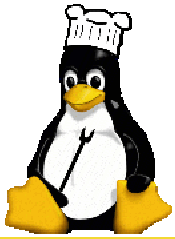
The Open Pitt

What's cooking in Linux and Open Source in Western Pennsylvania



Issue 41 June 2010 www.wplug.org

Book Review: *R in a Nutshell* by Bobbie Lynn Eicher

Author: Joseph Adler
Publisher: O'Reilly Media
ISBN13: 978-0-596-80170-0
$49.99, 640 pages, 2009

The R language and environment are GNU projects licensed under the GPL. R is designed to make it easy to do statistical calculations and create graphics representing the results. The project's web site is hosted at <http://www.r-project.org/>.

R in a Nutshell is broken into four main parts. The author begins by going over the the basics, including how to set up the R environment. There's also a brief tutorial about the workings of the language. As you progress through the book, it provides a more detailed overview of the language, discussion of how to work with data, and an explanation of how R handles statistical modeling.

Books about programming languages often make an awkward attempt to be both a tutorial and a reference. In the case of *R in a Nutshell*, however, this approach actually makes sense. It covers a specialized language that most programmers will only use when they're faced with a very particular type of problem. Since it's not the sort of language that most people will be using for their everyday programming, keeping a tutorial on hand to refresh your memory is worthwhile.

The author does what he can to make the explanations as clear as possible. Given that it's a combination of two different subjects that few people understand well—programming and statistics—it's unavoidable that the material sometimes becomes very complex. The use of sample data from a variety of fields is particularly helpful, since it makes it easier to get a sense of when the language might come in handy. Explanations of the mathematical calculations involved are brief, so if you don't already understand statistics well you'll need a separate reference focusing on that.

Most application programmers are probably never going to run into a situation where R is the best language for the job, so it's not really the type of material that's worth studying just in case you need it someday. However, if you have any projects that involve processing large amounts of data and using statistics to turn it into useful information, this is definitely a valuable resource.

Bobbie Lynn Eicher is a long-time member of WPLUG and holds a B.S. in Computer Science from the University of Pittsburgh.

O'Reilly Media provided a free electronic copy of R in a Nutshell *to Bobbie so that she could write this review. There was no other compensation involved.*

June Roundup

Jun. 12 General User Meeting: **Vance Kochenderfer** tackled a couple shell scripting exercises to demonstrate the capabilities of some of the standard UNIX text processing tools like *awk*, *sed*, *tr*, *wc*, *cat*, and *grep*. The first looked at various ways of processing a text file to determine how many non-blank lines it contained. The second exercise involved parsing a provided value to decide whether or not it could be considered numeric. A description of the exercises and some accompanying explanation of the solutions can be found on the WPLUG wiki at <http://www.wplug.org/wiki/Meeting-20100612>.

Call for Presentations

Have something to say? WPLUG holds a General User Meeting each month, and invites presentations on suitable topics.

Take this opportunity to connect with Pittsburgh's Free and Open Source software community.

Maybe you want to speak about how you use Linux or BSD in business, school, or at home.

Or you could go into the technical details of a particular utility or programming language.

Perhaps there's a new web technology or desktop framework you'd like to show off.

Or maybe Free and Open Source software allowed you to accomplish something you didn't think would be possible.

Contact the WPLUG Program Committee with your proposals at <events@wplug.org>. We look forward to hearing from you!

Coming Events

Jul. 10: General User Meeting, Topic: Extending OpenOffice.org. 10:30AM to 12:30PM, Wilkins School Community Center

Jul. 31: Installfest. 11AM to 4:30PM, Northland Public Library

Aug. 14: General User Meeting. 10:30AM to 12:30PM, Wilkins School Community Center

Aug. 29: 9th Annual Open Source Picnic. Snyder Park, Whitehall

The public is welcome at all events

UNIX Curio

This series is dedicated to exploring little-known—and occasionally useful—trinkets lurking in the dusty corners of UNIX-like operating systems.

Imagine, if you will, a Jane Austen novel about three sisters. The first is well-known and celebrated by everyone; the second, while slightly smarter and more capable, is significantly less popular; and the third languishes in near-total isolation and obscurity. These three sisters live on any UNIX-like system, and their names are *grep*, *egrep*, and *fgrep*.

We will assume you are already familiar with *grep*—*egrep* works pretty much the same, except she handles extended regular expression syntax. (When writing shell scripts intended to be portable, be careful to call *egrep* if your expression has backreferences or uses +, ?, |, or braces as metacharacters. Some versions of GNU *grep* make no distinction between basic and extended regular expressions, so you may be surprised when your script works on one system but not another.)

But our subject for today is poor, unnoticed *fgrep*. While the plainest sister of the three, she really doesn't deserve to be ignored. The "f" in her name stands either for fixed-string or fast, depending on who you ask. She does not handle regular expressions at all; the pattern she is given is taken literally. This is a great advantage when what you are searching for contains characters having special meaning in a regular expression.

Suppose you have a directory full of PHP scripts and want to find references to an array element called `$tokens[0]`. You can try *grep* (note that the single quotes are necessary to prevent the shell from interpreting `$tokens` as a shell variable):

```
$ grep '$tokens[0]' *.php
```

But there is no output. The reason is that the brackets have special significance to *grep*; `[0]` is interpreted as a character class containing only 0. Therefore, this command looks for the string `$tokens0`, which is not what we want. We would have to escape the brackets with backslashes to get the correct match (some implementations may require you to escape the dollar sign also):

```
$ grep '$tokens\[0\]' *.php
parser.php:   $outside[] =
$tokens[0];
```

Instead of fooling with all that escaping (which might get tedious if our pattern contains many special characters), we can just use *fgrep* instead:

```
$ fgrep '$tokens[0]' *.php
parser.php:   $outside[] =
$tokens[0];
```

One place where *fgrep* can be particularly handy is when searching through log files for IP addresses. With ordinary *grep*, the pattern `43.2.1.0` would match 43.221.0.123, 43.2.110.123, and a bunch of other IP addresses you're not interested in because the dot metacharacter will match any character. To make sure you only matched a literal dot you'd have to escape each one with a backslash or, better yet, use *fgrep*.

But what about the claim that *fgrep* is fast? On GNU systems, there is usually one single binary that changes its behavior depending on whether it is called as *grep*, *egrep*, or *fgrep*. (Actually, this is in line with the POSIX standard, which deprecates *egrep* and *fgrep* in favor of a single *grep* command taking the -E option for using extended regular expressions and the -F option for doing fixed-string searches.)

In testing, we found that when specifying a single pattern on the command line, *fgrep* wasn't really any faster than *grep*. However, when using the -f option to specify a file containing a list of a couple dozen patterns, *fgrep* could consistently produce a 20% time savings. On systems where *grep* and *fgrep* are different binaries, there can potentially be a more dramatic difference in speed and even memory usage.

In our hypothetical Austen novel, the neglected sister would probably be driven to a bad end, to be only spoken of afterward in hushed whispers. Don't let that happen! Whenever you need to search for a string, but don't require the power of regular expressions, get into the habit of calling on *fgrep*. She can be very helpful and deserves more attention than she gets. You'll save yourself the trouble of worrying about metacharacters and maybe some running time as well.

The Open Pitt is published by the Western Pennsylvania Linux Users Group
<http://www.wplug.org/top>

Editor: Vance Kochenderfer

What is Linux?

Linux is a *kernel*, the core of a computer operating system, created by Linus Torvalds. It is typically packaged as a *distribution*, which includes the extra programs necessary to make a computer functional and useful. Since 1991, it has grown from a one-man project which ran on one computer to one with thousands of contributors running on everything from mobile phones to million-dollar supercomputers.

What are Open Source and Free Software?

Open Source and Free Software provide you, the user, with the opportunity to see the source code of the programs you use. You are free to use it, share it with others, and even make changes to it if you wish. While the Free Software and Open Source communities differ in their philosophical approach, in practical terms they share nearly identical goals. Learn more at <http://www.opensource.org/> and <http://www.gnu.org/>.

This newsletter was produced using Open Source and Free Software.