vsh — Command

Interactive graphical shell
vsh [-d*directory***] [-eirt]**

vsh is a visual shell. With it, you can use arrow keys or simple keystrokes to perform common operating-system tasks, such as change directories, edit files, and execute programs. Each user can program a bank of up to nine function keys to perform complex tasks with a single keystroke. With **vsh**, a naive user can access much of the power of his operating system without having to learn the details of a line-oriented shell.

Unlike X or other windowing systems, **vsh** works on a character-based terminal and requires only a modest amount of memory. It does not require a mouse.

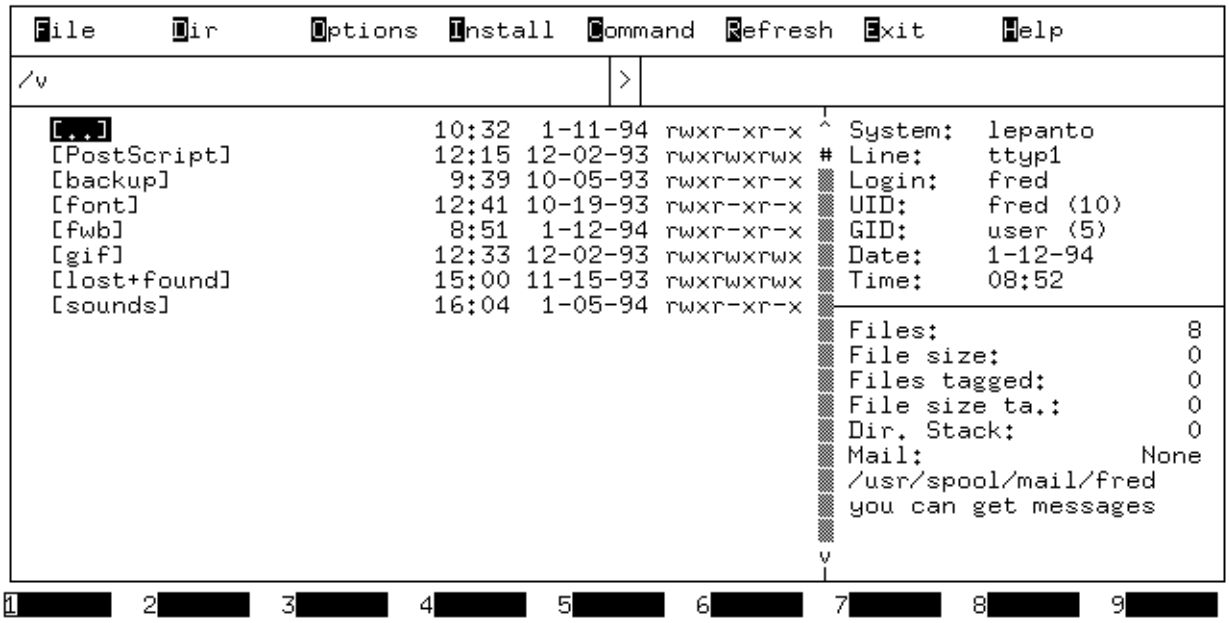
Graphics Interface

vsh uses the **curses** library and **terminfo** descriptions. To use **vsh**, you must have a **terminfo** description installed for the device upon which you wish to run it. Under UUCP or COHERENT, you must set the environmental variable **TERM** to point correctly to that description. See your system's documentation for details on how you should set up the terminal description and environment.

If you have a non-standard terminal or have trouble displaying **vsh**, try invoking it with the options **-e** or **-t**. All of **vsh**'s command-line options are described below.

Main Screen

When you invoke **vsh**, you see a screen that appears as follows:



As you can see, the screen is divided the following six sections, or *windows*:

- The first window, the *Command* Window, is the narrow window that runs across the top of the screen. This window lists the commands that **vsh** can perform. You will enter this window frequently as you work with **vsh**.
- The second window, the *Current Directory* window, names the directory that you are currently in.
- The third window, the *Destination Directory* window, names the default destination directory.
- The fourth window, the *File* Window, extends down the left side of the screen. It lists the contents of the current directory. You will also work frequently in this window.

- The fifth window, the *System* Window, is the upper window on the right side of the screen. It gives information about the system, that is, who is running **vsh**, the device she is running it on, and the current date and time. Your cursor never enters this window.
- The last window, the *Status* Window, gives information about the work you have performed under **vsh**. Again, your cursor will never enter this window.

Across the bottom of the screen are nine "stubs," one each for function keys one through nine. The stub's text indicates the command that **vsh** executes when you press that key.

The following sections discuss each window in detail.

File Window

The file window lists all of the files and directories within the current directory. This is the default window for **vsh**; the cursor ordinarily rests in this window, and you will do most of your work in it.

The leftmost column in the File Window gives the name of each file and directory. Directories are given at the top of the list; they are enclosed within brackets '[ ]'. The other columns give, respectively the time the file or directory was last updated; the date it was last updated; and its permissions. For information on how to interpret the permissions string, see the documentation that came with your operating system.

The top listing in the File Window is always **[..]**, which represents the current directory's parent directory.

The top listing in this list is highlighted by being shown in reverse video. To move the highlighting bar up and down the list, use the arrow keys. If you press the arrow keys on your keyboard's number pad, be sure to turn the **<NumLock>** key *off*, or the keys will not work as you expect. If you press the (°) key, the bar shifts down one row on the list. Pressing the (ª) key moves the bar up one row.

You can page up or page down by pressing, respectively, the keys **<PgUp>** and **<PgDn>**. The key **<Home>** moves the cursor to the top of the list, and **<End>** moves it to the bottom. If your terminal does not implement these keys, you can use the following control characters:

<ctrl-N>	Next page (like **<PgDn>**)
<ctrl-P>	Previous page (like **<PgUp>**)
<ctrl-A>	Beginning (top) of list (like **<Home>**)
<ctrl-E>	End (bottom) of list (like **<End>**)

Note that if the list of files and directories is too large to fit into the window, moving the bar to the bottom of the window and pressing (°) will scroll the list. If you press the **<End>** key, the row moves to the last row in the list; and if you press **<Home>**, it moves to the top of the list.

A scroll bar runs down the right side of the File Window. As you scroll up and down this window, the scroll bar moves. Note that the position of the scroll bar is proportional to the highlighting bar's position in relation to the entire list of files, not just to its current position within the File Window. This gives you an easy way to see just where you are in the entire file list.

If you position the highlighting bar over the name of a directory and press (¢), **vsh** names that directory in the Current Directory Window and displays its contents in the File Window. For example, if you position the highlighting bar over the entry for directory **[letters]** and press (¢), **vsh** displays the contents of directory **letters** in the File Window. (If you are familiar with the Bourne or Korn shells, this has the same effect as typing the command **cd letters**.) To return to the directory you had just been displaying (that is, the parent directory of **letters**), use the arrow keys to move the highlighting bar to the entry **[..]**; then press (¢). **vsh** changes the contents of the Current Directory Window, and in the File Window erases the contents of **letters** and displays the contents of its parent directory.

If you press (¢) while a file is highlighted instead of a directory, **vsh** does the following:

1. If the file is executable, **vsh** executes it.
2. If the file matches a pattern from the file-action list, **vsh** executes the action from the list with the file as input. The file-action list is in file **.vsh** in your home directory; and it looks something like the following:

```
[Mm]akefile:make
*.mk:make -f %F
*.sh:sh %F
*.c:cc -c -O %F
*.sc:sc %F
*.a:ar tv %F | more
*.[1-9]:nroff -man %F | more
*.tar.F:fcat %F | tar xvf -
*.F:fcat %F | more
*.tar.Z:zcat %F | tar xvf -
*.Z:zcat %F | more
```

vsh recognizes most common wildcard characters: that is, it interprets an '*' as any string of characters, and it interprets an '?' as any single character. The token **%F** stands for the file that is currently highlighted. For example, in the above example the entry ***.Z:zcat %F** means that if you select a file with the suffix **.Z** (which usually means that a file has been compressed), **vsh** should pass that file to the command **zcat** to uncompress and display it. **vsh** defines many defaults for you when it creates this file, which you can use as a model. To change the file-action list, use the **File actions** sub-command of the **Install** command, which is described below.

3. If the file appears to be ASCII **vsh** displays it with the default viewer.

While **vsh** is working, it displays a large letter 'X' in reverse video in the lower right corner of the screen. This shows that **vsh** is doing some internal task. **vsh** cannot accept any commands while the 'X' is displayed, so please be patient.

Also, note that **vsh** cannot handle more than 1,000 files in any given directory. If a directory contains more than 1,000 files, only the first thousand will be available for use.

System Window

The system window is the upper of the two windows on the right side of the screen. The cursor never enters this window; rather, this window simply displays information about your system, and how you are currently using it. It contains the following entries:

System: This gives the name of your system, as you (or your system administrator) has set it.

Line: This gives the device by which you are accessing your system. If you are working on your system's console, then you should see **console** on this line; whereas if you are accessing your system via a PC plugged into a serial port, then you should see here the name of that port. If you are using virtual consoles, the line is shown as **mono[0-8]** or **color[0-8]**.

Login: This gives the name under which you logged into your system. For example, if your login identifier is **fred**, then you should see **fred** on this line.

UID: This shows your user-identification number (or UID). This is the unique number by which your system knows you, as set in file **/etc/passwd**.

GID: This gives the number and name of the user group to which you belong. Users on a UNIX or COHERENT system can be organized into groups; permissions on files can be set to include the members of your group, but exclude all others.

Date: This gives today's date (or rather, what your system thinks today's date is).

Time: This gives what your system thinks the current time is. If your system's time is not set correctly, then the time shown here will not be correct.

Command Window

The Command Window is the top window, and stretches across the width of the screen. This window gives you access to **vsh**'s commands. Some commands in the command window actually open an entire menu of commands, with which you can perform all manner of work.

The command window contains the following entries. For convenience, the following displays the entries vertically; the actual window displays them horizontally:

File
Directory
Options
Install
Command
Refresh
Exit
Help

When the cursor is in the File Window (which is the default) and you wish to execute one of the commands in the Command Window, press its initial letter. For example, to execute the **Refresh** command, press **R**.

Note that the commands on this window are in two groups. A command's behavior differs, depending upon which group it belongs to.

The commands **File**, **Directory**, **Options**, and **Install** display a drop-down menu when you invoke it. That is because they have more than one option available under it. If you do not wish to invoke any of the sub-commands on that menu, you can do either of the following: You can press the **<Esc>** key, which erases the drop down-menu and returns you to the File Window; or you can press the (æ) or (Æ) keys, which move you to the command in this group that lies, respectively, to the left or to the right of the current command. For example, suppose that you were in the File Window, and you pressed **F** to invoke the **File** command. **vsh** would move the cursor into the Command Window, and display the File Command's drop-down window, which displays its sub-commands. If you then pressed the **<Esc>** key, **vsh** would return you to the File Window. If you pressed the (Æ) key, **vsh** would erase the **File** command's drop-down window and display, instead, the drop-down window for the **Directory** command. If, however, you pressed the (æ) key, **vsh** would erase the **File** command's drop-down window and display, instead, the drop-down window for the **Help** command. As you can see, **vsh** "wraps-around" the cursor — it considers the command at the far right to be to the left of the command to the far left, and vice versa. This concept is a little difficult to grasp when you read about it, but once you try it, it will quickly become clear.

The other group of commands are the commands **Command**, **Refresh**, **Exit**, and **Help**. Each has only option, so when you invoke one of them, it immediately begins to execute that option. When you access one of these commands through the (Æ) and (æ) keys, each displays a drop-down menu that shows its one option.

The following describes each command in detail.

File Pressing **F** invokes the **File** command. This displays a drop-down menu that lists a set of sub-commands. These sub-commands let you manipulate files; with them, you can edit a file, create a file, change its permissions, rename it, erase it, print it, or do other common tasks.

To invoke a sub-command, you can do either of the following: Press the letter in the sub-command that is underlined (each sub-command has its own unique letter with which you can invoke it); or use the (ª) and (°) keys to move the highlighting bar to that command, and then press (¢).

The following discusses each sub-command in detail:

Copy This sub-command copies a file. Please note that the behavior of this subcommand depend upon whether you have tagged files.

If you have tagged one or more files, **vsh** opens a pop-up window that requests the path name of a directory. By default, **vsh** displays the destination directory, if you have set one. When you enter the path name, **vsh** copies every tagged file into that directory.

If you have not tagged any files, **vsh** opens a pop-up window that requests that you enter a file name or a path name. Again, if you have set a destination directory, the window displays it by default. If you enter only a file name into this window, **vsh** copies the highlighted file into the newly named file in the current directory; if you have named an existing file, **vsh** prompts you before it overwrites that file. If you enter a path name, **vsh** copies the highlighted file into the directory you have named; the copied file retains its current name. If, however, you enter both a file name and a path name, then **vsh** copies the highlighted file into the directory you named, and gives it the file name that you entered.

Note that this command will not overwrite a file that you do not own; nor will it create a new file in a directory in which you do not have write permission, or copy a file on which you do not have read permission.

Move This sub-command prompts you for the name of a directory; if you have set a destination directory, **vsh** displays it by default. When you confirm the destination, **vsh** then moves all tagged files into it. (If no files are tagged, **vsh** moves only the highlighted file. For more information on tagging, see the entry for the sub-command **Tag**, below.) The files retain their names in the new directory.

This command does not move a file for which you do not have read permission, or move a file into a directory into which you do not have write permission; nor will it move a file into a non-existent directory (of course).

Delete This sub-command deletes the tagged files. (If no files are tagged, then it deletes only the highlighted file. For more information on tagging, see the entry for the sub-command **Tag**, below.) It will prompt you to confirm that you really do want to delete the file or files in question. With regard to the mass deletion of tagged files, this sub-command lets you choose whether to do a mass deletion or delete files one at a time.

Note that this sub-command will not delete a file that you do not own.

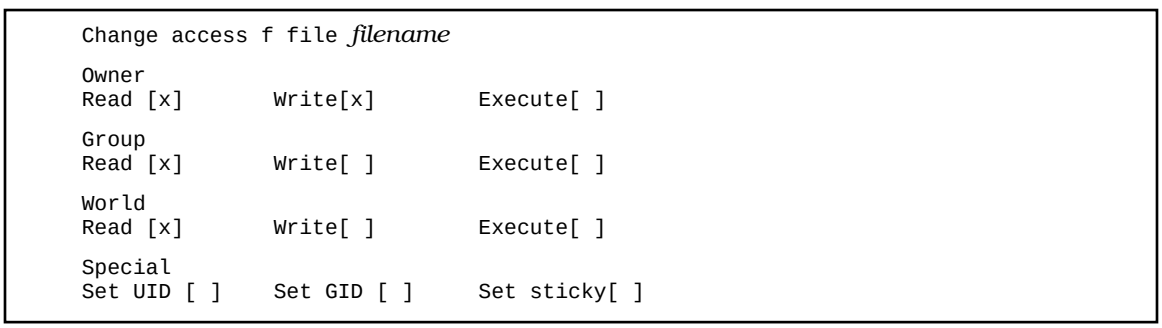
Rename
This sub-command lets you rename the highlighted file. It opens a pop-up window that shows the current name of the file, and prompts you to type the new name. Press **<Esc>** to abort this sub-command, or type the new name and press (◊).

It does not work with directories. It will not let you rename a file that you do not own.

Execute
This sub-command executes the highlighted file. **vsh** prompts you to type the arguments you wish to pass this file, then invokes the file with those arguments.

Note that **vsh** will not execute a file for which you do not have execute permission.

Access This sub-command lets you change the manner in which every tagged file can be accessed. (If no files are tagged, the default is the highlighted file.) When you invoke it, **vsh** displays the following pop-up window for each tagged file:



An 'x' in a field means that that permission is turned on; a blank means that it is turned off. Use the arrow keys to move to the cursor the field whose status you wish to change, then enter a space or 'x' to, respectively, turn off or turn on that given permission. To abort this command, press **<Esc>**. Note that you can reset permissions only on the files you own.

Owner This lets you change the owner and group that owns each tagged file. If no files are tagged, then this applies only to the highlighted file. When you invoke this sub-command, **vsh** opens a pop-up window that shows the user and group that own a file: type the name of the user or group you want to own the file. **vsh** repeats this step for each tagged file. To abort this command, press **<Esc>**.

Print This passes every tagged file to the print spooler for printing. To change the default print spooler, use the **Install** command's **Print spooler** sub-command.

Note that **vsh** simply passes the file to the spooler for printing; you cannot use this to process a file before printing it. Under some operating systems, if you try to use this feature of **vsh** to print a file on a PostScript printer, the printer will hang. We suggest that you use the **Command** feature to print a file on a specialized printer; it's a little more difficult, but it works. Another approach under UNIX or COHERENT is to use the spooler **lp** and prepare a special backend script to

do the processing automatically.

View This sub-command invokes the default viewer to display the contents of every tagged file. If you try to view the contents of a binary file, the results may not be what you expect.

Note that **vsh** will not display a file for which you do not have read permission. To change the default viewer, use the **Install** command's **File viewer** sub-command.

Edit This sub-command invokes the text editor to edit every tagged file. If no files are tagged, then edit only the highlighted file.

The default text editor is **vi**, which can create problems for persons who do not know how to exit from that editor. To change the default text editor, use the **Install** command's **Editor** sub-command. Note that **vsh** will not let you edit a file for which you do not have read permission.

Edit new
This sub-command prompts you to type the name of a file, then invokes the editor for that file. This can be a new file (that is, one that does not yet exist in the current directory), or a file that already exists.

Note that if you do try to edit a binary file, you may find yourself running into difficulties.

Touch This "touches" every tagged file — that is, it changes the date and time that the file was last modified, just the same as if you had just edited it.

Note that you cannot touch a file for which you do not have write permission.

Tag all This sub-command "tags" every file in the current directory. This lets you do mass moves or deletions of files. When you tag a file, **vsh** updates the entries **Files tagged** and **File size ta.** in the Status Window, to reflect the number and total size of the files you have just tagged. It also prints an asterisk next to the tagged file.

When the cursor is in the File Window, you can toggle tagging on the highlighted file by pressing the space bar. Note that the highlighted file is implicitly tagged, whether an as asterisk appears next to it or not. For details, see the section on the Status Window, below.

Untag all
This sub-command untags all files that are tagged in the current directory. As noted above, you can toggle the tagging of the highlighted file by pressing the space bar. This command updates the Status Window to reflect your changes.

Select This sub-command opens a pop-up window and lets you enter a regular expression; it then tags all files that match the expression. For example, if you enter ***.c**, then this sub-command tags all files that end in the string **.c**.

File type
This sub-command prints a summary of information about the type of the highlighted file.

File info
This sub-command opens a pop-up window that displays the following information about the highlighted file or directory:

```
Filename
Filetype
I-Node
Links
Owner UID
Owner GID
access
modification
status changed
```

Filename is the name of the file. *Filetype* is its type, e.g., directory or regular file. *I-Node* gives the number of this file's i-node. *Links* gives the number of links to the file. *Owner UID* and *Owner GID* identify the owner and group that own this file. *access*, *modification*, and *status changed* give, respectively, the date and time the file was last accessed, last modified, or last had its status changed.

Directory

Pressing **D** invokes the **Directory** command. This displays a drop-down menu that lists a set of sub-commands. These sub-commands let you manipulate directories; with them, you can create a directory, remove a directory, change permissions, and other common tasks. You can also manipulate a "directory stack," which lets you jump quickly from one directory to another without having to retype its name.

The following discusses each sub-command in detail:

Change

This lets you change the current directory. When you invoke this subcommand, **vsh** displays the following pop-up window:

```
Enter destination path
```

Type the full path name of the directory you wish to enter. If this directory does not exist, or if you cannot access it, **vsh** leaves you in the current directory; otherwise, it moves you to the requested directory.

Home This moves you to your home directory.

User's Home

This moves you to the home directory of another user. When you invoke this sub-command, **vsh** asks you to name the user whose home directory you wish to enter. To abort, press **<Esc>**. If the user you enter does not exist or if you do not have permission to read her home directory, **vsh** leaves you in the current directory; otherwise, **vsh** moves you into that user's home directory.

Set dest

Set the destination directory. This directory is saved in your **.vsh** file, and is restored the next time you invoke **vsh**.

Push The next three sub-commands makes it easy for you to maneuver your way around the your machine's file system. The work by using what is called a "directory stack". In effect, you can tell **vsh** to remember the directory you are in (this is termed "pushing" the directory onto the stack); then, when you have switched to another directory, you can returned to this directory simply by "popping" this pushed directory from the directory stack. This lets you move around among directories without having to retype them continually.

The **Push** sub-command pushes the current directory onto the directory stack. When you push a directory, **vsh** increments the number next to the entry **Dir. Stack** in the Status Window. This tells you how many directories you have pushed onto the directory stack.

Pop & cd

This sub-command moves you to the last directory you pushed onto the directory stack. It also removes that directory from directory stack. When you pop a directory from the directory stack, **vsh** decrements the number next to the entry **Dir. Stack** in the Status Window. This tells you how many directories remain on the directory stack.

Note that directories are popped in the order opposite from that in which they were entered. For example, if you pushed directory **/usr/bin/sys** onto the directory stack, then directory **/usr/lib/mail**, then **/bin**, invoking the **Pop** sub-command will return you to directory **/bin**, then to **/usr/lib/mail**, and finally to directory **/usr/include/sys**.

Switch This command switches the current directory and the top entry in the directory stack.

Copy This copies the highlighted directory plus all of its contents into another directory whose name you type into a pop-up window. It behaves much like the command **cpdir**.

Delete This deletes the highlighted directory. It does not work with files. If the directory has files in it, **vsh** will prompt you and ask if you want the directory to vanish. If you answer 'Y', then **vsh** removes it, files and all — just as if you had executed the command **rm -rf**.

vsh will not delete a directory that you do not own.

Rename

This sub-command renames the highlighted directory. **vsh** opens a pop-up window and prompts you to type the new name of the directory. Press **<Esc>** to abort this sub-command. Note that

you can rename only directories that you own. This sub-command does not work with files.

Create This sub-command creates a new directory in the current directory. **vsh** prompts you for the name of the new directory, and then creates it. Note that you can create a directory only if you have write permission in the current directory.

Access This lets you reset the access permission on the highlighted directory. This is the directory equivalent of the **File** command's **Access** sub-command.

Owner This lets you reset the user and group that own a given directory. This is the directory equivalent of the **File** command's **Owner** sub-command. Note that only the superuser **root** can run this command.

Read new
This tells **vsh** to re-read the current directory. **vsh** copies the contents of the current directory into memory for its own use; thus, if other people manipulated the directory and its contents after **vsh** read its contents, what you see in the File Window will not reflect the true state of affairs in that directory. If you are working with a directory that is being manipulated by one or more other people, you should issue this command from time to time, to ensure that you are working with an accurate image of the directory's contents.

Switch CWD
This command switches the current working directory with the destination directory.

Switch TOS
This switches the destination directory with the directory on top of the stack.

Info This is the same as the **File info** sub-command under the **File** command, described above.

Options
Pressing **O** invokes the **Options** command. Its sub-commands let you perform common system tasks. The following discusses each sub-command in detail:

Shell This command invokes an interactive shell. When you exit from the shell (either by typing **exit** or **<ctrl-D>**), you will be returned to **vsh**.

By default, **vsh** invokes the Bourne shell **sh**; to change the default shell, use the **Shell** sub-command under the **Install** command, which is described below.

Lock terminal
This command locks your terminal. When the terminal is locked, no command can entered into it; this lets you walk away from your terminal briefly without worrying whether anyone (e.g., your cat) will do anything untoward under your login. The terminal remains locked until you retype the secret password that you entered when you invoked this sub-command

When you invoke this sub-command, a pop-up window appears with the following:

```
Lock Enter Password
```

vsh prints a '#' to echo each character that you type. If you wish to abort the **Lock** sub-command, press **<Esc>**. When you have finished entering your password, press (¢). When you have entered the password, the following window appears:

```
This Terminal is locked!

Enter Password to unlock
or hit return to logoff
```

Type the password to return to **vsh**. If you (or someone else) presses (¢), you will be logged out of your system.

Messages
This