Chapter No. 1
"Taming vi"
In this package, you will find:

The author’s biography
A preview chapter from the book, Chapter no.1 "Taming vi"
A synopsis of the book’s content
Information on where to buy this book

About the Author

Andrew Mallett has worked in the IT field for more years than he cares to mention, well, since 1986, and with Linux technologies in Red Hat Linux 7 since 1999. Not only does he have Linux skills and certification, he consults and teaches Linux and other technologies and has had a book published with Packt Publishing on Citrix. He has also been an active participant in support communities, and works as a volunteer sysop on the SUSE Linux instructor to help, support, and develop the official Novell SUSE curriculum worldwide.

Andrew currently works for his own company and can be contacted at http://theurbanpenguin.com and @theurbanpenguin on Twitter. Video courses on Linux that he has published can be found at http://www.pluralsight.com.

I would like to thank Say Mistage (available on Twitter at @sayomgwtf) for keeping me sane with all of her doodles and inspiration during the writing of this book. Let me say that there are a few people in this world who suffer that never should. These people are often the most inspirational and happy people you find. Say is one of those people who suffers a lot in life but never lets it show.

For More Information:
www.packtpub.com/networking-and-servers/centos-system-administration-essentials
CentOS System Administration Essentials

Welcome to CentOS System Administration Essentials. My name is Andrew Mallett, and I will be offering you expert guidance and tuition, enabling you with the skills to tame this powerful and popular Linux distribution. I have chosen to write about CentOS primarily as it will not cost you to use it, neither while learning nor during production. Additionally, CentOS closely follows the Red Hat Enterprise Linux distribution, so the skills that you learn and develop here can be put to good use across both CentOS and Red Hat. Should you be interested, your reading can act as an investment in your career by pursuing the Red Hat certification paths. Although not directly written to fit into any existing curricula, the Red Hat exams are all based on practical exercises, so the more you know and understand about the operation of Linux, the better.

CentOS stands for Community Enterprise Operating System, and even though community is such a small word, it encompasses so much. The support emanates from the community, via fora and the Linux community, to help develop the services and applications, and provide remedies to bugs that occur. The community has taken ownership of this distribution. The distribution collectively becomes stronger with the continued involvement of a growing community.

While we talk of community, I would like to thank Say Mistage (@sayomgwtf) for her inspiration and doodles. Writing about an Enterprise Linux distribution is important as we see the increase in the number of organizations deploying Linux and, as a result, require knowledgeable professionals to manage these systems. In 2013, the Linux Foundation with Dice, a specialist recruitment company, surveyed many large organizations and found the following results:

- 93 percent of the organizations polled were looking to employ Linux professionals
- 91 percent of hiring managers reported that they found it difficult to find skilled Linux administrators
- As a side note to this, it was additionally noted that salaries for Linux professionals had increased by 9 percent during the previous 12 months

With such confidence in Linux within so many organizations, the focus of this book has to be commercially driven for both myself and you, the reader. I want you to be able to improve your career prospects as well as your Linux knowledge.

For More Information:
www.packtpub.com/networking-and-servers/centos-system-administration-essentials
Enterprise Linux distributions such as CentOS, Red Hat, Debian, and SUSE Enterprise Linux generally do not deploy the latest and greatest bleeding edge technology that you might find in home or enthusiast-oriented distributions such as Fedora or openSUSE. Rather, they allow these to be development platforms to hone and perfect the software before migrating it to the enterprise platforms some months or even years later. Enterprise Linux has to be dependable, reliable, and resilient. On top of this, it must be well supported by both the organization deploying it, as well as the backend support coming from the community or paid support teams. The very latest in software development does not lend itself well to this by definition; as they are the most recent, the knowledge of these advancements, as well as their best practices, will without a doubt take time to evolve and develop.

What This Book Covers

Chapter 1, *Taming vi*, will make sure that you are fully versed in the shortcuts that exist to make your shell quickly navigable before entering into the realms of mastering vi. You may have some experience with vi but most often, I find that the experience has not been a good one. I am going to make sure that you are the master of vi and not vice versa.

Chapter 2, *Cold Starts*, is all about understanding the boot process in CentOS and learning how to not only modify the GRUB menu to make it more secure, but also how to use the GRUB command line to debug and repair boot issues. We will include a little boot splashing with Plymouth as well as explain when the root filesystem is not actually the root filesystem.

Chapter 3, *CentOS Filesystems – A Deeper Look*, tells us that we have files and directories but they are all just different file types. However, when it comes to links, pipes, and sockets, we will discuss what they are and how they are used. Regarding links, we will discuss what is the difference between a hard and soft link. Let's also challenge the traditional filesystem design; you may have worked with logical volumes manager (LVM) in the past, but let me tell you just how last century that is. You are going to be blown away by the power and ease of your enterprise filesystem management using BTRFS, pronounced as Better FS.

Chapter 4, *YUM – Software Never Looked So Good*, gets you to grips with YUM repositories and software management; you are going to love this. You will learn how to download packages without installing them, thus allowing you to easily distribute packages in your enterprise. If this is not good enough, then you'll learn how to set up a local repository to share packages across your LAN and create your own RPMs.

For More Information:
[www.packtpub.com/networking-and-servers/centos-system-administration-essentials](http://www.packtpub.com/networking-and-servers/centos-system-administration-essentials)
Chapter 5, Herding Cats – Taking Control of Processes, tells us that too often, administrators, without the insight that you and I have, will leave services running that aren't required, and do not understand the tools they have to manage processes. You will learn here to control services and processes using upstart and traditional service scripts as well as become homicidal with the kill and pkill weapons of choice.

Chapter 6, Users – Do We Really Want Them?, tells us, of course, that we do not want them (users) on our system, but it is often dictated, so we have little choice. Rather than be grumpy about this, you will learn how to manage users with a smile and keep them on a tight rein.

Chapter 7, LDAP – A Better Type of User, tells us that rather than having silos of users and groups on each machine, it is better to get back on the golf course by spending more time improving the system and less time managing users. Adding users to a central directory and sharing them across all systems as required is your gateway to freedom.

Chapter 8, Nginx – Deploying a Performance-centric Web Server, tells us that commonly, Linux administrators and publications concentrate on the Apache web server; I will introduce you to the new kid on the block, Nginx (pronounced Engine X). Introduced in 2004, Nginx is rapidly taking market share from Apache and has already surpassed IIS in a number of deployed web servers worldwide. We will deploy Nginx and PHP.

Chapter 9, Puppet – Now You Are the Puppet Master, shifts our focus from Linux in the enterprise to taking control of your enterprise systems with the renowned Puppet software from Puppet Labs. Central configuration control is as good as centralized user management in giving you more time to spend on the golf course, not that I want you to think that golf dominates my life.

Chapter 10, Security Central, introduces you to Pluggable Authentication Modules (PAM). It is your friend and will help you manage when and how users connect. SELinux, again, is a friend, albeit a temperamental one. When treated well, it will help you ensure correct use of your system. You will learn how to harden your Linux system and gain a set of best practices!

Chapter 11, Graduation Day, tells us that as we prepare to leave with our newfound skills, we will remind ourselves the need for security and adhere to the best practices. We can revisit some of the products that we have seen before, such as Puppet and Nginx, and outline some industry-recognized guidelines for the deployment of these services, along with some of the new features of CentOS 7.

For More Information:  
www.packtpub.com/networking-and-servers/centos-system-administration-essentials
You may have some experience with vi, or what is now known as Vim (which is when simply put—vi improved). All too often, I find that those first experiences have never been good ones or to be looked back upon with much fondness. Guiding you through the initially unfathomable regime of vi, we are going to make sure that you are the master of vi and you leave wanting to use this tool from the gods. vi is like everything else in the sense that you just need to stick with it in the early days and keep practicing. Remember how you persevered for many hours riding your bicycle as a toddler and became a master, despite a few bruised knees? I want you to persevere with vi too. We will start with a little command-line magic to make the whole command-line interface (CLI) experience a better one. We will then be ready to start our black-belt experience in vi.

In this chapter, we will go through the following topics:

- **CLI trickery** – shortcuts that you will love
- **Vim and vi**: In this section, you will learn to differentiate between these twins and meet their graphical cousin
- **Getting the .vimrc setup the way you like**
- **Search and replace**: In this section, you will learn how to quickly find and replace text within files from both inside and outside Vim
- **Learning to remove extraneous comments from a file with a few deft key strokes**
CLI trickery – shortcuts that you will love

So before we dice into the wonderful world of text editing that is vi, we will warm up with a few exercises on the keyboard. Linux is my passion, as is automation. I am always keen to create scripts to carry out tasks so that those tasks become repeatedly correct. Once the script is created and tested, we will have the knowledge and faith that it will run in the same way every time and we will not make mistakes or miss critical steps, either because it gets boring or we are working late on a Friday night and just want to go home. Scripting itself is just knowing the command line well and being able to use it at its best. This truth remains across all systems that you will work with.

On the command line, we may try a little more black magic by executing the following command:

```bash
$ cd dir1 || mkdir dir1 && cd dir1
```

With this, we have used the `cd` command to enter the `dir1` directory. The double pipe or vertical bar indicates that we will attempt the next command only if the first command fails. This means that if we fail to switch to the `dir1` directory, we will run the `mkdir dir1` command to create it. If the directory creation succeeds, we then change into that directory.

The `||` part denotes that the second command will run only on the failure of the first. The `&&` part denotes that the second command will run only if the first command succeeds.

The command history is a little more and hugely better than just an up arrow key! Consider the following commands:

```bash
$ mkdir dir1
cd !$
```

The `$!` part represents the last argument, so in this way, the second line evaluates to the following:

```bash
$ cd dir1
```

In this way, we can rewrite the initial command sequence, by combining both concepts, to create the following command:

```bash
$ cd dir1 || mkdir !$ && cd !$
```

For More Information:
www.packtpub.com/networking-and-servers/centos-system-administration-essentials
We can repeat the last command as well as the last argument. More importantly, we can specify the start characters for the last command. If it was merely the last command, then the up arrow key would suffice. If we were working on a web server configuration, we may want to edit the configuration file with vi, start the service, and then test with a command-line browser. We can represent these tasks using the following three commands:

```
# vi /etc/httpd/conf/httpd.conf
# service httpd restart
w3m localhost
```

Having run these three commands in the correct order, hoping for success, we may notice that we still have issues and that we need to start re-editing the configuration file for Apache, the web server. We can now abbreviate the command list to the following:

```
# !v
# !s
# !w
```

The `!v` command will rerun the last command in my history that begins with a `v`, and likewise with `s` and `w`. This way, we can appear to be terribly proficient and working really quickly, thus gaining more time to do what really interests us, perhaps a short 9 holes?

In a similar fashion to our first glance at the history using the `!$` symbols to represent the last argument, we can use `!?73`. This would look for `73` anywhere as an argument or part of an argument. With my current history, this would relate to the date command we ran earlier. Let's take a look:

```
$ !?73
```

With my history, the sequence will expand to and run the following command:

```
$ date --date "73 days ago"
```

Looking at my command history from the last command run to the first, we search for `73` anywhere as a command argument. We make a note that we exclusively look for `73`, meaning we are looking for the character `7` followed by the character `3`. We have to then bear in mind that we would also match `273` or `733` if they existed in my history.

Having mastered a little of the Bash shell history functions, we should practice to make this second nature.

For More Information:
www.packtpub.com/networking-and-servers/centos-system-administration-essentials
Ah yes, Vim and vi! They sound like some ancient mystic potion that ensures long life and wisdom. Alas though, they are not.

The command-line text editor vi was first written in 1976 and became part of the first release of BSD Unix in 1978. Even though it is command line driven and with no Graphical User Interface (GUI) or menu, a 2009 survey conducted by Linux Journal found that vi was the most popular editor, beating even gedit, the GUI GNOME editor, into second place. I am not averse to the GUI, but I find a GUI editor to be restrictive and slow. I can honestly say that the majority of, if not all, tasks can be performed by me more quickly in vi.

That being said, in CentOS, you will not find vi; vi is purely a default alias that is provided for convenience, and links to the vim command. We can view this on my CentOS 6.5 console using the following command:

```
$ alias | grep vi
```

The output of the command should look similar to the following screenshot:

```
[andrew@centos ~]$ alias | grep vi
alias vi='vim'
[andrew@centos ~]$
```

Vim is a contraction of **Vi IMproved** and was first publicly released in 1991 and authored by Bram Moolenaar, initially targeted at the Amiga system. It has been common in the Linux platform since the early 2000s. As the name suggests, it is based on vi and is improved; on CentOS, it is distributed with the vim-enhanced package. These improvements are most commonly useful with the syntax-highlighting feature available for languages such as PERL, Python, and PHP. Another such improvement is that it can work traditionally on the command line or with a GUI frontend. To install the graphical interface for Vim, you will need to add the `vim-X11` package as follows:

```
# yum install -y vim-X11
```

One limitation, of course, is that you will require the X11 server to be running. In an enterprise, the server will often run without a GUI and you can connect using secure shell to a command-line shell only.

For More Information:
www.packtpub.com/networking-and-servers/centos-system-administration-essentials
If you are new to vi, then using the graphical version can be helpful, as the menus also display the command-line shortcuts. To edit a file with vi or Vim on the command line, we can simply use a command similar to the following:

$$\text{vi <filename-to-edit}>$$

It is possible to use the graphical version of an editor when you are working on the CentOS desktop as follows:

$$\text{gvim <filename-to-edit}>$$

or

$$\text{vimx -g <filename-to-edit}>$$

I would recommend using the gvim command, as it doesn't require the additional option and causes less confusion. Starting vimx without the -g option just starts the normal Vim program.

**Getting the .vimrc setup the way you like**

As with many programs in Linux, Vim has the option to read settings from a run-control file. This can be centralized via the /etc/vimrc file, or for each user via the ~/.vimrc file. With this file, especially with our own version, you can customize how Vim appears and controls its functionalities.

Firstly, we will look at line numbering. Often when we edit a file, we do so as the console has reported an error on a particular line just after we have tried running a script or starting a service; we know we have a syntax error. Let's say we want to go directly to the offending line 97 of the test.php file. Then, we would duly type:

$$\text{vi +97 test.php}$$

This is assuming that we were in the same directory as our file. Similarly, should we want to go directly to the first occurrence of the word **install** within the readme file, we could issue the following command:

$$\text{vi +/install readme}$$

Then, as if by magic, we are transported to the correct line that we require. However, in the case of the word **search**, the word that was **search** is highlighted in color. If that is not desirable, then we can simply turn off that feature. Within Vim, we can type:

```
:nohlsearch
```

For More Information:

www.packtpub.com/networking-and-servers/centos-system-administration-essentials
If there are settings that we want to make permanent within Vim, we can edit the `.vimrc` file in our home directory. This is our own personal settings file and as such, changes made here will not affect anyone else. If we want to affect system-wide settings, then we can use the `/etc/vimrc` file. Try adding the following line to the `~/.vimrc` file to persistently disable the highlight search:

```
set nohlsearch
```

With this addition, each time we start Vim, the setting is ready for us. As we view our files though, from within Vim, we may prefer to have line numbering turned on. Sometimes this makes life easier, but other times, we may prefer to have line numbering off, especially in cases where we have lines starting with numbers (because the display can become confusing). To enable line numbering, run the following command:

```
:set number
```

To turn line numbering off, we can use the following command:

```
:set nonumber
```

As before, we can always put the desired start-up value in the `.vimrc` file. However, before we do this, let’s look at key mappings within Vim and how we can create a shortcut to toggle line numbering on and off. We would like to create a mapping for the normal mode in Vim. This is the mode when we first enter Vim and we are not editing, just navigating the file; using the `Esc` key, we can always return to the normal mode. Execute the following command:

```
:nmap <C-N> : set invnumber<CR>
```

The `nmap` command denotes that we are making a mapping for the normal mode only. We are mapping the `Ctrl + N` keys to run the sub command `:set invnumber` followed by `<CR>`.

With this in place, we can now use the combination of `Ctrl + N` to toggle line numbering on and off. Now we are really starting to make some steam with this product, and you can gain some appreciation of why it is so popular. Before we make the final edit to the `.vimrc` file, we will see how to navigate lines by number while in vi or Vim. Making sure that we are in the normal mode using the `Esc` key, we can use `2G` or `2gg` to navigate to line 2 of the current file; likewise, `234G` or `234gg` would go to line 234 and `G` or `gg` would navigate to the end of the file. Simple but not simple enough; I would prefer to type the line number followed by the `Enter` key. For this, we map the `Enter` key to `G`. If we choose to use the `Enter` key without a preceding number, then we are taken directly to the end of the document, just as we would is we used the key `G` by itself. Execute the following command:

```
:nmap <CR> G
```

For More Information:

[www.packtpub.com/networking-and-servers/centos-system-administration-essentials](http://www.packtpub.com/networking-and-servers/centos-system-administration-essentials)
Now we simply type in the desired line number followed by Enter. This in turn is interpreted as the number followed by G. In this way, we can navigate easily to the correct line. We can persist this setting by adding the following text to the .vimrc file, which should now read similar to the following text as we review all the settings made within this subsection:

```
set nohlsearch number
nmap <C-N> : set invnumber<CR>
nmap <CR> G
```

Now sit back and enjoy what you have achieved, remembering though that practice is the key to knowledge being retained.

### Search and replace

So we are not exactly on a "search and destroy" mission, but if it helps by adding a little enjoyment to our learning, then we can embark upon a search and replace mission. Linux has a huge amount of power available on the command line and nothing less than the stream editor, sed. Even without entering the Vim editor, we can search for and replace text in a single file or even across multiple files. Not having to use an interactive editor opens up more administrative scope to us by being able to script updates across a single or many servers. The functionality we have in the `sed` command is available to us for use from within Vim or as a standalone application. We will be learning in this subsection how to search for and replace text within files using `sed` and from within Vim, building skills that we can use across CentOS and other operating systems including OS X on the Mac.

Firstly, let's take a scenario that we have recently changed our company name and we need to change all the references of Dungeons in a text document to Dragons. Using sed, we could run the command directly from the console:

```
$ sed -i 's/Dungeons/Dragons/g' /path/file
```

This will read the file line by line, replacing all occurrences of the string Dungeons with Dragons. The `-i` option allows for in-place edits, meaning we edit the file without the need to redirect the output from sed to a new file. The `g` option allows for the replacement to occur across all instances of Dragon even if it appears more than once per line.

To do the same within Vim where we have the file open, run the following command:

```
:%s/Dungeons/Dragons/g
```

For More Information:

www.packtpub.com/networking-and-servers/centos-system-administration-essentials
The percent symbol is used to specify the range as the whole document; whereas if we use the following command, we would only search lines 3 through 12 inclusive of the search string. In this case, the range is said to be lines 3 to 12 whereas with %, the range is the complete document.

`:3,12s/Dungeons/Dragons/g`

The range can be very useful when perhaps we want to indent some code in a file. In the following line, we again search lines 3 through to 12 and add a Tab to the start of each line:

`:s/3,12s/^/\t/`

We have set the range in the previous command within Vim to represent lines 3 to 12 again. These lines may represent the contents of an if statement, for example, that we would like to indent. We search first for the carat symbol, ^ (the start of a line), and replace it with a tab (\t). There is no need for the global option as the start of a line obviously only occurs once per line. Using this method, we can quickly add indents to a file as required, and we are again Zen superheroes of Vim.

---

**Learning to remove extraneous comments from a file with a few deft key strokes**

Now that we are the administrator, the Zen master of search and replace, we can use these skills to tidy configuration files that often have many hundreds of commented lines within them. I do not mind documentation but when it becomes such an overwhelming majority, it can take over. Consider the httpd.conf Apache configuration file under `/etc/httpd/conf/`. This has 675 commented lines. We perhaps want to keep the original file as a reference. So let's first make a copy by executing the following command; we know how to do this from the Preface of this book and if you did not read it, now is your chance to read it before a letter goes home to your parents.

```
# cd /etc/httpd/conf
# cp httpd.conf httpd.conf.$(date +%F)
```

We can easily list the commented lines using the following command that counts the lines that begin with the # sign, a comment:

```
# egrep -c '^#' httpd.conf
```

---

For More Information:

[www.packtpub.com/networking-and-servers/centos-system-administration-essentials](http://www.packtpub.com/networking-and-servers/centos-system-administration-essentials)
On my system, we see that there are 675 such lines. Using sed or Vim, we can remove the comments, firstly, with sed, as follows:

```
# sed  -i '/^#/d' httpd.conf
```

Then, within Vim with the file open, it is a little different:

```
:g/^#/d
```

The result is the same in both examples where we have reduced the numbers of lines in the file by about two-thirds.

**Summary**

In each chapter, I want to make sure that there has been at least one item of value that you feel you can take away with you and use; how did I do in this chapter? If you recall, we have reviewed a few shortcuts that may help us navigate the command history effectively. Quickly, we moved on to discover the text editor vi or, more commonly now, Vim. For those that need a little help getting started with Vim, we additionally have gVim available to use if we are working on the desktop. Customizing any system is important to make us feel that we own the system and it works for us. With Vim, we can use the `.vimrc` file found in our home directory. We were able to add a little bling to Vim with some extra key mapping and desirable options. From then on, it was straight down to work to see what Vim could do, and how the search and replace and delete options that we reviewed worked.
Where to buy this book


Free shipping to the US, UK, Europe and selected Asian countries. For more information, please read our shipping policy.

Alternatively, you can buy the book from Amazon, BN.com, Computer Manuals and most internet book retailers.