DirB, Directory Bookmarks for Bash

Inspired by browser bookmarks, DirB allows you to create directory bookmarks for moving around faster on the command line.

IRA CHAYUT

Imagine browsing the Web and having to type the full Uniform Resource Identifier (URI) path each time you visit a Web page—painful. However, since 1993, when browser bookmarks were added to the Mosaic browser, they have made short work of returning to sites you go to often (see **en.wikipedia.org/wiki/Internet_bookmark**). Regardless of whether you call them "Bookmarks", "Favorites", "Hotlists" or "Internet Shortcuts", they are great time-savers.

As a developer of consumer product software, I frequently work concurrently in multiple directory trees. I often bounce between the source code directories for each of my active development products, the directories that hold vendor documentation, and my desktop (where I keep all my active but as-of-yet unfiled work). I used to open a separate xterm window for each active directory, but mousing between the various windows and keeping track of which window had what directory was tedious and error-prone.

If command-line bookmarks existed, they would transport me to often-visited directories with a few keystrokes. Of course, the Linux change directory command (cd) comes with one built-in shortcut: the one to go to your home directory. To go home, I need to enter only the `cd` command without an argument. It's even easier than clicking the heels of my ruby slippers (which is not an unrelated reference to a popular scripting language, but instead a spurious reference to *The Wizard of Oz*). But, this is where the convenience ends.

I created Directory Bookmarks (DirB, pronounced "derby") to extend the concept of bookmarks to the command line and to move between directories quickly. DirB is implemented as a set of Bash shell functions and consists of a few simple commands:

- `s` — save a directory bookmark.
- `g` — go to a bookmark or named directory.
- `p` — push bookmark/directory onto dir stack.
- `r` — remove a saved bookmark.
- `d` — display a bookmarked directory path.
- `sl` — print the list of directory bookmarks.

These commands can be used alongside the usual Bash commands: `cd`, `pushd` and `popd`.

As you will see, DirB means fewer keystrokes and greater productivity. Now, I (almost) never leave home without it.

If DirB's function names conflict with commands or aliases that you already use, change the names of the offending functions in the .bashDirB file to ones that work for you.

Installation

To install DirB, download the source file .bashDirB from **www.DirB.info/bashDirB**, and save it as ~/.bashDirB to your home directory. Then, edit your ~/.bashrc file and include the following in the file:

```
source ~/.bashDirB
```

Each new Bash session now will have the power of DirB. If you use the DirB commands within the ~/.bashrc file, place the `source` line above where the DirB commands are used. I find that placing this near the top of the file works for me.

After installing DirB, open a new xterm window and follow along with the rest of this article.

DirB comes with a small bonus. When working in multiple windows at the same time, I find it handy to have each xterm window display the current directory's name in its title bar. To accomplish this, the .bashDirB file sets up the primary Bash shell prompt, $PS1, to output an escape sequence. This string then will be output as part of the command-line prompt, and the X11 windowing software will respond to the escape sequence by updating the xterm window's title bar. If you are not using X11, or if this behavior is not desired, edit ~/.bashDirB and insert a pound sign (#) in front of the `PS1=` on line 18 of the file to comment out that feature.

Saving and Using Bookmarks

The desktop is one of my most common destinations. I saved a bookmark for my desktop by going there and then entering an `s` command:

```
% cd ~/Desktop
% s d
```

(Note that the `%` represents the shell's command-line prompt and is not typed as part of the command.) The second line above creates a new bookmark named d.

Wherever I am, I now can go to my desktop with the `g` command:

```
% cd /tmp    # go somewhere
% pwd
/tmp
% g d         # go to the desktop
% pwd
/home/Desktop
```

Going to a Specified Directory

Now it's possible to move to a directory using `cd` or `g`. Wouldn't it be simpler to have one way that worked for both bookmarks and directory paths? Of course it would. So, DirB's `g` command has been extended to be able to replace `cd` fully:

```
% pwd
/home/Desktop
% g /tmp
% pwd
/tmp
```

The `g` command behaves the same as the `cd` command if the first character of the argument is a period (.) or if the argument is not the name of a saved bookmark. The special case of the first character being a period allows you to move to a current subdirectory that has the same name as a previously saved bookmark:

```
% cd /tmp
% mkdir d
% g ./d
% pwd
/tmp/d
```

If you use the command: `g d` instead of `g ./d` above, DirB takes you to your desktop, as a bookmark for the desktop with the name of d already exists.

If the argument to `g` is the relative or absolute path of a directory and there is no bookmark by that name, you are taken to the specified path:

```
% cd /tmp
% mkdir subdir
% g subdir
% pwd
/tmp/subdir
```

As with the `cd` command, if you enter the `g` command without an argument, you go to your home directory:

```
% cd /tmp
% g
% pwd
/home
```

Traveling with Relatives

Most of the source code directories I work in are organized with the same layout. From the application's source code directory, I frequently need to refer to header files in my standard library. These headers are located two directories up and two directories down in the filesystem: ../../stdlib/inc.

DirB can save relative bookmarks or bookmarks of any specified path. It is not necessary to be in the directory to be bookmarked. A longer version of the `s` command can be used to specify a bookmark's path:

```
% g projA
% pwd
/home/projectA/source/application/main
% s stdh ../../stdlib/inc
% g stdh
% pwd
/home/projectA/source/stdlib/inc
```

Once the relative bookmark has been created with the `s` command, relative movements can be made easily from anywhere that the relative path exists:

```
% g projB
% pwd
/home/projectB/source/application/main
% g stdh
% pwd
/home/projectB/source/application/main
```

This longer version of the s command sets a full path directory bookmark without changing to the target directory first:

```
% g projA
% pwd
/home/projectA/source/application/main
% s t /tmp
% pwd
/home/projectA/source/application/main
% d t
/tmp
```

Note that the current working directory was not changed by the s command and that the bookmark was set to the argument of the s command and not the current directory. The bookmark can be used later, the same as simpler bookmarks:

```
% g t
% pwd
/tmp
```

Manipulating the Directory Stack

As the g command extends Bash's built-in cd command, DirB has the p command to extend the shell's pushd command and also replaces the most common usage of the shell's popd command.

In its most-used form, the p command changes to a new directory, while remembering the current directory on a stack. The state of the directory stack then is printed:

```
% g
% pwd
/home
% p /tmp
/tmp
~
```

The tilde (~) is Bash's shortcut for the home directory. The target just as easily can be a bookmark:

```
% p d
~/Desktop
/tmp
~
```

The directory stack listing is done with one directory per line, instead of the default listing style of pushd with all the directories printed across the line. This is a personal preference and is accomplished by discarding the output from the invoked pushd command and then running a dirs -p command afterward.

Except for bookmark targets and the target dash (-), the p command works just as Bash's pushd command. In fact, all the real work is accomplished, behind the scenes, by pushd. So the normal pushd behavior, as well as the enhanced bookmark functionality, is valid (and useful):

```
% p directory   # adds dir to top of dir stack
% p bookmark    # adds bookmark to dir stack
% p             # swaps top two stack entries
% p +n          # rotate nth entry from top to top
% p -n          # rotate nth entry from bottom to top
```

To rotate the directory stack, so that the bottom directory moves to the top of the stack as the current directory, use p -0. In addition to replacing pushd, the p command also can replace the shell's popd command in its simplest form:

```
% g
% pwd
/home
% p /tmp
/tmp
~
% p -
~
```

If the full functionality of the popd command is needed, the standard popd command (along with pushd and cd) still is available and can be used alongside the DirB commands.

To get a listing of the current directory stack, the shell's dirs command works as it did before DirB.

Listing the Saved Bookmarks

DirB's sl command prints a saved bookmark listing. It has two forms. The simplest form lists the files across the line, from left to right, in reverse time order, most recently accessed bookmark first:

```
% sl
d test prod tmp beta alpha
```

In this example, the bookmark for my desktop, d, was accessed most recently.

In the longer form, sl lists the date and time that each bookmark was last referenced:

```
% sl -l
2010-03-10 14:42 d
2010-03-01 14:19 test
2010-02-27 10:17 prod
2010-02-27 14:21 tmp
2009-10-22 17:26 beta
2009-08-05 11:37 alpha
```

In this fuller listing, you can see that the d bookmark was referenced on March 10th, and the last time that the test bookmark was referenced was nine days earlier. If the long listing does not fit on a screen, the less command will page through the listing automatically.

It is possible to pass a regular expression to sl and list only the matching bookmarks. To list the saved bookmarks that begin with the letter t:

```
% sl "t*"
test  tmp
% sl -l "t*"
2010-03-10 14:19 test
2010-02-27 14:21 tmp
```

Note that the regular expression needs to be protected by double (or single) quotes to prevent the shell from trying to expand it before it is seen by the sl command.

Whenever a bookmark is the target of a g, p or s command, its timestamp is updated to record the reference. However, timestamps are not updated when a directory is accessed using `cd`, `pushd` or by directory stack manipulations.

Removing Stale Bookmarks

Directory bookmarks are so easy to make that I create them frequently. Many of my bookmarks are short-lived. If left unchecked, the saved bookmark listing would become very long and cluttered. DirB's `r` command simplifies the removal of unwanted bookmarks:

```
% sl
test   prod   d   tmp   beta   alpha
% r alpha
% sl
test   prod   d   tmp   beta
```

The second saved bookmark listing shows that the `r alpha` removed the unwanted alpha bookmark.

DirB or the underlying Bash commands issue error messages when a problem is encountered. Accessing a deleted bookmark results in such a message:

```
% g alpha
bash: cd: alpha: No such file or directory
```

This is the error message issued when a bookmark does not exist, possibly due to a misspelling.

Using Bookmarks in Scripts

Bookmarks save keystrokes and allow for fast movement between directories. Bookmarks also can be used to make scripts more portable. By referencing bookmarks, instead of fixed paths, it is possible to re-use scripts in different environments easily. I work on both Linux and Cygwin platforms. (Cygwin is a Linux-like environment for Windows platforms. For more information, or to download Cygwin, see **www.cygwin.com**.) Because Cygwin presents a very Linux-like look and feel, the transitions are painless. However, the Linux and Cygwin directory structures are different. I use DirB to set up the same list of common bookmarks on each system. This way, I can change between directories on the command line with the same keystrokes, regardless of the platform.

In addition to Linux and Cygwin environments, DirB has been tested on BSD UNIX and Mac OS X platforms. So, the flexibility of DirB bookmark references can span across a variety of systems.

The `d` command extends the DirB facility to shell scripts. (The d is short for either "display bookmark path" or "dereference bookmark path", your choice.) It allows a script to obtain the full pathname of a bookmark's directory.

Bash's command substitution `$(command)` feature usually is used to access `d`:

```
% DTOP="$(d d)"
% echo $DTOP
/home/Desktop
```

The double quotes need to surround the shell substitution in case there are spaces in the directory path. Unfortunately, this is all too common on the Windows-based Cygwin platform, so I always use the quotes. In the above example, the shell variable `$DTOP` could be used to access the desktop. To create a new log file on the desktop, the output of a command could be redirected

GENSTOR

Linux - FreeBSD -OpenSolaris - etc.

When you can't afford to take chances with your business data or productivity, rely on a Genstor server powered by the Intel® Xeon® processor.
Proven Technology.
Proven Reliability.

POWER PERFORMANCE

4 and 6 Core Xeon® Processors

-Up to 12 cores in a 1U, 5500/ 5600 series.
-Low 350W power for dual processing.
-(CPUs with TDP <100w).
-Up to 48GB DDR3 memory.

-Up to 12 cores in a 2U, 5500/ 5600 series.
-Dual redundant high efficiency power.
-Up to 96GB DDR3 memory.
-Server Power Capping via Intel® Intelligent Power Node Manager.

- Up to 20 DP nodes, 2.56TB memory in 7U.
- Infiniband QDR/DDR support.
- 93% power efficiency.
- Up to 60x 2.5" HDDs in 7U.

Genstor Systems, Inc.

intel Xeon inside™
Powerful. Intelligent.

780 Montague Express. # 604
San Jose, CA 95131
www.genstor.com
E-mail: sales@genstor.com
Phone: 877-25 SERVER
408-383-0120

to `$DTOP/logfile`. Do not forget the double quotes, in case the dereferenced path includes spaces.

I recommend the use of Bash's substitution feature, as shown above. However, a shorter way to print out the name of the path is to use DirB's `d` command directly:

```
% d d
/home/Desktop
```

Looking behind the Curtain

DirB keeps all directory bookmarks in ~/.DirB, a "hidden" subdirectory of the user's home directory. When the file ~/.bashDirB is sourced from within ~/.bashrc, it checks to see whether the ~/.DirB directory exists. If the directory does not exist, it is created. This guarantees that the bookmark repository exists.

Each bookmark has an associated file in the ~/.DirB directory with the same name as the bookmark. The bookmark file contains a one-line command, such as:

```
$CD /home/Desktop
```

The shell variable $CD is set by the `g` and `p` commands to `cd` or `pushd`, respectively, and the variable is expanded by the shell when the bookmark is invoked. In essence, the command `g d` is transformed into `cd /home/Desktop`, and `p d` is transformed into `pushd /home/Desktop`.

The DirB commands are implemented as Bash functions that do some error checking, determine which action is to be performed, and then invoke a standard command. For example, the `g` command does a couple checks before invoking the `cd` command:

```
# "g" - Go to bookmark
function g () {
    # if no arguments, go to the home directory
    if [ -z "$1" ]
    then
        cd
    else
        # if $1 is in ~/.DirB and does not
        # begin with ".", then go to it
        if [ -f ~/.DirB/"$1"
            -a ${1:0:1} != "." ]
        then
            # update the bookmark's timestamp a
            # and then execute it
            touch ~/.DirB/"$1" ;
            CD=cd source ~/.DirB/"$1" ;
        else
            # else just "cd" to the argument,
            # usually a directory path of "-"
            cd "$1"
        fi
    fi
}
```

The function `g` checks to see whether there is an argument. If $1 is a zero-length string, the user is sent home with a `cd` invoked with no argument. Otherwise, a check is made to see if the argument is the name of a saved bookmark and the first character of the argument is not a period.

If both conditions are met, the bookmark is run as part of the current shell by sourcing the bookmark file. Before execution, the shell variable `$CD` is set to the `cd` command. `source` is used instead of calling the bookmark as a shell script so that the directory change will affect the current shell. A called script would have a unique shell session that would terminate after the `cd` or `pushd`. Thus, it would have no lasting effect on the current shell session.

If the argument is not the name of a bookmark, or if it begins with a period, the `cd` command is invoked with the argument to go to the specified directory path.

Note that the `source` command in the `g` function above starts with a variable assignment:

```
CD=cd source ~/.DirB/"$1" ;
```

Bash syntax allows a command to be preceded by one or more variable assignments.

Error Handling

Most DirB commands eventually call `cd`, `pushd` or `popd` to perform the requested action. If one of these standard commands encounters a problem, it issues an error message to the standard error (stderr) stream and exits with a failing return code.

Note that because bookmarks are the names of their associated files in the ~/.DirB repository, they cannot have slashes in their names. If a bookmark cannot be created (most likely due an invalid character in the name), `s` will print an error message to the standard error:

```
% s a/d
bash: DirB: /home/.DirB/a/b could not be created
```

An error message will result if an argument to either `g` or `p` is neither a bookmark nor a valid directory path:

```
% p missing
bash: pushd: missing: No such file or directory
```

This will occur if the bookmark name is misspelled or if the bookmark has been removed. A similar error message results from the `d` and `r` commands if their arguments are not valid names of a saved bookmark:

```
% d missing
bash: DirB: /home/.DirB/missing does not exist
% r missing
bash: DirB: /home/.DirB/missing does not exist
```

If an error is encountered, DirB commands will exit with a failing return code. This behavior allows other Bash scripts to use these functions and take appropriate action in the event of an error.

Conclusion

DirB was created as a set of Bash functions to extend the concepts of bookmarks to Linux directories. It accelerates the movement between frequently accessed directories from the command line or from shell scripts. Although it's a simple tool, I rely upon DirB daily and hope that others will find it useful too.■

Ira Chayut is a longtime UNIX/Linux developer, having first worked on version 6 UNIX in 1976. He is the author of C and UNIX reference booklets, runs www.verilog.net, and has given talks on integrated circuit verification. Currently, he is the founder of a consumer products company and is responsible for all of the embedded and DSP programming. He can be reached at ira@dirb.info.