

The Software Developer's Guide to Linux

A practical, no-nonsense guide to using the Linux command line and utilities as a software developer



David Cohen
Christian Sturm



The Software Developer's Guide to Linux

A practical, no-nonsense guide to using the Linux command line and utilities as a software developer

David Cohen

Christian Sturm

<packt>

BIRMINGHAM—MUMBAI

Contributors

About the authors

David Cohen has, for the past 15 years, worked as a Linux system administrator, software engineer, infrastructure engineer, platform engineer, site reliability engineer, security engineer, web developer, and a few other things besides. In his free time, he runs the *tutoriallinux* YouTube channel where he's taught hundreds of thousands of people the basics of Linux, programming, and DevOps. David has been at Hashicorp since 2019—first as an SRE, then as a reference architect, and now as a software engineer.

Thank you, Aleyna, for your unwavering support over the past few years as I've been developing and writing this book. Without you, this would just be another of my promising-but-unfinished projects languishing in some forgotten "Archive" directory. Thanks to Christian, who has stuck with me for over a decade as a friend and a partner on practically every wild tech project idea I've come up with since we met. Finally, a big "thank you" is also due to my friends and colleagues at Hashicorp and everywhere else I've been over the past 15 years, who have made me a better engineer and encouraged projects like this.

Christian Sturm is a consultant on software and systems architecture, having worked in various technical positions for well over a decade. He has worked as an application developer for the frontend and backend at companies large and small, such as zoomsquare and Plutonium Labs. On top of that, he is also an active contributor to various open source projects and has a deep understanding of fields including operating systems, networking protocols, security, and database management systems.

About the reviewers

Mario Splivalo works as a consultant dealing with databases extended into modern cloud-based architectures. He also helps companies design their infrastructure using IaaS tools such as *Terraform* and *AWS CloudFormation*. For five years, Mario worked with *Canonical* as an OpenStack engineer.

Mario's fascination with computers started back when Commodore 64 dominated the user space. He took his first steps using BASIC on his dad's C64, quickly shifting to Assembler. He gradually moved to PCs, finding a great love for programming, systems design, and database administration. He switched to Linux (Knoppix, then Ubuntu, and never looked back) in the early 2000s, continuing as a database administrator, programmer, and system administrator.

Nathan Chancellor is an independent contractor working on the Linux kernel, based in Arizona, US. As a developer, his focus is on improving the compatibility between the Linux kernel and the LLVM toolchain. He has used Linux since 2016 and it has been his primary development operating system since 2018. His distributions of choice are Arch Linux and Fedora.

Table of Contents

| | |
|---|--------------|
| Preface | xxiii |
| <hr/> | |
| Chapter 1: How the Command Line Works | 1 |
| <hr/> | |
| In the beginning...was the REPL | 1 |
| Command-line syntax (read) | 3 |
| Command line vs. shell | 4 |
| How does the shell know what to run? (evaluate) • 5 | |
| A quick definition of POSIX • 6 | |
| Basic command-line skills | 7 |
| Unix filesystem basics • 7 | |
| Absolute vs. relative file paths • 8 | |
| <i>Absolute vs. relative pathname review</i> • 10 | |
| <i>Opening a terminal</i> • 10 | |
| Looking around – command-line navigation • 10 | |
| <i>pwd - print working directory</i> • 10 | |
| <i>ls - list</i> • 11 | |
| Moving around • 12 | |
| <i>cd – change directory</i> • 12 | |
| <i>find – find files</i> • 13 | |
| Reading files • 13 | |
| <i>less – page through a file</i> • 14 | |
| Making changes • 14 | |

| | |
|---|-----------|
| <i>touch</i> – create an empty file, or update modification time for an existing one • 14 | |
| <i>mkdir</i> – create a directory • 14 | |
| <i>rmdir</i> – remove empty directories • 15 | |
| <i>rm</i> – remove files and directories • 15 | |
| <i>mv</i> – move or rename files and directories • 16 | |
| Getting help | 17 |
| Shell autocompletion | 18 |
| Conclusion | 20 |
| | |
| Chapter 2: Working with Processes | 21 |
| <hr/> | |
| Process basics | 22 |
| What is a Linux process made of? • 23 | |
| Process ID (PID) • 24 | |
| Effective User ID (EUID) and Effective Group ID (EGID) • 25 | |
| Environment variables • 25 | |
| Working directory • 25 | |
| Practical commands for working with Linux processes | 26 |
| Advanced process concepts and tools | 28 |
| Signals • 28 | |
| <i>Practical uses of signals</i> • 29 | |
| <i>Trapping</i> • 29 | |
| <i>The kill command</i> • 29 | |
| <i>lsof</i> – show file handles that a process has open • 31 | |
| Inheritance • 33 | |
| Review – example troubleshooting session | 33 |
| Conclusion | 35 |
| | |
| Chapter 3: Service Management with systemd | 37 |
| <hr/> | |
| The basics | 38 |
| init • 39 | |
| Processes and services | 39 |

| | |
|--|-----------|
| systemctl commands | 40 |
| Checking the status of a service • 40 | |
| Starting a service • 41 | |
| Stopping a service • 41 | |
| Restarting a service • 41 | |
| Reloading a service • 42 | |
| Enable and disable • 42 | |
| A note on Docker | 43 |
| Conclusion | 43 |
| | |
| Chapter 4: Using Shell History | 45 |
| <hr/> | |
| Shell history | 45 |
| Shell configuration files • 46 | |
| History files • 46 | |
| Searching through shell history • 47 | |
| Exceptions • 47 | |
| Executing previous commands with ! | 48 |
| Re-running a command with the same arguments • 48 | |
| Prepending a command to something in your history • 48 | |
| Jumping to the beginning or end of the current line | 49 |
| Conclusion | 49 |
| | |
| Chapter 5: Introducing Files | 51 |
| <hr/> | |
| Files on Linux: the absolute basics | 52 |
| Plaintext files • 52 | |
| What is a binary file? • 52 | |
| Line endings • 53 | |
| The filesystem tree | 53 |
| Basic filesystem operations | 54 |
| ls • 54 | |
| pwd • 55 | |

| | |
|---|-----------|
| cd • 55 | |
| touch • 56 | |
| less • 57 | |
| tail • 57 | |
| mv • 57 | |
| <i>Moving</i> • 57 | |
| <i>Renaming</i> • 58 | |
| cp • 58 | |
| mkdir • 58 | |
| rm • 58 | |
| Editing files | 59 |
| File types | 60 |
| Symbolic links • 61 | |
| Hard links • 62 | |
| The file command • 62 | |
| Advanced file operations | 63 |
| Searching file content with grep • 63 | |
| Finding files with find • 64 | |
| <i>Copying files between local and remote hosts with rsync</i> • 65 | |
| <i>Combining find, grep, and rsync</i> • 66 | |
| Advanced filesystem knowledge for the real world | 67 |
| FUSE: Even more fun with Unix filesystems • 68 | |
| Conclusion | 69 |
| | |
| Chapter 6: Editing Files on the Command Line | 71 |
| <hr/> | |
| Nano | 72 |
| Installing nano • 72 | |
| Nano cheat sheet • 72 | |
| <i>File handling</i> • 73 | |
| <i>Editing</i> • 73 | |
| <i>Search and replace</i> • 73 | |

| | |
|--|-----------|
| Vi(m) | 73 |
| Vi/vim commands • 74 | |
| Modes • 74 | |
| <i>Command mode</i> • 74 | |
| <i>Normal mode</i> • 75 | |
| Tips for learning vi(m) • 76 | |
| <i>Use vimtutor</i> • 76 | |
| <i>Think in terms of mnemonics</i> • 76 | |
| <i>Avoid using arrow keys</i> • 76 | |
| <i>Avoid using the mouse</i> • 77 | |
| <i>Don't use gvim</i> • 77 | |
| <i>Avoid starting with extensive configuration or plugins</i> • 77 | |
| Vim bindings in other software • 79 | |
| Editing a file you don't have permissions for | 79 |
| Setting your preferred editor | 79 |
| Conclusion | 80 |
| | |
| Chapter 7: Users and Groups | 81 |
| <hr/> | |
| What is a user? | 81 |
| Root versus everybody else | 82 |
| sudo | 82 |
| What is a group? | 84 |
| Mini project: user and group management | 84 |
| Creating a user • 85 | |
| Create a group • 86 | |
| Modifying a Linux user • 86 | |
| <i>Adding a Linux user to a group</i> • 87 | |
| <i>Removing a user from a group</i> • 87 | |
| <i>Removing a Linux user</i> • 87 | |
| <i>Remove a Linux group</i> • 87 | |
| Advanced: what is a user, really? | 88 |
| User metadata / attributes • 88 | |

| | |
|--|------------|
| A note on scriptability | 90 |
| Conclusion | 90 |
| Chapter 8: Ownership and Permissions | 93 |
| Deciphering a long listing | 93 |
| File attributes | 94 |
| File type • 94 | |
| Permissions • 95 | |
| Number of hardlinks • 95 | |
| User ownership • 95 | |
| Group ownership • 95 | |
| File size • 96 | |
| Modification time • 96 | |
| Filename • 96 | |
| Ownership | 96 |
| Permissions | 97 |
| Numeric/octal • 98 | |
| Common permissions • 99 | |
| Changing ownership (chown) and permissions (chmod) | 99 |
| Chown • 99 | |
| <i>Change owner • 99</i> | |
| <i>Change owner and group • 100</i> | |
| <i>Recursively change owner and group • 100</i> | |
| Chmod • 100 | |
| <i>Using a reference • 101</i> | |
| Conclusion • 101 | |
| Chapter 9: Managing Installed Software | 103 |
| Working with software packages | 104 |
| Update your local cache of repository state • 105 | |
| Search for a package • 105 | |

| | |
|--|------------|
| Install a package • 106 | |
| Upgrade all packages that have available updates • 106 | |
| Remove a package (and any dependencies, provided other packages don't need them) • 106 | |
| Query installed packages • 107 | |
| Caution required – curl bash | 107 |
| Compiling third-party software from source | 108 |
| Example: compiling and installing htop • 110 | |
| <i>Install prerequisites</i> • 110 | |
| <i>Download, verify, and unarchive the source code</i> • 110 | |
| <i>Configure and compile htop</i> • 111 | |
| Conclusion | 112 |
| | |
| Chapter 10: Configuring Software | 115 |
| <hr/> | |
| Configuration hierarchy | 116 |
| Command-line arguments | 118 |
| Environment variables | 119 |
| Configuration files | 121 |
| System-level configuration in /etc/ • 121 | |
| User-level configuration in ~/.config • 121 | |
| systemd units | 122 |
| Create your own service • 122 | |
| Quick note: configuration in Docker | 124 |
| Conclusion | 125 |
| | |
| Chapter 11: Pipes and Redirection | 127 |
| <hr/> | |
| File descriptors | 128 |
| What do these file descriptors reference? • 129 | |
| Input and output redirection (or, playing with file descriptors for fun and profit) | 129 |
| Input redirection: < • 129 | |
| Output redirection: > • 130 | |
| <i>Use >> to append output without overwriting</i> • 131 | |

| | |
|---|------------|
| Error redirection with 2> • 132 | |
| Connecting commands together with pipes () | 133 |
| Multi-pipe commands • 133 | |
| <i>Reading (and building) complex multi-pipe commands • 134</i> | |
| The CLI tools you need to know | 134 |
| cut • 134 | |
| sort • 135 | |
| uniq • 136 | |
| <i>Counting • 136</i> | |
| wc • 137 | |
| head • 138 | |
| tail • 138 | |
| tee • 138 | |
| awk • 139 | |
| sed • 139 | |
| Practical pipe patterns | 140 |
| “Top X”, with count • 140 | |
| curl bash • 140 | |
| <i>Security considerations for curl sudo bash • 141</i> | |
| Filtering and searching with grep • 142 | |
| grep and tail for log monitoring • 143 | |
| find and xargs for bulk file operations • 143 | |
| sort, uniq, and reverse numerical sort for data analysis • 144 | |
| awk and sort for reformatting data and field-based processing • 145 | |
| sed and tee for editing and backup • 145 | |
| <i>ps, grep, awk, xargs, and kill for process management • 146</i> | |
| <i>tar and gzip for backup and compression • 146</i> | |
| Advanced: inspecting file descriptors | 147 |
| Conclusion | 149 |

| | |
|--|------------|
| Chapter 12: Automating Tasks with Shell Scripts | 151 |
| Why you need Bash scripting basics | 152 |
| Basics | 152 |
| Variables • 152 | |
| <i>Setting</i> • 152 | |
| Getting • 153 | |
| Bash versus other shells | 153 |
| Shebangs and executable text files, aka scripts | 154 |
| Common Bash settings (options/arguments) • 154 | |
| /usr/bin/env • 155 | |
| Special characters and escaping • 156 | |
| Command substitution • 157 | |
| Testing | 157 |
| Testing operators • 157 | |
| [[file and string testing]] • 158 | |
| <i>Useful operators for string testing</i> • 158 | |
| <i>Useful operators for file testing</i> • 158 | |
| ((arithmetic testing)) • 159 | |
| Conditionals: if/then/else | 160 |
| <i>ifelse</i> • 160 | |
| Loops | 160 |
| C-style loops • 160 | |
| for...in • 161 | |
| While • 161 | |
| Variable exporting • 162 | |
| Functions | 162 |
| Prefer local variables • 163 | |



Lituz.com

**To'liq qismini
Shu tugmani
bosish orqali
sotib oling!**