns yellow to indicate that it is selected. To select multiple files and folders, hold down the Shift key and click on them, or box-select (drag a rectangle around) several icons.

2.2 RENAMING FILES

To rename a file or folder, click on it to select it, then begin typing the new name. When you are done, type Enter. If you wish to go back to the original name, use Bksp to erase all characters in the name being displayed, and then type Enter. The original name will be re-entered, because you can't have a filename with no characters.

3 NEW DIRECTORIES, DELETING FILES, PERMISSIONS, ETC.

Select a file with , then click with . A menu appears like the one below:



The items in the menu allow you to perform various action to your files. These are detailed below.

3.1 MAKE LINKED COPY

This creates a symbolic link. Basically, after you've created the linked copy, you can drag the linked copy to another directory, and double-clicking it will open it just as if you had double-clicked the original. The purpose of this? It allows you to have things in more than one place at once.

This is equivalent to the UNIX *ln* command. See *What is a symbolic link?* p. 302 for details.

3.2 REMOVE

Select *Remove* from the menu if you want to Delete the file. You need to *Empty Dumpster* from the Desktop menu if you do this. If you want to retrieve the file after you have removed it, you can still get to it by opening the dumpster icon in your home directory – so long as you haven't performed the *Empty Dumpster* command yet. If you've emptied the dumpster, it's gone for good.

3.3 NEW DIRECTORY

Creates a new UNIX directory in the window of the directory that is currently displayed name "empty.dir". If you begin typing a new name immediately after creating it and before you've clicked anywhere else, you can rename it. Otherwise you will need to select the icon again before being able to rename it.

3.4 PERMISSIONS

You can view and set the permissions of any file or folder by selecting it, and selecting permissions. A window appears displaying the current permissions. Click on the check-boxes, then click *Apply* to set new permissions.

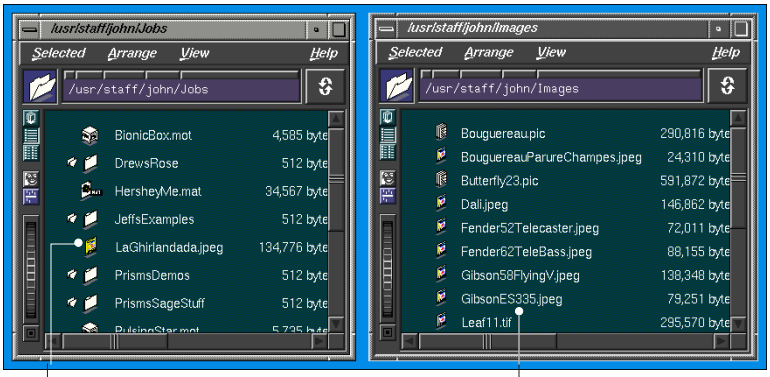
You can also enter in a new owner or group name if you are the owner of the file / directory, or if you have the *root* password.

3.5 GET INFO

Select a file and select *Get Info* from the pop-up menu displays various information about a file. It also allows you to display the size of that all the files a subdirectory contains.

4 COPYING FILES

Click on an icon, and drag it to the destination window. The file is copied or moved to the directory displayed in the destination window.

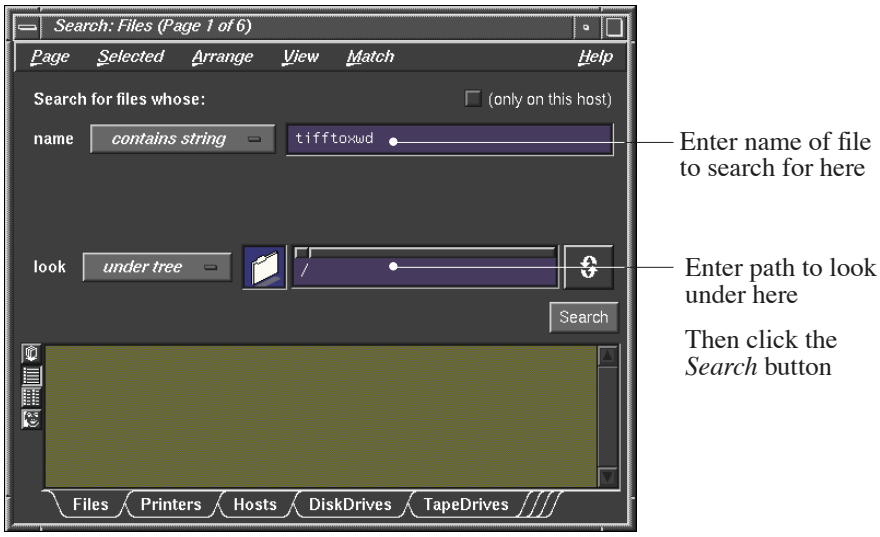


Drag this icon... ...to this window

If the directory is that of a different drive, a copy is performed; if it is on the same drive, a move is performed.

5 FINDING FILES

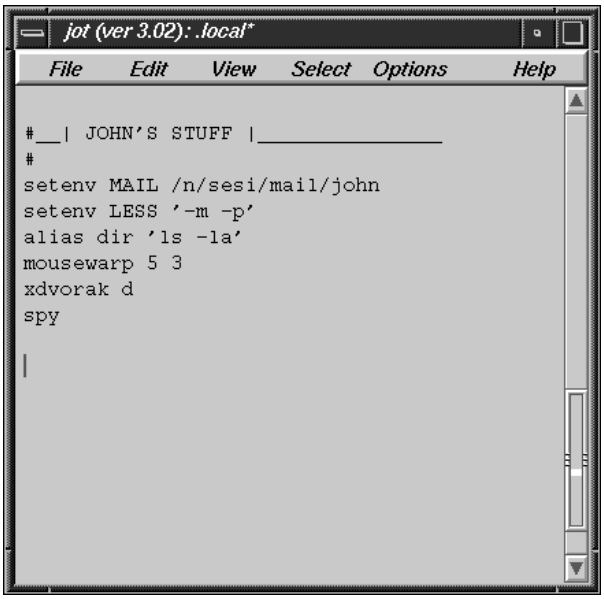
To find files, use the *Find > Search For > Files* command available from the IRIX desktop menu. This displays a window in which you can enter the name of the file to search for, and the path under which the search should take place.



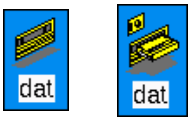
- Use a pathname of: / to search all machines on the network.
- Use a pathname of: */usr* to search only your own directories.

6 DISPLAYING AND EDITING TEXT FILES

If the file is a text file, simply double-click it's icon, and you will see the file displayed in a window. To make changes, click where you want the cursor, and start typing. To save and quit, use the File menu.



7 TAPE BACKUPS



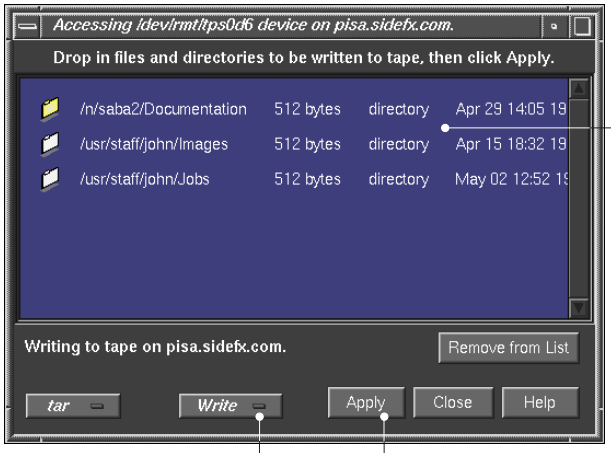
FINDING & SETTING UP A DRIVE ICON

Before you can read and write files to tape, you need to find a tape drive to read and write with. Use the *Find > Search For > Tape Drives* command from the IRIX desktop menu. To search for a type that *is 4mm DAT*. When an icon appears (there may be several), drag it onto your desktop, and close the search window. It is now ready for use.

The icon will appear empty or full depending on if a tape is actually in the drive.

TAR THE EASY WAY

To tar files to and from a tape, double click ( ) the DAT icon – a Tape window appears. Drag the icons of the files and folders you want to write into the Tape window, select *Write* from the pop-up menu, and click *Apply*. The operation will proceed from there.



To read files, select *Read* from the pop-up menu, and enter the directory for where you want the files to be read *to*. For example: */usr/staff/john*. You must have write permissions for the directory that you wish to read into.