Toolman Features Steve Kinzler

Highlighted in this article: `webrowse`, a tool for viewing the world (or text) through the eyes of a browser.

by Daniel E. Singer

Dan has been doing a mix of programming and system administration since 1983. He is currently a systems administrator in the Duke University Department of Computer Science in Durham, North Carolina, USA.

<des@cs.duke.edu>

Tool: *webrowse*

Abstract: *remote load into browser and markup text*

Platforms: *most UNIX*

Language: *Perl 4 or 5, plus standard modules: Getopt*

Author: *Stephen B. Kinzler* <kinzler@cs.indiana.edu>

Availability: <http://www.cs.indiana.edu/~kinzler>

Open for Business

When I first visited Steve Kinzler's Web page, it was like I had stumbled into an old-style hardware store. Tools were hanging on the walls and lying on shelves, some of them elaborate and exotic, others simple and mundane. He had responded to one of my solicitations for tools and had invited me to visit his shop, enticing me with the prospect of custom-made tools of unique character.

And, well, Toolman can carry on with this figurative language for only so long. Suffice it to say that Steve is a system administrator after my own heart, who looks for creative solutions to simplify and expedite the common, repetitive, computer-related tasks that beset him and his users, and who authors software tools as a means toward that end.

Text 2 Browser

One of the tools that Steve highlighted in our correspondence is a program called `webrowse`, which can be used as a quick interface to a Web browser (hence the name) on a UNIX system.

`webrowse` is a handy tool for us command-line types living in a point-and-click world. With a browser running somewhere on your workstation (even iconified or in another virtual desktop), point `webrowse` at an HTML document (a file or STDIN), and it will bring it up in the browser; point it at some plain text, and it can first mark up the text, adding appropriate hypertext links on the fly. Now some of this might sound like something you could do with a few simple aliases, but various aspects of this are not so easily handled.

`webrowse` can currently interface to both the Netscape (default) and Mosaic browsers, selectable via command-line or environment, and will issue appropriate commands to activate an already running browser. (Both of these browsers have the "remote control" features that `webrowse` exploits.) With the `-m` (markup) option, it will HTML-ize the input by adding standard HTML header and body tags and by scanning the text for anything that looks like an address or a URL and adding an appropriate link. So, for instance, an email address will be marked up with a `mailto:` link. Other possible markups include links for `http:`, `ftp:`, `file:`, and `news:`. `webrowse` employs sophisticated pattern matching as a basis for its heuristic approach to these transformations.

`webrowse` is also handy as a filter (with `-o`) such that the converted text output can be directed to a file or piped on to some other process.

`webrowse` Examples

As a simple example, let's say you have an email message about virus hoaxes that contains some URLs for Web pages on this topic. You can save the message to a file named "virus-hoax", then type `webrowse -m virus-hoax`. The text of the message will be marked up with HTML tags, including the URLs, and will then pop up in your browser window, where you can follow the links easily. A more efficient method would be to map some function of your mail reader to the command `webrowse -m` or just *pipe* the message to this command, reducing steps and avoiding the need for the temporary file holding the message (and you know how Toolman despises temporary files!).

webrowse has a plethora of command line options and environment variables for fine tuning and customizing its operation, making it handy to embed in other scripts as well as use on its own.

As another example, Steve uses the following key mapping with the nn newsreader:

```
map both I (
  save-full "|webrowse -mw"
)
```

This allows him, by hitting I, to view the selected articles in a new browser window, with all the URLs, email addresses, etc., converted to links.

He also defines some macros in his .exrc file for use with the vi editor. For example:

```
map ^V^Iwb :w !webrowse -m^M
map ^V^Iww :!webrowse %^M
```

The first (wb) will bring up the text currently being edited in the browser with markup added. The second (ww) will tell the browser to load the current file. For each of these, type TAB and the two letters to invoke the macro.

webrowse has a plethora of command line options and environment variables for fine tuning and customizing its operation, making it handy to embed in other scripts as well as use on its own. The -h (help) option and the man page will shed some light on these.

Other Aisles

Steve's shop, er, Web page includes many other tools addressing various aspects of system administration and general UNIX usage. Here's a quick survey of a few that might warrant an evaluation:

- Web:

  ClipControl: this Java applet is an AudioClip controller for flexibly embedding audio files in Web pages.

- Web administration:

  ftw: file tree walker, for Web document tree checking. Checks validity of symbolic links, especially if the server's running with FollowSymLinks set.

  starthttpd: start, restart, or kill an HTTP daemon, as needed. Helps to keep a server up near 100%.

  rolllogs: rollover NCSA-style httpd log files, works with starthttpd. Flexible roll-over of Web logs at various resolutions.

- Systems administration:

  dumpdates: produce readable and organized listing of dump dates for mounted filesystems.

  rdistsumm: produce readable summary of rdist output, highlighting errors.

- General use:

  push and pop: conveniently and safely push/pop files into/out of a subdirectory.

  rename: move or copy files and directories based on a sed or perl expression.

  vigrep: edit all files containing the given regular expression, such as for multi-file software development.

  wh: list all instances of given files in a search path. When which just isn't enough . . . .

  width: determine the printing widths of input lines, find the longest line in a file, etc.

`z`: convenient, safe front-end for (un)tarring and (un)compressing, with intuitive use of subdirectories. `z <something>` usually does the right thing. Good for naive users.

About the Shop Proprietor

Steve hung his hat for quite some time at Indiana University, Bloomington, where he completed his M.S. in computer science, taught, worked on many projects, and performed Web and UNIX systems administration. He is creator and maintainer of the Picons database (<http://www.cs.indiana.edu/picons/ftp/>) and the Internet Oracle (a.k.a. the USENET Oracle) (<http://www.pcnet.com/~stenor/oracle/>) and is a longtime member of USENIX. His other accomplishments are far too numerous to list here. He currently lives in Ann Arbor, Michigan, and is working for the Health Management Research Center at the University of Michigan.

Thanks, Steve, for making your materials available.

More Browsing

I surfed the Web a bit to see if other tools similar to `webrowse` were available. I found one called `txt2html` by Seth Golub that has some interesting features. It's quite versatile in its ability to add markup and allows you to define a private "dictionary" of conversion rules. It is strictly a filter, and lacks the ability to automatically interact with a browser. (But you can do `txt2html < foo.txt | webrowse -s`). `txt2html` can be found at <http://www.cs.wustl.edu/~seth/txt2html/>.

Many other public domain tools for converting between various formats and HTML are available. A good starting point for a search is <http://www.yahoo.com/Computers_and_Internet/Software/Internet/World_Wide_Web/HTML_Converters/>. (Try looking *that* one up in your Funk and Wagnalls!)

By the way, on the topic of conversions to/from HTML, I've written a script called `index2html` that can be used in conjunction with the `check` program (June 1997) to create interconnecting HTML-ized `INDEX` files in a directory hierarchy. `index2html` is still in its adolescence; comments are welcome. It can be found at <ftp://ftp.cs.duke.edu/pub/des/scripts/>.

As a final example, here's how I've used `webrowse` and `index2html` together. Somewhere in our extended filesystem, we have a directory hierarchy of documentation rooted at `/home/lab/doc/`. Each directory has an `INDEX` file, and `index2html` is used to generate the `INDEX.html` files. The following script, called `labdoc`, easily brings up a browser displaying the top level of this mini-web of documentation.

Many other public domain tools for converting between various formats and HTML are available.

```
I#!/bin/sh
#
# @(#) labdoc: bring up a browser on the /home/lab/doc/ hierarchy
#
# this script uses the script `webrowse', which will bring up
# the document in an already running browser; if that fails, a
# new browser is started;
# `WB_BROWSER' is one of the environment variables recognized by
# `webrowse', and is used here for consistency;
#
# 1/98, D.Singer

  DOC="/home/lab/doc/INDEX.html"
  WB="/home/lab/bin/webrowse"
  DFLT_BROWSER="netscape"

  $WB $DOC 2>&- ||

  { ${WB_BROWSER:-$DFLT_BROWSER} $DOC & }

  exit
```

Closing time

What we've seen here, via examples from Steve's Web page and beyond, is the tool approach in action: an approach that is very natural to the UNIX environment. A well-designed tool can make your life easier and can often be utilized as a component of other tools, as in the `labdoc` example above. And the design process can make life more interesting (to us command-line types, anyway). Sh, Perl, Tcl, . . . , these are powerful, high level languages (I've even seen an example of Bourne shell used as a formal object-oriented language [1]), and they're relatively easy to use. Need a tool? Write one! (Or cop one out on the net.)

Are there any tool topics that you would like to see covered? Be sure to let me know if you have any suggestions for future articles.

Note

[1] Jeffrey S. Haemer, "A New Object-Oriented Programming Language: *sh*," *Proceedings, USENIX Summer 1994 Technical Conference*, pages 1-13, June 1994.

Got a tool that's useful, unique, way cool? Toolman will make you famous! Please send a description to <Toolman@usenix.org>.