$HOME sweet ~

Dave Harding

2004-12-03

Contents

1 `Text::Autoformat` - Text Formatting Nirvana

Have you ever tried to reply to an email where the original message was poorly formatted? Perhaps you had to delete part of a sentence in a long paragraph and now you have one line that is shorter than the rest. Rather than spend your time making manual editing changes give `Text::Autoformat` a try!

Input:	Output:
My computer isn't booting GNU/Linux, could somebody please help me...	My computer isn't booting GNU/Linux, could somebody please help me...
## This is an ugly comment in a shell ## script ## that needs to be fixed.	## This is an ugly comment in a shell ## script that needs to be fixed.

Table 1: An example of: `cat $INPUT | perl -MText::Autoformat -e 'autoformat'`

1.1 Keybindings

`Text::Autoformat` is best used when called from your text editor, of course. The recommend procedure is to find a key that the editor doesn't already have mapped (or a key that you don't use anyway). Once you have found that key you need to adjust your configuration to pipe any text *from the current point onwards*. By default `Text::Autoformat` will only process the first paragraph it receives on *STDIN*.

1.2 Getting a copy of `Text::Autoformat`

- CPAN: `perl -MCPAN -e 'install Text::Autoformat'`
- Debian: `apt-get install libtext-autoformat-perl`
- Fedora Core 2: http://atrpms.net/dist/fc2/perl-Text-Autoformat/

1.3 Additional Resources

Upon installation you should be able to: `man Text::Autoformat`. This will provide you with an excellent article written by the module's author, Damian Conway, that will walk you through getting the most out of `Text::Autoformat`. Generated from the same POD sources; you may view the article online: http://search.cpan.org/~dconway/Text-Autoformat- 1.12/lib/Text/Autoformat.pm

2 SSH-Agent - Rapid, passwordless & scriptable secure logins

Anyone who manages multiple GNU/Linux systems has to be on good terms with OpenSSH. This sometimes can be tough though, especially when `ssh` demands that you type your password every time and requires you to run your scripts interactively. With a little bit of setup `ssh-agent` will save you a lot of frustration.

`ssh` without `ssh-agent`:	`ssh` with `ssh-agent`:
$ ssh dave@jupiter Password: Linux jupiter 2.6.5. . . #Login Successful	$ ssh dave@jupiter Linux jupiter 2.6.5. . . # Login Successful

Table 2: An example of `ssh` in a regular shell and a shell with `ssh-agent` as a parent

2.1 `ssh-agent` Prerequisites and Setup

`ssh-agent` requires a few things be setup before you can perform non-interactive logins. Not mentioned further in this document are the security concerns, they should be obvious once you have read this section.

2.1.1 SSH Keys

OpenSSH includes a facility to allow you to generate a *key* for use as an authentication method. Under most default setups this key may be used to authenticate(allow a user to login), rather than the account password. Additionally, these keys may carry their own password in order to protect them from un-authorized use. Creating and instituting these keys is easy and once setup using them is transparent.

- Create the Key: `ssh-keygen` is the utility you will use to create the key.

```
dave@callisto$ ssh-keygen -t dsa
Generating public/private dsa key pair.
Enter file in which to save the key (/home/dave/.ssh/id_dsa):
Enter passphrase (empty for no passphrase):
```

```
Enter same passphrase again:
Your identification has been saved in /home/dave/.ssh/id_dsa.
Your public key has been saved in /home/dave/.ssh/id_dsa.pub.
The key fingerprint is:
f9:58:14:27:0a:74:8a:dc:a0:cc:3f:b1:88:90:e7:bf dave@ganymede
```

- Use the key: Possibly unbeknownst to you, two keys were generated. One is the *public* key, one is the *private* key. The public key behaves a lot like a lock, only those with the private key can unlock the lock provided by `sshd` + the public key. We just need to place the public key (lock) on any accounts we wish to access with the private key[1]:

  ```
  $ cat ~/.ssh/id_dsa.pub | ssh dave@jupiter 'cat ->> ~/.ssh/authorized_keys'
  ```

  Next time you `ssh` to that account you should see the password prompt has changed to something like: "`Enter passphrase for key '/home/dave/.ssh/id_dsa':` ".

2.1.2 Finishing it off: `ssh-agent` & `ssh-add`

Ok, up at the top of this section I promised you *passwordless* logins—I'm now ready to make good on that promise.

`ssh-agent` is actually a daemon that stores passwords to the keys you made in the last subsection. The trick about `ssh-agent` is that it will only talk to child processes[2]. `ssh-agent` starts up not knowing any passwords at all, a program exists called `ssh-add` in order to provide it with that information.

If you're using `X` you should be able to call the two commands together with your window manger, allowing your window manager and any programs run from it to use passwordless authentication. Simply replace the current instance of your window manager in your .xinitrc or .xsession with the following snippet(adjust as necessary), restart your `X` session, type your password *once* and your off! Enjoy!

```
exec ssh-agent sh -c 'ssh-add </dev/null && /path/to/wm'
```

2.2 Getting a copy of SSH-Agent

`ssh-agent` is included with the `ssh(d)` distribution. In the context used in these notes `ssh-add` will call an additional program `ssh-askpass`. `ssh-askpass` is very popular and can likely be directly obtained from your distro.

2.3 Additional Resources

The manpages: `ssh-agent(1)`. `ssh-keygen`, `ssh-add` are very useful. There are a great many documents about setting up `ssh-agent` on the web, one I suggest is: http://mah.everybody.org/docs/ssh

3 XPlanet - Your own private spaceship

It's hard to capture in words the beauty of space. It's even harder when trying to write a *brief* introduction. Rather than waste space[3], lets try making some awe inspiring backgrounds.

3.1 Usage

Reading the documentation it quickly becomes apparent that `xplanet` has a lot of options. We can ignore most of those for now and just get started.

[1] Note: Older versions of OpenSSH used the file ~/.ssh/authorized_keys2 to store public keys. (Hint: It's *past* time to upgrade)

[2] Not quite true, talk to me for details

[3] Bad pun, I apologize. :-)

- Quick start, This should present you with a simple background of Earth centered at 0° 0°: `xplanet`
- Spy Satellite, center the image over a certain location, for example; 39° −74°: `xplanet -lat 39 -long -74`
- Custom origins, show the earth as seen from a particular (and possibly moving) point in the solar system: `xplanet -origin moon`

3.2 Additional Resources & Getting a copy of XPlanet

`xplanet`'s home page is at http://xplanet.sourceforge.net/ from there you can get a copy of `xplanet` and grab a copy of some other really nifty stuff like maps for different planets and regally rebuilt cloud maps. Hans Ecke's `xplanet` site contains a bunch of perl scripts to extend and enhance `xplanet`'s behavior. http://hans.ecke.ws/xplanet/

4 `xtitle()` - A simple bash function

Have you ever wanted to be able to know what was in an `xterm` just by looking at it's title? If so, you'll be relieved to know that it's an easy task to write a function that will allow you to rename your `xterm`'s title on-the-fly.

Without `xtitle`:	With `xtitle`:
xterm	The bash manual
BASH(1) NAME bash - GNU Bourne-Again SHell...	BASH(1) NAME bash - GNU Bourne-Again SHell...

Table 3: An example of: two `xterm`'s running `man bash` with and without `xtitle`

4.1 Control Sequence and Function Code

There are two pieces to this puzzle, our terminal emulator must be able to accept a certain "control sequence" and our shell must offer us an easy way to take advantage of that feature.

`xterm`, `rxvt` and many descendants of either of those terminal emulators allow you to change their titles in the same way. If, at a command prompt, you type: `echo -n -e "033]0;<new title>007"` you should notice your terminal's title change. If not, you may need to find another control string (or a another terminal).

Now that we know how to change our terminal's title on-the-fly we can write some spiffy GNU `bash` code. Here are some examples of what I use, but anything you can imagine is fair game. Add this to your ∼/.bashrc:

```
function xtitle ()
{
    case $TERM in
        aterm | rxvt | xterm)
            echo -n -e "\*033]0;$*\*007" ;;
        *)  ;;
    esac
}
```

Simply restart `bash` and you should now have a function `xtitle`. You can test this by typing: `type xtitle` or simply `xtitle <new title>`.

4.2 Extending the code

Extending this code to do some neat tricks is even easier; add something like the following to your ∼/.bashrc below the xtitle() function:

```
## Yay manpages!
function man ()
{
    xtitle "The $@ manual"
    /usr/bin/man $@
    xtitle xterm
}
```

Copy and paste this simple function and change the command names to get automatically titled xterms whenever you use your favorite commands.