Graphical shells

MUSCLE SOUP

The Hotwire shell removes the complexities of interactive consoles. We'll show you how to Hotwire your Linux with this handy text and graphical shell for developers and administrators.

BY JÖRG HARMUTH

Hotwire is an intelligent, hybrid text and graphical shell for developers and administrators [1]. According to developer Colin Walters, Hotwire "… is intended to replace the interactive command-line portion of a typical Unix shell."

Hotwire combines the functions of a terminal window, a shell, and many of the familiar command-line utilities. One interesting feature of Hotwire is its ability to separate input and output streams (Figure 1). But is Hotwire truly more than just a shell? I decided to investigate.

Hotwire is not included with the major distributions, so the first task is to install the program (see the "Installation" box for details).

The following section describes an installation of Hotwire on Fedora 7, which uses the Gnome desktop, Hotwire's "natural habitat," by default. That said, nothing prevents you from using Hotwire in a KDE environment.

Ready to Launch

After completing the install, launch Hotwire on the Gnome desktop by going to *Applications* | *System Tools* | *Hotwire Shell*; users with KDE will need to look in *System* |*Terminals* | *Hotwire Shell*. Alternatively, you can launch the tool by entering *hotwire &* at the command line. The only arguments the tool accepts at the command line are *--debug[-modules…]* and *--help*, and these options are only useful for debugging or if you need some help.

Input/Output

The upper part of the Hotwire window includes a menu bar; the output area is in the center, with the Hotwire command history – showing you the most recent three commands – just below.

At the bottom of the window is a command input line. Because Hotwire separates input and output, you can enter

GLOSSARY

Builtins: Commands included in the program code that do not reference external programs. Hotwire implements the *ls* command as a builtin.

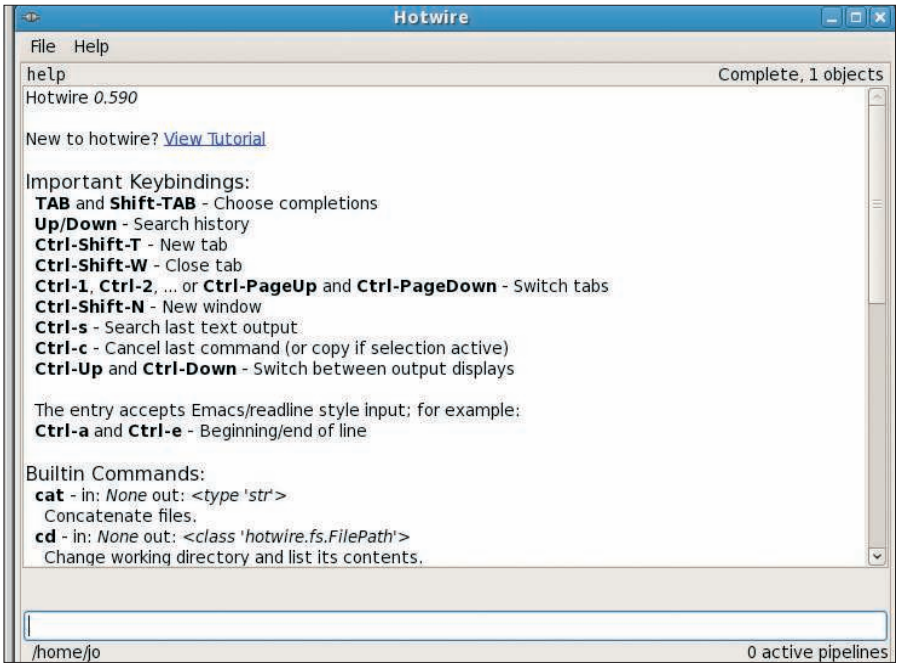

Figure 1: The interactive Hotwire shell separates input and output.

new commands while watching the file output in a pager. The status line at the bottom of the window shows the current directory and any pipes that are open.

As soon as you type the first letter of a command, a display area appears above the input line. The display area is used for auto-completion and lists all the commands in alphabetical order that match your current input (Figure 2).

Builtins

Hotwire's internal commands, or **builtins**, are listed in the tree above the external commands. For example, if you type *ls*, the program will run the built-in command rather than the system's own *ls*.

Pressing the Tab key moves from the top down through the list of system commands; you can press Shift + Tab to move in the opposite direction. Use the Up arrow and Down arrow keys to browse Hotwire's internal history. The currently selected command is automatically displayed, and you can press Enter to run it (Figure 3). Hotwire does not support point-and-click selection.

This lets users execute simple commands, such as *ls*, quickly. After changing directory (*cd*), Hotwire automatically lists the content of the new folder in the windows above.

Ls

The *ls* builtin assigns an icon for each file type in the output and displays the icons like a file manager.

If you click a file name or folder icon, Hotwire will translate this to the matching default action and add the action to the history. This allows you to change directory in the shell or edit text files by clicking with the mouse; however, you can only change to a directory up to two levels down the directory tree.

Below this, Hotwire displays underlying subdirectories, but you can't click to open them.

Hotwire does not support the ability to launch programs by clicking, but this feature is on the developer's roadmap.

History

The Hotwire shell history is quite different from the history function of shells such as Bash. Hotwire stores commands from other shells in a file called *~/.hotwire/persist/history.p*, which means that the program can't access commands you enter in a normal shell and vice versa. The same thing applies to aliases – if you define an alias as *alias c = clear*, the alias will only be available in the shell in which you created it.

Show, Remove, Hide

Hotwire displays the last three commands from the current session between the menu bar and the output area. If you click on one of the commands, a small menu pops up with the *Show* and *Remove* entries.

Show opens another output window and displays the content for the action in it, and *Remove* deletes the command with its output from the history. To hide it, click on the command above the visible output area and select *Hide*. Ctrl + PgUp/PgDn toggles the focus between output areas.

The keyboard shortcut Ctrl + S searches the current output area for text patterns, and the search is effected in real time when you start typing. Hotwire shows matches in bold type; however, the program will only search in text output and will not search rendered output, such as *ls* results. Pressing the Esc key stops searching.

Figure 2: At the top of the window, Hotwire shows a classic auto-completion area with matches from history below.

Installation

The current version of Hotwire is 0.590. As the low version number suggests, integration with various distributions is at various stages of completion. Users with Fedora Core 7 can install Hotwire by entering *yum install hotwire*. If you use Mandriva Linux Spring 2007, you will need to enable the */contrib/backports* branch. Ubuntu users can download Hotwire online [2]. The shell is still on the Debian wish list, which means building the tool from source code.

SUSE users can download RPM packages from the Packman [3] site. In our lab, Hotwire failed to run on SUSE Linux 10.0 and claimed that *cairo* was missing, although it wasn't. In all cases; you will need the latest version of Python; however, your distribution's package manager should handle these dependencies.

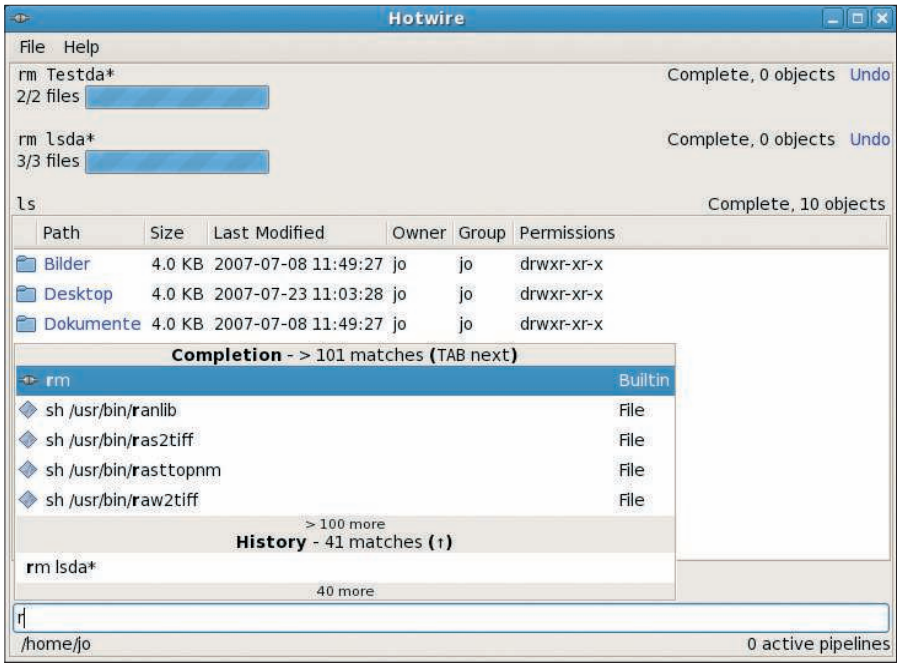

Figure 3: Hotwire lets you use autocompletion to search the command history.

An *Undo* button to the right of the history lets users undo the last three actions. The Undo feature even works for the built-in *rm* command, which does not need any parameters to delete directory trees (compared with the *-r* of legacy Unix shells).

Filter

The *filter* builtin, which will search rendered output for regular expressions, is the Hotwire equivalent to *grep*. Filter-based searches are case sensitive. The command is piped; for example, *ls | filter Desk* in your home directory will only show the desktop folders.

Additionally, filter supports a number of parameters. *ls | filter conf cmd* lists all the executables (*cmd*) in the current directory that contain the *conf* string. The documented list of available parameters is fairly short, though, and just includes *cmd* and *owner_name* for filtering by owner. As a useful side effect, filter never lists itself, thus removing the need for *| grep -v grep*.

Pitfalls

According to the documentation, Hotwire supports pipes that let users string commands together like in a normal Unix shell. In our lab, however, the program converted *ls | less* to *ls | sh /usr/bin/less* and thus output the current directory but failed to page it. The Less pager acted as if it had nothing to do with the output. Because the program has a scrollbar at the side, this failure was excusable.

However, this is not the case for *less < file >*. The output is displayed as in *cat < file >* and thus does not launch the default editor when you press *V*. Hotwire interprets this key press as the start of a new command.

The Hotwire builtins do not support redirection of input and output via < and >; unfortunately, the website gives you no clue as to whether the developers are planning to implement this function in the future.

As a workaround, you can use external programs for redirection; the *sh /bin/ls -l /bin > ~/ls file* works as intended and stores the file and its content in your home directory.

Sh

If you look at autocompletion, you will note that the *sh* prefix keeps cropping up. The reason is that Hotwire uses the builtin *sh* command to launch external shell commands, which is why the shell automatically converts a simple *ifconfig* to *sh /sbin/ifconfig*.

The important thing is that *sh* is only suitable for text output, and this is noticeable with an interactive program such as the text-based Links browser. If you try to launch Links by typing *sh links*, you will not see an error message. Because Links not only generates text, but also expects input, *sh* can't handle this and simply complains that there is *No Output*.

Term

Another builtin called *term* handles this situation by opening a new tab with the program you call. The correct command here is *term links*.

Hotwire's documentation even advises you to try *term* if a command does not work as intended. The *term less < file >* command opens the file in a new tab with the Less pager. Now *V* works as intended.

When you quit *less*, Hotwire automatically closes the tab; however, I was unable to pipe *less* in our lab. These experiments revealed that you can't use *term* to the right of a pipe. The input *ls | term less* – and variants of it – generated syntax errors in all cases.

Wish List

Hotwire feels strange at first for users familiar with normal shells because it uses a completely different approach from that of other consoles. Both the rendered display output that lets you click the results of, for example, the *ls* command and the Undo and history functions mean that Hotwire really can add convenience to working with the shell after a short learning curve.

Of course, Hotwire still lacks several major features before it can compete with traditional shells.

Comprehensive documentation [4] tops the wish list, followed by integration of the original shell environment (aliases and profiles) and the complete implementation of pipes and redirection.

Hotwire should also remove the need to differentiate between *sh* and *term*.

Also, a scrollable history with more than just three entries would be extremely useful. ■

INFO

[1] Hotwire shell: *http://code.google.com/p/hotwire-shell/*

[2] Hotwire for Ubuntu: *http://www.getdeb.net/app.php?name=Hotwire*

[3] Hotwire for SUSE.: *http://packman.links2linux.com/package/hotwire*

[4] Online documentation: *http://code.google.com/p/hotwire-shell/wiki/GettingStarted*