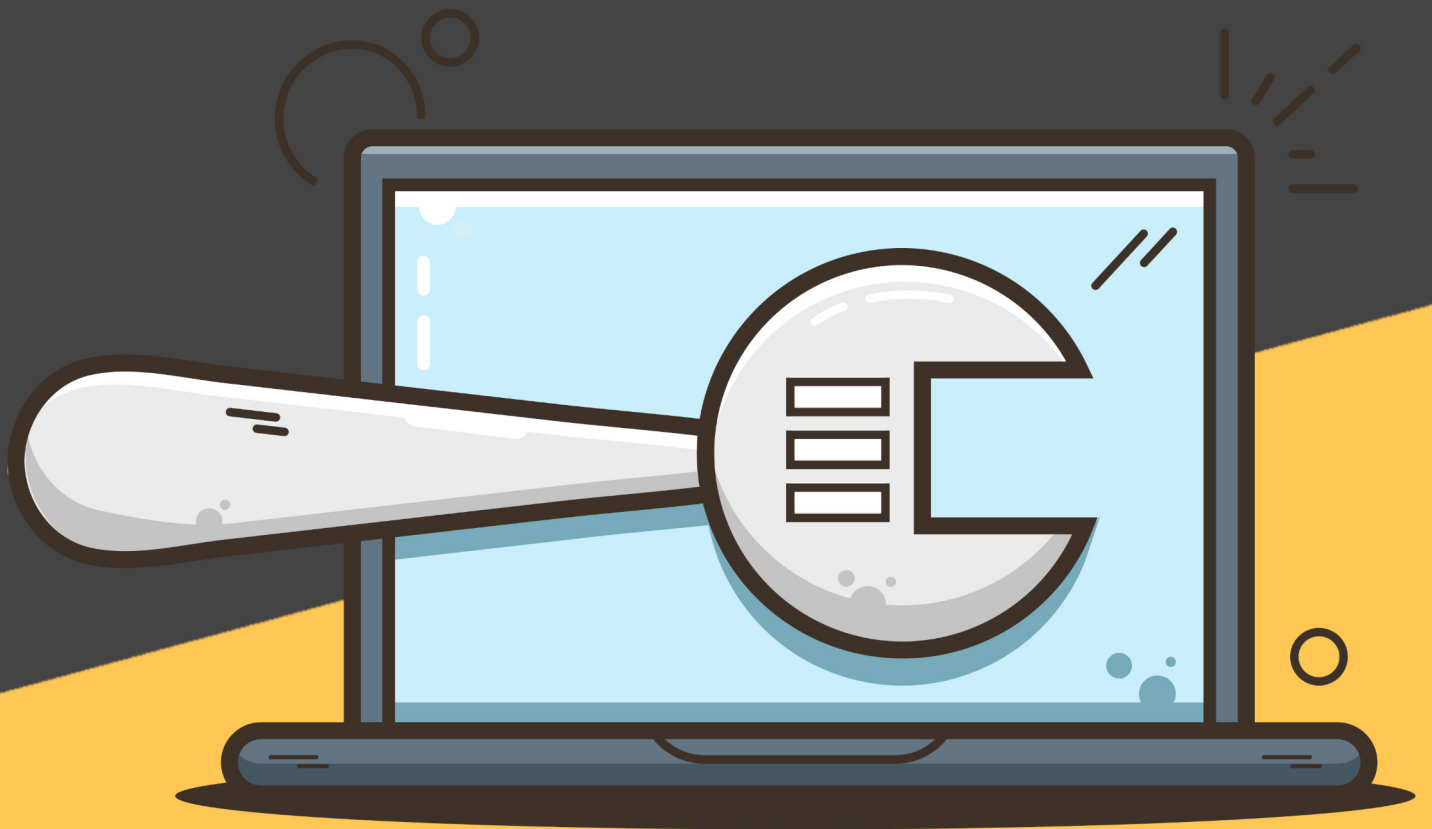


# Building your Mouseless Development Environment



Matthieu Cneude

# Contents

<b>Introduction</b>	<b>11</b>
<b>Acknowledgments</b>	<b>12</b>
<b>Welcome, Mouseless Developers</b>	<b>13</b>
Who Should Read This Book?	14
What Is a Mouseless Development Environment?	14
What Do You Need to Follow Along?	15
Creating Your Own Cheatsheets	15
Experimenting Is Key	16
Styling Conventions	16
Choose Your Tools	16
In a Nutshell	16
 <b>Part I – Linux</b>	 <b>18</b>
<b>A General Linux Overview</b>	<b>19</b>
Diving Inside Linux	19
The Linux Filesystem	20
Linux Distributions	21
Packages and Repositories	22
Why Arch Linux?	22
The Glory of Rolling Distributions	22
The Arch Linux Community	23
Official Repositories and the Arch User Repositories (AUR)	23
The Fabulous Manual	23
Troubleshooting	24
General Recommendations	24
Using VMWare Software	24
In a Nutshell	24
Going Deeper	24
 <b>The Power Is In Your Fingers</b>	 <b>26</b>
Efficient Typing: The Two Rules	26
The First Week	27
The Second Week	27
Speed and Accuracy	28
Keyboard Layout	28
In A Nutshell	28
Going Deeper	28
 <b>Preparing Your System for Arch Linux</b>	 <b>29</b>
Prerequisites	29
Burning the Arch Linux ISO	29
Configuring the Arch Linux Live System	30
Keyboard Layout	31

Connecting to the Internet . . . . .	32
System Clock . . . . .	33
BIOS or UEFI? . . . . .	33
Partitioning the Hard Disk . . . . .	34
Wiping Your Hard Disk . . . . .	35
Using fdisk . . . . .	36
Boot Partition . . . . .	36
Root and Swap Partition . . . . .	38
Formatting the Partitions . . . . .	39
The UEFI Boot partition . . . . .	39
Root and Swap Filesystems . . . . .	39
Mounting the Filesystems . . . . .	40
Mounting the Root Partition . . . . .	40
UEFI and The Boot Partition . . . . .	40
Continuing The Installation . . . . .	41
Troubleshooting Your Internet Access . . . . .	41
In a Nutshell . . . . .	41
Going Deeper . . . . .	42
<b>Installing Arch Linux</b> . . . . .	<b>43</b>
Installing the Base Packages . . . . .	43
Mounting Partitions Automatically . . . . .	44
Changing The Root Directory . . . . .	44
The Root User's Password . . . . .	45
Through Time and Space: Configuring the Timezone . . . . .	45
Choose Your Locale . . . . .	46
Naming Your New World . . . . .	47
GRUB, The Linux Bootloader . . . . .	48
Bootloader With a UEFI . . . . .	48
Bootloader With a BIOS . . . . .	48
Enabling the Network Manager . . . . .	49
Diving in the Shell . . . . .	49
Command-Line 101 . . . . .	49
A Good Complement to the Manual . . . . .	50
Input Output Redirections . . . . .	50
Pipes . . . . .	51
Input and Arguments . . . . .	51
Rebooting The System . . . . .	52
Troubleshooting Pacman . . . . .	53
In a Nutshell . . . . .	53
Going Deeper . . . . .	53
<b>Welcome To Linux</b> . . . . .	<b>54</b>
Connecting to The Internet, Third Round . . . . .	54
Using a Cable . . . . .	54
Using the Wi-Fi . . . . .	55
Installing the Manuals . . . . .	55
Processes . . . . .	55
The Init Process: systemd . . . . .	56
Logging the Init Process With journald . . . . .	58
Processes As Files . . . . .	59
Environment Variables . . . . .	59
Creating A New User . . . . .	59
The Default Text Editors . . . . .	61
The Return of the Environment Variable . . . . .	62
Trying visudo Again . . . . .	62
First Steps In Neovim . . . . .	63
Neovim Modes . . . . .	63

Editing with Neovim . . . . .	64
Using Sudo . . . . .	64
Best Practices For Linux Shells . . . . .	65
Strings . . . . .	65
Globbing . . . . .	65
Strings And Quotes . . . . .	65
In a Nutshell . . . . .	66
Going Deeper . . . . .	67
Links . . . . .	67
Manual . . . . .	67
<b>Graphical Interface</b>	<b>68</b>
The X Window System . . . . .	68
The X What? . . . . .	68
Installing X . . . . .	69
URxvt, Our Terminal Emulator . . . . .	69
Video Terminals and TTYS . . . . .	70
Installing i3 Window Manager . . . . .	71
Launching X . . . . .	71
Keyboard Layout for X . . . . .	72
Installing Fonts . . . . .	72
Launching i3 . . . . .	72
Graphic Cards and Video Drivers . . . . .	72
In a Nutshell . . . . .	72
Going Deeper . . . . .	73
 <b>Part II - Mouseless Tools</b>	 <b>74</b>
<b>First Steps in i3 Window Manager</b>	<b>75</b>
Screen Resolution . . . . .	75
The Basics of i3 . . . . .	76
i3bar and i3status . . . . .	76
Installing Your Favorite Browser . . . . .	77
Program Launcher . . . . .	77
Copy and Paste . . . . .	77
Pasting From a PDF . . . . .	78
The Book Companion . . . . .	78
Troubleshooting . . . . .	79
In a Nutshell . . . . .	79
Going Deeper . . . . .	79
 <b>Configuring URxvt</b>	 <b>80</b>
The Colors of the Terminal . . . . .	80
Making URxvt Prettier . . . . .	81
Configuring URxvt . . . . .	82
Character Fonts . . . . .	83
Window Default . . . . .	83
URxvt Daemon . . . . .	83
In a Nutshell . . . . .	84
Going Deeper . . . . .	84
 <b>The Basics of Neovim</b>	 <b>85</b>
Neovim Modes . . . . .	85
Configuring Neovim . . . . .	86
Forget the Arrow Keys . . . . .	86
Switching To Insert Mode . . . . .	87
Undo And Redo . . . . .	87
Motions . . . . .	87

Horizontal Motions . . . . .	88
Vertical Motions . . . . .	88
Scrolling commands . . . . .	88
The Language of Neovim . . . . .	88
Operators . . . . .	89
Text Objects . . . . .	89
A Basic Configuration . . . . .	89
Neovim's Options . . . . .	89
Swap Files . . . . .	90
Undo Tree . . . . .	90
General Options . . . . .	91
In a Nutshell . . . . .	91
Going Deeper . . . . .	91
<b>Pacman and Yay</b> . . . . .	<b>92</b>
Operations and Options . . . . .	92
Official Repositories . . . . .	92
Updating Your System . . . . .	93
Removing Packages . . . . .	93
Searching Packages . . . . .	94
Pacman's Cache . . . . .	94
The Arch User Repository (AUR) . . . . .	94
The Package Manager Yay . . . . .	95
Clearing Yay Cache . . . . .	96
Installing Packages With Yay . . . . .	96
Adding Tabs for URxvt . . . . .	96
Neovim Language Extensions . . . . .	97
Pacman Troubleshooting . . . . .	97
Pacman Configuration . . . . .	98
In a Nutshell . . . . .	98
Going Deeper . . . . .	98
<b>Dotfiles</b> . . . . .	<b>99</b>
Hard and Symbolic Links . . . . .	99
Hard Links . . . . .	99
Symbolic Links . . . . .	100
The Structure of the Dotfiles Project . . . . .	100
Our First Bash Script . . . . .	101
Running A Shell Script . . . . .	102
Changing Directories And Files Permissions . . . . .	103
Files and Directories Ownership . . . . .	104
Making Our Dotfiles Public . . . . .	105
The SSH Protocol . . . . .	105
Adding Your SSH Public Key to GitHub . . . . .	106
In a Nutshell . . . . .	107
Going Deeper . . . . .	107
General Articles . . . . .	107
Getting Inspired . . . . .	108
<b>i3: A Deeper Dive</b> . . . . .	<b>109</b>
How To Use i3? . . . . .	109
General Organization . . . . .	109
The Default Shortcuts . . . . .	110
Configuring i3 . . . . .	111
Configuration Files . . . . .	111
Default Configuration . . . . .	111
Program Launcher . . . . .	112
Moving Windows and Changing Focus . . . . .	112
Split containers . . . . .	113

Workspaces . . . . .	113
Resizing Windows . . . . .	115
Locking Your Screen . . . . .	116
Lock, Shutdown, and Reboot Menu . . . . .	117
Wallpaper . . . . .	118
Floating Windows . . . . .	118
Colors and Style . . . . .	119
Scratchpad . . . . .	120
The i3 Status Bar . . . . .	121
Managing Your Screen . . . . .	122
Updating Our Dotfiles . . . . .	123
In a Nutshell . . . . .	123
Going Deeper . . . . .	124
<b>The Z-Shell (Zsh)</b>	<b>125</b>
Framework Or No Framework? . . . . .	125
Zsh Config Files . . . . .	126
Basic Zsh Configuration . . . . .	127
Environment Variables . . . . .	127
Aliases . . . . .	128
Options . . . . .	129
Zsh Completion System . . . . .	129
Pimp My Zsh Prompt . . . . .	130
Zsh Directory Stack . . . . .	131
Zsh, Your New Best Friend . . . . .	132
Zsh With Vim Flavors . . . . .	132
Enabling the Vi Mode . . . . .	132
Changing Cursor . . . . .	132
Vim Mapping For Completion . . . . .	132
Editing Commands In Neovim . . . . .	133
Zsh Plugins . . . . .	133
Zsh Additional Completion . . . . .	133
Zsh Syntax Highlighting . . . . .	134
Jumping To A Parent Directory . . . . .	134
Custom Scripts . . . . .	134
The Fuzzy Finder fzf . . . . .	135
Automatically Starting i3 . . . . .	136
In a Nutshell . . . . .	136
Going Deeper . . . . .	137
Zsh Documentation . . . . .	137
Author's dotfiles . . . . .	137
<b>Improving Your Mouseless Development Environment</b>	<b>138</b>
Improving the Dotfiles Install Script . . . . .	138
Adding And Configuring Fonts . . . . .	140
Installing Fonts . . . . .	140
Changing URxvt's Fonts . . . . .	141
Changing i3's Fonts . . . . .	141
Desktop Notifications With Dunst . . . . .	142
Automatically Mounting Devices . . . . .	143
Visual Configuration . . . . .	143
Remapping Your Caps Lock . . . . .	144
A New Escape Key . . . . .	144
Two Keys in One . . . . .	144
The Problem of the External Keyboard . . . . .	145
Git diff . . . . .	145
In a Nutshell . . . . .	145
Going Deeper . . . . .	146

<b>Neovim: A Deeper Dive</b>	<b>147</b>
Neovim Spatial Organization	147
Buffers	147
Windows	149
Tabs	150
Argument List (arglist)	150
Mapping Keystrokes	151
Jump! Jump! Jump!	153
Jump List	153
Change List	153
Method Jump	153
Repeating Keystrokes	153
Single Repeat	153
Complex Repeat: The Macro	154
The Command Window	154
Revisiting the Undo Tree	154
Plugins	155
Plugin Manager	155
Closing Buffers Without Closing Windows	157
Managing Windows Easily	157
Navigating Through The Buffer List	157
Manipulating the Undo Tree	157
Automatically Installing the Plugin Manager	157
In a Nutshell	158
Going Deeper	158
<b>The Terminal Multiplexer tmux</b>	<b>159</b>
What's tmux?	159
Why use Tmux?	160
Background Operations	160
More Terminals! Everywhere!	160
Saving tmux Sessions	160
Remote Pair Programming	160
How to use tmux?	161
General Organization	161
tmux Workflow	162
Managing tmux Sessions	162
Configuring tmux	162
The Essentials of tmux	163
Increasing The Maximum Output Lines	165
Managing Windows	165
Design	167
Plugins	168
The tmux Plugins Manager	168
Fuzzy Search And Copy with fzf and Extrakto	169
Creating tmux Sessions Automatically	169
i3 Scratchpad Running tmux	170
Choosing Your tmux Session With fzf	171
In a nutshell	172
Going Deeper	172
<b>Neovim Plugins</b>	<b>173</b>
The Language Server Protocol	173
The Plugin coc.nvim	173
Extensions to coc.nvim	174
tmux completion	174
Fuzzy Finder in Neovim With fzf	174
Navigating files	175

Linter . . . . .	175
Surrounding . . . . .	175
Navigating in Open Buffers . . . . .	176
Text Objects . . . . .	176
Register History . . . . .	176
Snippets . . . . .	176
Search And Replace . . . . .	177
Status Bar . . . . .	177
Color Scheme . . . . .	177
Manual Pages In Neovim . . . . .	177
The Undo-tree . . . . .	178
Tmux . . . . .	178
Startup . . . . .	178
Git . . . . .	179
Syntax Highlighting . . . . .	179
Misc . . . . .	179
In a Nutshell . . . . .	179
Going Deeper . . . . .	180
<b>Mouseless Browsers</b> . . . . .	<b>181</b>
Lynx . . . . .	181
Help . . . . .	182
Navigation . . . . .	182
History . . . . .	182
Page Data . . . . .	182
Bookmarks . . . . .	182
Downloading . . . . .	182
Options . . . . .	183
Quitting Lynx . . . . .	183
Qutebrowser . . . . .	183
Basics . . . . .	183
Navigation . . . . .	183
Tabs . . . . .	184
Modes . . . . .	184
Browser Plugins . . . . .	184
Firefox . . . . .	184
Chrome . . . . .	184
In a Nutshell . . . . .	185
Going Deeper . . . . .	185
 <b>Part III - Building Your System Installer</b> . . . . .	 <b>186</b>
<b>The Installer</b> . . . . .	<b>187</b>
The Project . . . . .	187
The User Interface . . . . .	188
Preliminary Configuration . . . . .	189
Is It the Good Time? . . . . .	189
Return Code and Operators . . . . .	189
Output Redirection . . . . .	191
UEFI or BIOS, That Is The Question . . . . .	192
Choosing The Hard Disk . . . . .	193
Bash arrays . . . . .	193
Pipes . . . . .	195
Awk . . . . .	195
Grep . . . . .	195
Putting Everything Together . . . . .	196
In a Nutshell . . . . .	197



Going Deeper . . . . .	197
<b>Partitioning and Installing Arch Linux</b>	<b>198</b>
Size of the Partitions . . . . .	198
Erasing the Hard Disk . . . . .	199
Creating Partitions . . . . .	200
Boot Partition With BIOS or UEFI . . . . .	200
Automating fdisk . . . . .	200
Formatting partitions . . . . .	202
General Case . . . . .	202
Special Case . . . . .	202
Generating fstab And Installing Arch Linux . . . . .	202
The Adventure Continue! . . . . .	203
The End of the Installer . . . . .	203
In a Nutshell . . . . .	204
Going Deeper . . . . .	204
<b>Creating Users and Passwords</b>	<b>205</b>
Getting Back the Block Devices and the Boot Mode . . . . .	205
Naming Your Newborn System . . . . .	206
Keyboard Layout . . . . .	206
Installing The Bootloader GRUB . . . . .	206
Clock and Timezone . . . . .	206
Configuring the Locales . . . . .	207
Root Password and User Creation . . . . .	207
Arch Linux Is Now Fully Configured . . . . .	210
In a Nutshell . . . . .	211
Going Deeper . . . . .	211
<b>Installing The Tools</b>	<b>212</b>
The List of Applications . . . . .	212
Groups of Applications . . . . .	214
Parsing the CSV . . . . .	215
Updating the System . . . . .	216
Installing the Packages . . . . .	217
Permission For Power: sudo . . . . .	219
Invoking The Last Installer Script . . . . .	219
In a Nutshell . . . . .	219
Going Deeper . . . . .	220
<b>The User Installer</b>	<b>221</b>
Usual Linux Directories . . . . .	221
Keyboard Layout . . . . .	221
Installing Packages From the AUR . . . . .	222
Installing the Dotfiles . . . . .	223
The One Command To Invoke The Installer . . . . .	224
In a Nutshell . . . . .	224
Going Deeper . . . . .	225
 <b>Part IV - Customizing Your Environment</b>	 <b>226</b>
<b>Alternative Tools</b>	<b>227</b>
Linux Distribution . . . . .	227
An Alternative to X . . . . .	228
Tiling Window Manager . . . . .	228
Shells . . . . .	228
Terminal Emulators . . . . .	229
Editors . . . . .	229

Non-Modal Editors . . . . .	229
Modal Editors . . . . .	230
Terminal Multiplexer . . . . .	230
In a Nutshell . . . . .	230
<b>Inspiring Communities</b>	<b>232</b>
Going Deeper . . . . .	232

# **Part I - Linux**

# Preparing Your System for Arch Linux

It's time to get our hands dirty! In this chapter, we'll prepare our system for Arch Linux for us to feel at home. More precisely, we'll answer these questions:

- How to burn an Arch Linux ISO on a USB key?
- How to configure the Arch Linux live system?
- How to partition a hard disk using `fdisk`?
- How to create and mount new filesystems?

This is the very beginning of the journey, so I'll try to explain most things in detail. Ready for the challenge? Let's go!

## Prerequisites

To install our new Mouseless Development Environment, we'll need the following:

1. A computer with a 64-bit CPU (not older than 10 years).
2. Internet access via Ethernet *and* Wifi is recommended, in case **one or the other doesn't work** on the Arch Linux live system.
3. An empty hard disk, or a hard disk you'll wipe out (at least 20 GB to feel comfy).
4. A USB key (at least 1 GB).
5. A second computer, a tablet, or any other device allowing you to read this book while installing everything.
6. Internet access on your second device is highly recommended, in case you have a problem in the first chapters.

As we mentioned already, the computer can be a virtual machine or a physical one.

## Burning the Arch Linux ISO

If you wonder what the heck an ISO is, it's a file representing a physical disk. We can burn ISO files on a real disk, which means copying everything from the ISO to the disk.

To install Arch Linux, we need to start (or *boot*) our computer using the Arch Linux live system, and install the OS from there. Here's how to do that:

1. Download the [Arch Linux ISO](#). Find your country, click on the first link, and download `archlinux -< the_last_release_date >-x86_64.iso`.

2. Verify that the ISO is the official one, and has not been modified by horrible hackers. You need to generate the MD5 hash of the ISO and compare it with the one provided on the download page.
  - On Windows, you can use the command `CertUtil -hashfile < path_to_ISO_file > MD5`.
  - On Linux, you can use the command `md5sum < path_to_ISO_file >` to get the MD5 of the file.
3. Burn the ISO on a USB key. You can use:
  - On Windows: [rufus](#).
  - On Ubuntu: Startup Disk Creator.
  - On another Linux distro: [UNetbootin](#).
4. Change the boot order to read USB devices before your hard disk. You can do that by using your BIOS interface.
5. Plug your USB key and start your computer.

For the fourth point, you can normally access the BIOS interface by hitting a specific key when your computer starts. If you're lucky, it might be displayed at startup long enough for you to see it. It's often `ESC`, `DEL` or `F10`. Every computer is different on this point, so I can't really help you more.

When you found a way to access your BIOS' interface, search for any option concerning the boot. You can normally change the boot order of the devices: the goal is to put your USB device before your hard disk. When you're done with that, there will be some options to save your changes and restart your computer.

If you did everything correctly, your computer will boot from your USB key, and you'll see a menu inviting you to install Arch Linux. Let's do it!

## Configuring the Arch Linux Live System

Your screen will then spit out many green "OK" messages telling you what systemd is starting. We'll talk about systemd later in this book. Then, you'll end up in a shell prompt. It will look like this:

```
root@archiso ~ #
```

A prompt is a set of characters indicating that you can execute commands. Here, it indicates what user you are ( `root` ) and the hostname `archiso`. The hostname is the name of your system as it will appear on a network.

Welcome to the Arch Linux live system! What's a live system? For now, nothing has been installed on your hard disk; instead, the system you have access to is running from your computer's memory (RAM). Many distros provide a live system for you to test it even before installing anything.

It also means that everything you're doing in the live system won't survive if you shut down or restart your computer, except if you deliberately write something on your hard disk. For example, any program you'll install on the live system won't be installed on your hard disk.

The shell we're using now is called the "Z shell", or Zsh. It's similar to the most famous shell, Bash. We'll dig more into Zsh later in this book, too. Here are some functionalities you can use with the shell:

- You can enter any command in a shell prompt and execute it by hitting `ENTER` .
- The shell can save the command history:
  - You can quickly access executed commands with the keys `ARROW UP` and `ARROW DOWN` .
  - You can search through the history with `CTRL+r` .
- You can use Zsh auto completion by hitting `TAB` .
- To stop a running command, you can use the keystroke `CTRL+c` . Be careful with it: patience is sometimes safer. Think about what your command is doing before interrupting it, and consider the price you might have to pay by stopping it before it's done.

You are now the *root user*. You can create different users on a Linux system, but the root user is special. It has almost all power a user can have. Don't confuse the root user and the root directory `/` we saw in the previous chapter. A filesystem and a bunch of users are different things, even if they have both root as their name.

By being the root user, you're in fact the demigod (or demigoddess!) of the "archiso" live system. Why only demi? Because a user can't control directly the kernel. The kernel is the real master around here.

## Keyboard Layout

By default, you'll end up with the American keyboard layout on the Arch Linux live system. If you're not comfortable with it, you can change it. Here are the two commands you can use to do so:

1. `ls /usr/share/kbd/keymaps/**/*.map.gz` - List all layout available.
2. `ls /usr/share/kbd/keymaps/**/*.map.gz | grep "< your_language_code >"` - Filter all layout with grep.

For example, if I want to use a French keyboard layout, I can do this:

```
ls /usr/share/kbd/keymaps/**/*.map.gz | grep "fr"
```

The result will look like the following:

```
/usr/share/kbd/keymaps/i386/azerty/fr-latin1.map.gz
/usr/share/kbd/keymaps/i386/azerty/fr-latin9.map.gz
/usr/share/kbd/keymaps/i386/azerty/fr.map.gz
/usr/share/kbd/keymaps/i386/azerty/fr-pc.map.gz
/usr/share/kbd/keymaps/i386/bepo/fr-bepo-latin9.map.gz
/usr/share/kbd/keymaps/i386/bepo/fr-bepo.map.gz
/usr/share/kbd/keymaps/i386/dvorak/dvorak-ca-fr.map.gz
/usr/share/kbd/keymaps/i386/dvorak/dvorak-fr.map.gz
/usr/share/kbd/keymaps/i386/qwertz/fr_CH-latin1.map.gz
/usr/share/kbd/keymaps/i386/qwertz/fr_CH.map.gz
/usr/share/kbd/keymaps/mac/all/mac-fr_CH-latin1.map.gz
/usr/share/kbd/keymaps/mac/all/mac-fr.map.gz
/usr/share/kbd/keymaps/sun/sunt5-fr-latin1.map.gz
```

You can then choose your layout by using the filename without the file extension `map.gz` :

```
loadkeys <filename>
```

For example, for my French keyboard layout, I would run the command `loadkeys fr-latin1`. Write this keymap somewhere, you'll need it again in the next chapter.

## Connecting to the Internet

What would we be without Internet? Nothing more than unconscious amoebas lost in a chain of misconceptions we call reality. Without Internet, I wouldn't even be able to write the previous sentence!

To connect to this infinite amount of knowledge and cat pictures, we need some network device. The command `ip link` will show you the truth about them. Let's try to run it in the terminal:

```
ip link
```

Something like this will be output:

```
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode
  DEFAULT group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: enp0s25: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc fq_codel
  state DOWN mode DEFAULT group default qlen 1000
    link/ether f0:de:f1:84:f3:a6 brd ff:ff:ff:ff:ff:ff
3: wlp3s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP
  mode DORMANT group default qlen 1000
    link/ether 08:11:96:0a:12:b4 brd ff:ff:ff:ff:ff:ff
```

Don't worry about the first result `lo`. The second one `enp0s25` is my network device for Ethernet cable; it often begins with an 'e'. The third one `wlp3s0` is for the Wi-Fi; this one often begin with a "w". We don't have the same computer, so it's more than likely we won't have the same network devices either.

If you want to connect to Internet using a cable, simply plug it. For the Wi-Fi, you can run the following:

```
iwctl
```

A new shiny prompt will appear. It works similarly to the shell prompt you had before. Here are the commands you can use:

1. `device list` - List all your Wi-Fi devices.
2. `station <device> get-networks` - Replace `<device>` with the name of one of yours, to get a list of networks you can connect to with this particular device.
3. `station <device> connect "<network>"` - Replace `<device>` and `<network>` by the device you want to use with the network you want to connect to.
4. If a password is required, enter it.
5. `station <device> show` - Show you if you're indeed connected.
6. `exit` - Exit `iwctl`.

If it failed, you might get the friendly message `Operation failed`. Don't worry and try again: even demigods fail, sometimes. Demigoddesses too. Another reason why we're only "demi" here, and not the full version.

Let's now verify that you're really connected. Millions of people use this command all around the world as we speak, to verify their Internet connection:

```
ping -c 5 thevaluable.dev
```

It will send 5 requests to the address `thevaluable.dev` and display the following:

```
64 bytes from v22017064675050008.supersrv.de (185.183.156.38): icmp_seq=1
ttl=58 time=32.6 ms
64 bytes from v22017064675050008.supersrv.de (185.183.156.38): icmp_seq=2
ttl=58 time=32.6 ms
64 bytes from v22017064675050008.supersrv.de (185.183.156.38): icmp_seq=3
ttl=58 time=32.1 ms
64 bytes from v22017064675050008.supersrv.de (185.183.156.38): icmp_seq=4
ttl=58 time=32.9 ms
64 bytes from v22017064675050008.supersrv.de (185.183.156.38): icmp_seq=5
ttl=58 time=32.2 ms
```

If you've got something similar, you're connected to the infinite Internet! Lucky you.

If it doesn't work, try:

```
ping -c 5 185.183.156.38
```

If this last command works, this means that you can't resolve addresses like `thevaluable.dev` to its IP address `185.183.156.38`. In that case, try to run a DHCP client:

```
dhcpcd &
```

What's `thevaluable.dev`? It's my blog, to show you how to do some subtle product placement in your own book.

## System Clock

Let's now use one of the oldest protocols on the Internet. Anxious? Don't worry, Internet is the only stuff which never breaks in the computing world. Not like enterprise Java code.

Run this command:

```
timedatectl set-ntp true
```

We synchronized our system clock with the Network Time Protocol. That's all. Next!

## BIOS or UEFI?

You remember the interface you used to change the boot order of your devices, to boot your USB key before another device?

This was the interface of some *firmware*, a piece of software tightly linked to a piece of hardware. In this precise case, this firmware is called a *BIOS* (for **B**asic **I**ntput **O**utput **S**ystem) and it's directly embedded in a chip in your motherboard. The BIOS is the first piece of software running when you're booting (starting) your computer.



This BIOS verifies that your hardware works properly. It also makes available an interface for you to configure some basics, like the order of the booting devices. It will also run the *bootloader*, another program which is used to boot an operating system, like Arch Linux.

That said, the BIOS is now considered deprecated for a better alternative, called *UEFI* (for **U**nified **E**xtensible **F**irmware **I**nterface). BIOS and UEFI are two different types of firmware, but confusion arises when the UEFI pretends to be a BIOS, sometimes using the term “BIOS” on its own interface! It’s often an ironic attempt not to confuse people who are used to having a BIOS.

Long story short, because the firmware of your motherboard starts the bootloader of Arch Linux, we need to know if you have a BIOS or a UEFI on your computer. To verify that, simply run in the shell:

```
ls /sys/firmware/efi/efivars
```

If you see the error `no such file or directory`, this means that your computer has a BIOS. If it outputs a bunch of files, you have a UEFI. You need to remember what kind of firmware you have to install Arch Linux properly. My advice: write it somewhere, on a piece of paper for example. We’ll need this information later to create our boot partition and install the bootloader itself.

We’ve just learned the first piece of software which runs on our computer. You know what we should do? Create a diagram we’ll complete as we go, to describe the boot process of a Linux-based system:

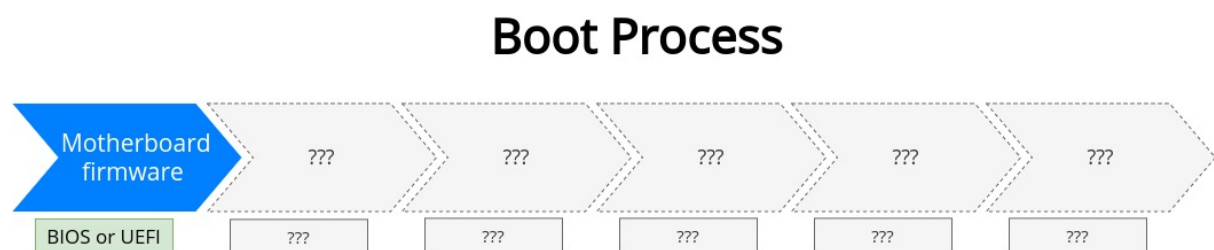


Figure 3: The boot process: BIOS and UEFI

This process is followed by all Linux distros out there. The green box under the arrow indicates the program (or group of programs) used during this process, sometimes specific to Arch Linux.

## Partitioning the Hard Disk

Enough explanations. Let’s do some serious stuff now. First, let’s run the following in the shell:

```
lsblk
```

This command will display your *block devices*. A block device is a file which represents a physical device. Confused? How can a device, such as a hard disk, be represented as a file?

A Linux-based system tries hard to have a consistent way for you, the applications running on your system, the kernel, and the hardware, to interact with each other. In other words, Linux

## **Part II - Mouseless Tools**

# i3: A Deeper Dive

We saw very quickly in the chapter [First Steps In Your Mouseless Development Environment](#) how to use i3. In this chapter, we'll dive deeper into i3 configuration to really understand what we can do with this fantastic windows tiling manager. I hope that at the end of this chapter you'll be able to understand i3's excellent documentation and what to modify in your configuration to answer your own needs.

A tiling window manager is often lighter than a full-blown desktop environment. It provides powerful mouseless functionalities to manage the windows displaying your favorite applications: the terminal, your favorite cat pictures viewer, and whatnot.

Do you think resizing windows manually is an activity providing a lot of value? I don't think so. As we saw already, i3 helps you manage your windows for 99% of your use cases, if you're not in dear needs of floating windows. This percentage has no scientific foundation whatsoever, but it doesn't stop me to believe in it. We have to believe my friends!

In this chapter, we'll see:

- The general organization of i3.
- How to configure i3.
- What are workspaces and how to configure them.
- How to configure a locking screen.
- How to set up a wallpaper.
- How to configure floating windows.
- How to configure the colors and styles of i3's graphical interface.
- How to configure scratchpads.
- What's the i3 bar and how to configure it.
- How to create a menu with dmenu to manage your screens (if you have more than one).
- What do we need to update in our dotfiles project.

We'll cover a lot here: take a solid beverage, a nice snack, and let's go exploring the depth of i3 together.

## How To Use i3?

### General Organization

Like tmux we'll see in a later chapter, i3 stores its information in a tree data structure. Let's see what each node can represent.

## Workspaces

At the top of our tree you'll find the *workspaces*. You can think of them as [virtual desktops](#). A workspace can contain *containers*.

## Containers

A container can contain one or multiple *windows*. These windows will be positioned depending on the container's layout. There are three different layouts possible:

- **Split** - Each window shares the container space and are split horizontally (*splith*) or vertically (*splitv*). This is the default layout.
- **Stacked** - The focused window is visible on top and the other ones are stacked below. You can change the window's focus via keystrokes easily. You can see the windows' headers at the top of the container itself.
- **Tabbed** - This layout is similar to the stacked layout, except that the windows' headers are vertically split, not horizontally.

Note that a container can contain other containers too.

To understand this tree of workspaces, container, and windows, here's a possible representation:

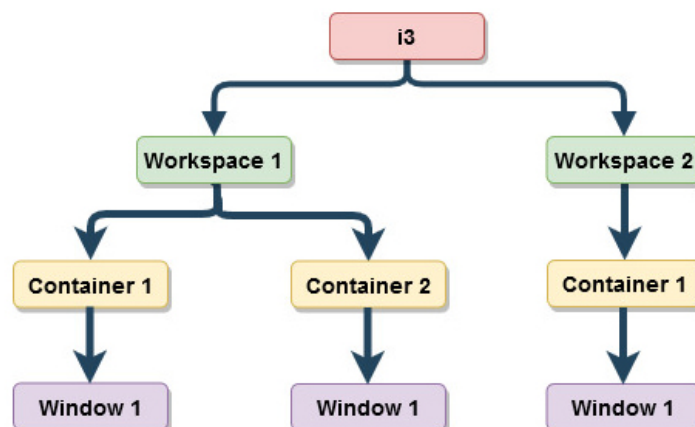


Figure 8: i3 general organization

Of course, you can have more workspace, containers, and windows, if you want to. This is just an example to show the hierarchy of the different elements.

## The Default Shortcuts

We've seen that i3 uses a special key for (almost) every keystroke. This key is called a *modifier*; by default, it's the [WIN](#) key (or [CMD](#) key, if you like Apple keyboards). The variable `$mod` in the configuration is there to set this modifier key. I'll use this variable in this chapter to express the modifier key for the different keystrokes, and you should always use it in your own configuration.

Enough mumbling. Let's practice!

As we saw already, you can move the windows around and see how they are resized automatically. To do so, try to open multiple windows and hit `$mod + SHIFT + ARROW KEY` to see what happens.

We can try to change the layout of the container, too, by using the following keystrokes:

- `$mod + e` - Switch to *split* layout ("splith" or "splitv" depending on your screen).
- `$mod + s` - Switch to *stacked* layout.
- `$mod + w` - Switch to *tabbed* layout.

You can try to create and move windows using each layout to see the differences.

You also need to know the following keystrokes:

- `$mod + SHIFT + c` - Reload i3's configuration. You need to use it each time you modify your configuration file, to apply the changes to the current i3 session.
- `$mod + SHIFT + e` - Logout and quit i3. We will modify that later.

## Configuring i3

### Configuration Files

Your config file ( `~/.config/i3/config` ) should already exist. This is not the only configuration file available for i3: different ones can be loaded in a specific order. Here's the complete list; the configuration files on top can overwrite every other configuration files below.

1. `~/.config/i3/config` (or `$XDG_CONFIG_HOME /i3/config` if `$XDG_CONFIG_HOME` is set).
2. `~/.i3/config`.
3. `/etc/xdg/i3/config` (or `$XDG_CONFIG_DIRS /i3/config` if `$XDG_CONFIG_DIRS` is set).
4. `/etc/i3/config`.

The first file overwriting everything else is `~/.config/i3/config`. That's great, because it's the configuration file we'll use. In practice, this means that any option or keystroke set in this file will overwrite the same options or keystrokes set in other files.

### Default Configuration

Let's dive a bit more into i3's config. First, let's open the file with Neovim:

```
nvim ~/.config/i3/config
```

One of the first lines of the config will define your modifier key ( `$mod` ), as I explained above. You can modify it here if you like.

To see every possible value you can use as modifier, you can run in your shell the command `xmodmap`. It will list everything you can use.

As you can see, you can define variables using the keyword `set` followed by the variable name ( `$mod` here) and its value ( `Mod4` ).

Below in the configuration file, you'll see something like this: