20 Killer Perl Programming Tips for Beginners on UNIX / Linux OS

by S.K. GANESHWARI
on JANUARY 11, 2010

If you are a Linux sysadmin who writes occasional perl code (or) a developer who wants to learn perl program language, these 20 basic **perl programming tips** and tricks explained in this article will give you a jumpstart.

1. List all Installed Perl Modules from Unix Command Line

Get a list of all installed perl modules as shown below.

```
$ perl -MFile::Find=find -MFile::Spec::Functions -Tlw -e 'find { wanted =>

/usr/lib/perl5/vendor_perl/5.8.8/i386-linux-thread-multi/HTML/Filter.pm
/usr/lib/perl5/vendor_perl/5.8.8/i386-linux-thread-multi/HTML/LinkExtor.pm
/usr/lib/perl5/vendor_perl/5.8.8/i386-linux-thread-multi/HTML/PullParser.pm
/usr/lib/perl5/vendor_perl/5.8.8/i386-linux-thread-multi/HTML/Parser.pm
/usr/lib/perl5/vendor_perl/5.8.8/i386-linux-thread-multi/HTML/TokeParser.pm
.....
```

In the above example,

- **File::Find** and **File::Spec::Functions** module are used to list all installed modules.
- **-M option** loads the module. It executes *use module* before executing the script
- **-T option** enables taint checking, which instructs perl to keep track of data from the user and avoid doing anything insecure with it. Here this option is used to avoid taking the current directory name from the @INC variable and listing the available .pm files from the directory recursively.
- **-l option** enables automatic line-ending processing in the output. Print statements will have the new line separator (\n) added at the end of each line.
- **-w option** prints any warning messages.
- **-e option** indicates that the following string is to be interpreted as a perl script (i.e., sequence of commands).

2. List all Installed Perl Modules from a perl script (Using ExtUtils::Installed package)

Use the following perl code snippet to get a list of installed perl modules.

```
my $Inst = ExtUtils::Installed->new();
my @Modules = $Inst->modules();
print "Current List of Installed PERL Modules:\n\n";
foreach my $mod(@Modules){
print "$mod\n";
}
```

Note: The module ExtUtils::Installed is not pre-installed with perl program. So you should install it from cpan.

3. List Perl Modules Using Perldoc Command

perldoc perlmodlib lists all the modules that comes pre-installed with the perl program.

```
$ perldoc perlmodlib

       Attribute::Handlers Simpler definition of attribute handlers
       AutoLoader  Load subroutines only on demand
       .........
```

Note: You can also use 'perldoc perllocal' command to identify additional perl modules that are installed.

```
$ perldoc perllocal
```

perldoc perllocal command lists all optional modules installed in the system with the following information:

- Installation date
- Directory location of where the perl module is installed
- Perl Module version number
- etc.,

4. View Perl Documentation From Unix Command Line

You can use either perldoc or man command to get help about a particular perl module as shown below.

```
$ perldoc Regexp::Common

(or)

$ man Regexp::Common
```

If the perl document is not enough, use perldoc option -m, to view both source code and unformatted pod documentation of the specified perl module.

```
$ perldoc -m Regexp::Common
```

To view documentation on a specific perl function, use the option -f, as shown below.

```
$ perldoc -f splice
```

5. View Online Perl Documentation (or Download it for Offline Use)

Lot of excellent **perl tutorials** are available online at Perl programming documentation.

From this website, you can also download the HTML or PDF version of the perl documentation for offline viewing.

- Full version (contains HTML and PDF files) – perldoc.tar.gz
- Lite version (contains HTML files only) – perldoc-html.tar.gz

6. Read Perl Documentation Using Podbrowser

Download the podbrowser. You can browse the installed perl modules, functions, and perl documentation visually from the podbrowser. From their website:

```
PodBrowser is a documentation browser for Perl.
You can view, search and print documentation for Perl's
builtin functions, its "perldoc" pages, pragmatic modules
and the default and user-installed modules.
```

Note: You can also use Vim editor as Perl-IDE as we discussed earlier.

7. Modify CPAN Module Configuration Manually

To install the perl modules from Linux command line, use CPAN. We discussed earlier about how to install perl modules — both manually and using CPAN command.

The first time you use the Perl module CPAN (perl -MCPAN), a script is executed to configure several options. For example, it configures the location of tar, gzip and unzip files, the cache size for the build directory, source file location etc.,

To reconfigure CPAN module configuration files manually you can edit one of the following files.

- **~/.cpan/CPAN/MyConfig.pm** – User specific Perl CPAN Configuration file

- **/etc/perl/CPAN/Config.pm** – System-wide Perl CPAN Configuration file is stored somewhere in the perl module directory tree.

```
$ vim ~/.cpan/CPAN/MyConfig.pm

(and / or)

$ vim /etc/perl/CPAN/Config.pm
```

8. Modify CPAN Module Configurations Interactively

Launch the cpan shell as shown below and execute **o conf init**, which will ask "are you ready for manual configure", give your option (Yes / No) to continue. This will list all the configuration parameters along with it's value.

```
$ perl -MCPAN -e shell

cpan> o conf init

cpan> o conf
```

9. Verify if a Perl Module is Installed

If you like to know whether a perl module (for example, Regexp::Common) is installed, execute the following command. If you get **"1"** as the output, then the specified perl module is installed. If not, you'll get the error message as shown below.

```
$ perl -MRegexp::Common -le 'print 1'

Can't locate Regexp/Common.pm in @INC (@INC contains: /etc/perl /usr/local/
```

Note: You can also check if perl module is installed using perldoc or man command. If the perl module is installed the manual page of the module will open successfully. If not, it will say "no manual / documentation found for this module"

```
$ perldoc Regexp::Common

$ man Regexp::Common
```

10. List the Directories Where Perl Modules are Located

The **perl array @INC** contains the list of places that the 'do EXPR', 'require', or 'use' constructs look for their library files. The following one-liner shows the contents of the @INC perl array:

```
$ perl -e 'foreach $folder (@INC) { print "$folder\n";}'
```

11. Check the Version of an Installed Perl Module

To check the version number of a module use the following command.

```
#-- check the version number of CGI module
$ perl -MCGI -e 'print "$CGI::VERSION \n"'

#-- check the version number of Regexp::Common module
$ perl -MRegexp::Common -e 'print "$Regexp::Common::VERSION \n"'
```

Note: Also, make sure to read our review about the Perl Best Practices book.

12. Specify the Minimum Perl Module Version to Use

Sometimes you might want to use a specific version of a perl module in your program. To avoid using earlier version of that module, append the minimum version number you want in the use 'module' statement as shown below.

```
#-- Use version 5.8 of later of module LWP
use LWP 5.8
```

Note: Your perl program will exit with an appropriate error message if the installed module version is lower than the version you specified in the use command.

13. Useful Perl Modules to Develop Web Applications

If you are developing web application, you might want to consider using the following perl modules.

- **CGI** – General purpose module for creating web pages
- **Template** – Template Toolkit to generating dynamic web content
- **LWP** – LWP is used to fetch web contents
- **WWW::Mechanize** – Use this to automate interaction with a website

14. Determine the Operating System the Perl Script is Running Under

The name of the underlying operating system is stored in the variable $^O. Following are some of the common $^O value

- linux – Linux
- MSWin32 – Windows
- aix – AIX
- solaris – Solaris

Note: The value stored in $^O contains only the name of the operating system, not the release number. To determine the release number, consider using POSIX::uname() from POSIX package.

15. Define Constant Values Inside Perl Script

The best way to define constant values is to use Perl **Readonly** or **Constant** module as shown below.

```
use Readonly;
Readonly my $PI => 3.1415926535;

(or)

use Constant PI => 3.1415926535;
```

Please note the following:

- Using Readonly module you can define Readonly scalars, hashes and arrays.
- If you try to modify a Readonly variable, the program will die.

16. Determine the OS User and/or OS Group Running a Perl Script

Use the following predefined variables to get the user and group information of the current process:

- **$<** – real user id (uid); unique value
- **$>** – effective user id (euid); unique value
- **$(** – real group id (gid); list (separated by spaces) of groups
- **$)** – effective group id (egid); list (separated by spaces) of groups

Please note the following:

- This information applies only to Unix systems
- The values that theses variables hold are integers.

- To get the user and group names, use '(getpwuid($<))[0]' (for user information) and 'getgrgid($()' (for groups).

17. Executing External Commands

There are many ways to execute external commands from Perl as explained below.

- **system()** – you want to execute a command and don't want to capture its output
- **exec** – you don't want to return to the calling perl script
- **backticks** – you want to capture the output of the command
- **open** – you want to pipe the command (as input or output) to your perl script

18. Parse Plain Messy Perl Script Using B::Deparse

Got a chunk of obfuscated or just plain messy and hard to read Perl code? The **B::Deparse** module may be able to help. It compiles, then decompiles the program it is given, expanding it out and formatting it nicely. It removes the comment lines written in the perl program.

To run it at the command line, type "perl -MO=Deparse prog.pl". Here is an example of its usage,

First create the input program:

```
$ cat scary.pl
for(74,117,115,116){$::a.=chr};(($_.='qwertyui')&&
(tr/yuiqwert/her anot/))for($::b);for($::c){$_.=$^X;
/(p.{2}l)/;$_=$1}$::b=~/(..)$/;print("$::a$::b $::c hack$1.");
```

Pass the perl script scary.pl to the Deparse module

```
$ perl -MO=Deparse scary.pl
foreach $_ (74, 117, 115, 116) {
$a .= chr $_;
}
;
$_ .= 'qwertyui' and tr/eiqrtuwy/nr oteah/ foreach ($b);
foreach $_ ($c) {
$_ .= $^X;
/(p.{2}l)/;
$_ = $1;
}
$b =~ /(..)$/;
print "$a$b $c hack$1.";
scary.pl syntax OK
```

19. List Installed DBI Drivers

DBI is the standard database interface module for Perl. The following Perl program prints a list of all installed DBI drivers.

```
use strict;
use warnings;

use DBI;

print map "$_\n",DBI->available_drivers;
```

Example output of the above program:

```
DBM
ExampleP
File
Proxy
Sponge
mysql
```

If the driver you require is not installed, use CPAN to find the right DBD (database driver) module.

20. Regular expression special variable

One of the most useful feature or the most powerful string manipulation facility in perl is

regular expression.

At heart of this is the regular expression which is shared by many other UNIX utilities. Perl has a host of special variables that get filled after every m// or s/// regex match.

- $1, $2, $3, etc. hold the backreferences.
- $+ holds the last (highest-numbered) backreference.
- $& (dollar ampersand) holds the entire regex match.
- $' (dollar followed by an apostrophe or single quote) holds the part of the string after (to the right of) the regex match.
- $` (dollar backtick) holds the part of the string before (to the left of) the regex match.

Using these variables is not recommended in perl scripts when performance matters, as it causes Perl to slow down *all* regex matches in your entire perl script.

All these variables are read-only, and persist until the next regex match is attempted. Example

```
$string = "This is the geek stuff article for perl learner";
$string =~ /the (g.*) stuff(.*) /;
print "Matched String=>$&\nBefore Match=>$`\nAfter Match=>$'\nLast Paren=>$
```

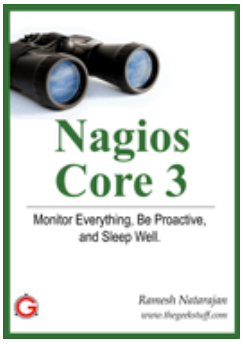
Output of the above example :

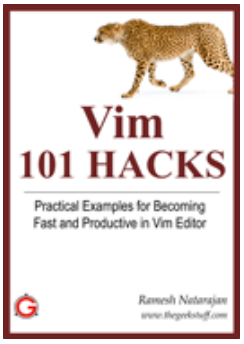
```
Matched String=>the geek stuff article for perl
Before Match=>This is
After Match=>learner
Last Paren=> article for perl
First Paren=>geek
```


If you enjoyed this article, you might also like..

1. 50 Linux Sysadmin Tutorials
2. 50 Most Frequently Used Linux Commands (With Examples)
3. Top 25 Best Linux Performance Monitoring and Debugging Tools
4. Mommy, I found it! – 15 Practical Linux Find Command Examples
5. Linux 101 Hacks 2nd Edition eBook Free

- Awk Introduction – 7 Awk Print Examples
- Advanced Sed Substitution Examples
- 8 Essential Vim Editor Navigation Fundamentals
- 25 Most Frequently Used Linux IPTables Rules Examples
- Turbocharge PuTTY with 12 Powerful Add-Ons





Tagged as: /etc/perl/CPAN/Config.pm, ~/.cpan/CPAN/MyConfig.pm, Perl Command, Perl Tutorial

{ 4 comments… add one }

Fredo

I think you should mention the following Modules in the Web Developement section:

For serious modern Perl Web Developement:
Catalyst http://search.cpan.org/dist/Catalyst-Runtime/

For more lightweight applications:
Titanium http://search.cpan.org/dist/Titanium/
(this is only a convenient bundle of the venerable CGI::Application)

A very interesting new Module for lightweight applications:
Dancer http://search.cpan.org/dist/Dancer/

All these Modules require the installation of other CPAN-Components. A very comfortable way to get this done is to use
local::lib http://search.cpan.org/dist/local-lib/

LINK

Ramesh Natarajan

@Fredo,

Thanks for adding more modules to the list. All of them you've mentioned looks very good.

LINK

sudershan

WORK=`Urgent/WEB-INF/lib/*.jar`

for c in $WORK
do
CLASSPATH="$CLASSPATH:$c"
done

Please find the attached above shell-script , it is very urgent to re-write the code the perl script . please Ramesh Natarajan or any perl scripters , please convert the above shell script to contents to perl script..

Please help in this regard , very very urgent

LINK

Leave a Comment

Name

Email

Website

Comment

Save my name, email, and website in this browser for the next time I comment.

Submit

Notify me of followup comments via e-mail

Next post: Yet Another Sudoku Puzzle Solver Using AWK
Previous post: 6 Steps for Minimal Ubuntu Installation Using debootstrap

RSS | Email | Twitter | Facebook | Google+

Search

POPULAR POSTS

15 Essential Accessories for Your Nikon or Canon DSLR Camera

12 Amazing and Essential Linux Books To Enrich Your Brain and Library

50 UNIX / Linux Sysadmin Tutorials

50 Most Frequently Used UNIX / Linux Commands (With Examples)

How To Be Productive and Get Things Done Using GTD

30 Things To Do When you are Bored and have a Computer

Linux Directory Structure (File System Structure) Explained with Examples

Linux Crontab: 15 Awesome Cron Job Examples

Get a Grip on the Grep! – 15 Practical Grep Command Examples

Unix LS Command: 15 Practical Examples

15 Examples To Master Linux Command Line History

Top 10 Open Source Bug Tracking System

Vi and Vim Macro Tutorial: How To Record and Play

Mommy, I found it! -- 15 Practical Linux Find Command Examples

15 Awesome Gmail Tips and Tricks

15 Awesome Google Search Tips and Tricks

RAID 0, RAID 1, RAID 5, RAID 10 Explained with Diagrams

Can You Top This? 15 Practical Linux Top Command Examples

Top 5 Best System Monitoring Tools

Top 5 Best Linux OS Distributions

How To Monitor Remote Linux Host using Nagios 3.0

Awk Introduction Tutorial – 7 Awk Print Examples

How to Backup Linux? 15 rsync Command Examples

The Ultimate Wget Download Guide With 15 Awesome Examples

Top 5 Best Linux Text Editors

Packet Analyzer: 15 TCPDUMP Command Examples

The Ultimate Bash Array Tutorial with 15 Examples

3 Steps to Perform SSH Login Without Password Using ssh-keygen & ssh-copy-id

Unix Sed Tutorial: Advanced Sed Substitution Examples

UNIX / Linux: 10 Netstat Command Examples

The Ultimate Guide for Creating Strong Passwords

6 Steps to Secure Your Home Wireless Network

Turbocharge PuTTY with 12 Powerful Add-Ons

CATEGORIES

Linux Tutorials

Vim Editor

Sed Scripting

Awk Scripting

Bash Shell Scripting

Nagios Monitoring

OpenSSH

IPTables Firewall

Apache Web Server

MySQL Database

Perl Programming

Google Tutorials

Ubuntu Tutorials

PostgreSQL DB

Hello World Examples

C Programming

C++ Programming

DELL Server Tutorials

Oracle Database

VMware Tutorials

About The Geek Stuff

My name is **Ramesh Natarajan**. I will be posting instruction guides, how-to, troubleshooting tips and tricks on Linux, database, hardware, security and web. My focus is to write articles that will either teach you or help you resolve a problem. Read more about Ramesh Natarajan and the blog.

Contact Us

Email Me : Use this Contact Form to get in touch me with your comments, questions or suggestions about this site. You can also simply drop me a line to say hello!.

Follow us on Google+

Follow us on Twitter

Become a fan on Facebook

Support Us

Support this blog by purchasing one of my ebooks.

Bash 101 Hacks eBook

Sed and Awk 101 Hacks eBook

Vim 101 Hacks eBook

Nagios Core 3 eBook