Computer Stuff for Dum Dums

Preface

Greetings, welcome to this super unnecessary book on computers that I'm writing because I'm done with giving tech support to people and telling various people the same thing over and over again. The purpose of this book is to excite people about computers, and motivate them to play around with software and hardware. This book is a free and open source project under a super chill license. If anyone is interested in converting this book into a wiki style website and host it on their git, you are very welcome to do so as long as you fork it from my own git. I've tried to keep it as noob-friendly as possible, because I've been a noob myself (some people on the arch irc will argue that I still am, I agree). This book is best viewed in a pdf reader (unless a web-based wiki is made) on a desktop/laptop with an internet connection, because I have abused the hyperlink feature to link to various resources online, contributed by people who can explain things better than I will ever be able to. Though there are some basic per-requisites you require in order to understand this book, which are: nah you're cool, if you could read these previous sentences, you can read this book pretty easily. I will not be requesting or accepting any form of donations or gifts for this book, because that isn’t the purpose if this book, I will not be charging a fee for the distribution of this book either, feel free to spread it far and wide. There is no point of printing this book on paper because the hyperlinks will be rendered useless.

Note: All links to the arch wiki are only meant to be read, most of the commands listed on those pages will only work on arch or arch-based systems. If you’re not an arch user, treat the wiki like a textbook, not like a manual.

no spacebars or enter keys were harmed during the writing of this book

In memory of Dennis Ritchie, Alan Turing and Nikola Tesla, and in honour of Ken Thompson, Richard Stallman, and the countless nameless contributors of free software worldwide, without whom, none of the technology we have would have been possible.

Introduction

The book follows a bottom to top approach in computers, starting at the basic hardware level, explaining the function of each component, and progresses towards the software end. It touches the subject of free software and why it is important for the industry to follow the ethics, and then guides the reader through installing a free and open source gnu/linux based distribution. The book then progresses towards mobile computers and handheld devices, priviledge escalation and ultimately towards a general guide to obtaining root access on android powered devices. Readers are encouraged to visit the hyperlinks in the book, which lead to external resources which are an aid to help understand the concepts better, along with links to statistics which support the claims made in this book. Good luck on the journey you’re aboout to undertake, and may the odds ever be in your favour.

Hardware

Lets begin with the hardware components of a computer, which I'll divide into 2 categories, core components and I/O devices.
Note: The plastic box you put all your components into is not called cpu, its called a computer/rig/chasis/case etc.
Core components (these are the things you put inside the case):

1: Motherboard: This is the big green flat board which sits at the heart of the case, it has a lot of ports, sockets and whatnot.

2: Processor/CPU: This is the chip that sits in the cpu socket of your motherboard, its a square-ish thing with silicon on one side and a grey label/lid on the other. The processor has a heatsink attatched onto it, which in turn is conncted to a fan or a liquid cooling system.

3: RAM: Also known as random access memory or just memory, RAM is the flat set of sticks sticking out of the motherboard. Ram sticks are usually 2-4 in number but you can also use a single stick (although it hinders performance, so 2 sticks of 4gb ram perform better than 1 stick of 8gb). This is volatile memory, which means it forgets data when it is switched off.

4: Power Supply: This is a big box that is usually location at the rear bottom of the computer case, you have to connect the power supply to the wall socket and the motherboard to the power supply in order for your computer to work =).

5: Fans(the more the merrier): These are, well, fans which are used to..... you guessed it, cool the computer using airflow. (Although some computers come with liquid cooling, but if you know about liquid cooling what are you doing reading my book?)

6: Graphics Card/Video Card/GPU: The graphics card is a thick card with fans built into it, which is attached to the motherboard and also has a video output port which sticks out the back of the computer case.

7: Storage Drives: There are two types of drives:

- Hard Disk Drives: these things are cheap, hold a large amount of data, have an actual disk spinning inside them, their only drawback is that they are slow.
- Solid State Drives: these things are small and quick, they are expesive though and you'll get less storage than a hard drive for the same price, they have another drawback, SSD's tend to have a shorter life-span than hard drives.

Storage Drives are non-volatile memory, which means they save their data which can be accessed after shutting down and booting up after any duration.
Learn about filesystems here.

This page is epic!

I/O Devices (input output devices):

1: The Monitor: This is the screen you're looking at right now unless you're using your phone.

2: Keyboard: Well, the thing with the keys you doof.

3: Mouse: I'm starting to realize this is pointless.

4: Printer: Converts soft copies into hard copies.

5: Scanner: Scans hard copies to make soft copies.

Now that we're done with hardware, let’s move on to other wares.

Not-So-Hard-Ware

Now that you've made it through an intense session of reading about hardware, lets start with the other things that your computer has other than things you can physically touch.

Firmware: This piece of code is hardwired at the basic level into the storage, its stored in non-volatile memory, and basically provides low level control over basic hardware devices.

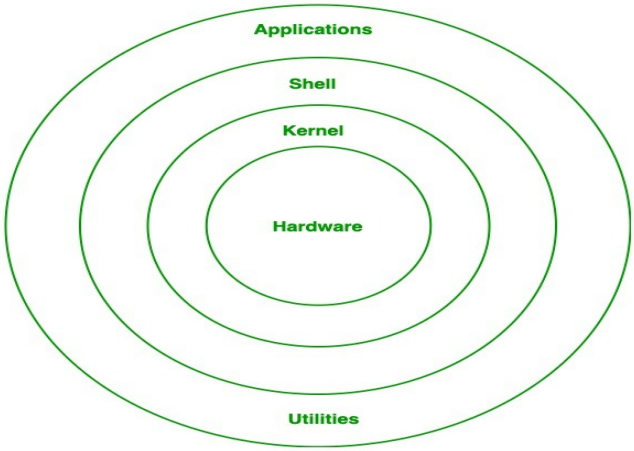
Software: Software is the rest of the operable code stored on your computer.

Software has the following categories, from the lowest level to graphical user interface:

1: The kernel: this is a very important piece of code which helps the operating system contact the hardware and make it do things. Examples: The linux kernel, FreeBSD, Windows NT kernel.

2: The Operating system: An operating system is basically everything a computer needs to fully function, it consists of ginourmous blocks of code which are essential in making your computer work in order for you to get those cute cat videos from the internet. Examples: Windows, macOS (bleh), GNU/Linux distributions like Ubuntu, android (yes phones are computers).

3: Applications: Applications, or apps are software designed to perform a specific task, be it browsing your storage or listening to music, or deleting your ex’s photos.



A note on Free Software:

(more like a senile rant)

Would you like for your car manufacturers to be able to drive your car wirelessly, and do anything that they want with it? Would you like the electrician who did the wiring in your house to be able to control your electronics, and dictate terms about how long you can can have the lights on? All of this might seem a little dummy dum to you but this is exactly what you have been doing with software.

A little lesson in how code works :

Computers (processors) can not understand anything other than 1's and 0's, which in the physical world translates to whether there is a voltage in a sensor or not (yeah they're sorta dumb). But humans use the decimal system because we have ten fingers, and we have alphabets and all sorts of razzle dazzle we use to communicate, and so we need computer languages to interact with computers, I'm not your cs prof so off you go to read about them. So whatever code you write (or ~~steal~~ borrow if you're a professional programmer), needs to be translated into 1's and 0's for a computer to execute. The process of translating this code is kinda sorta called compiling. Code from languages like C, C++ has to be compiled into executable binaries (because they're 1's and 0's, ya know...) in order to be executed. You only need the executable binaries to run a program, like a game or a pdf viewer. You don't need the actual code written by the programmer, also known as the source code, or just source.

Programmers are nifty little buggers, so they only give you the executable binaries, and not the sourcecode, and you can not read what a program is capable of doing. This type of software is called proprietary software. ← Click on this link, please, I beg you.
The opposite of proprietary software is free software, which fulfills these four requisites :

- The freedom to run the program as you wish, for any purpose.
- The freedom to study how the program works, and change it so it does your computing as you wish. Access to the source code is a precondition for this.
- The freedom to redistribute copies so you can help your neighbor.
- The freedom to distribute copies of your modified versions to others. By doing this you can give the whole community a chance to benefit from your changes. Access to the source code is a precondition for this.

There is a practice among certain users of treating all proprietry software as malware, and in my personal opinion it's a very practical approach because YOU JUST CAN **NOT** TRUST PEOPLE.

Also, free software is mostly community driven, and has a democratic approach, the freedom to achieve things in computation will overwhelm you atleast, and turn you into a caveman who keeps tinkering with linux, I'm a victim of the latter.

To further study software ethics, you can try the Free Software Foundation, and this gentleman who claims to cut water, this bald guy or this bald guy.

Glossary

Some terms before we get started with the cool stuff.

These terms are often used in the computer world:
GUI: Graphical User Interface - The gui is the way a program looks and feels, and interacts with you, you use the gui to perform opertions by clicking, typing or tapping on touch devices.

CLI: Command Line Interface: This type of interface only lets the user interact the computer via tha keyboard and nothing else (scary right?).

Terminal: A terminal is an application used to provide the user with a cli.

OS: Operating system.

Distro: Distribution - distros are various flavours of the gnu operating system paired with the linux kernel, they all essentially have the same basic features, but differ in the apps and other programs (desktop environments, package managers etc) provided with them.

Machine : In this book, the words machine/rig/system mostly refer to a computer.

VM: A virtual machine.

Booting

Now that you've acquired so much knowledge, lets start our computer shall we?

This is the boot process used by the computer, when it is turned on:

When a computer is switched on, the control is given to the BIOS or Basic Input Output System. It then POSTs, which means power on self test, it basically checks what all hardware it has connected and whether they are working correctly. Following this and a few other checks, the BIOS then jumps to the MBR or the Master Boot Record, which is a set of data written at the beginning of most storage devices. The MBR contains data about what filesystems are present on the disk, along with a piece of code to load the bootloader. Different types of operating systems have different types of bootloaders, for example Windows has Windows Boot Loader, linux distros can choose from a variety of bootloaders like GRUB, syslinux and rEFInd. The function of the bootloader is to start the operating system and load essential things into the RAM. After the operating system is loaded, you can proceed to log in and start wasting your time on these godforsaken machines.

This is an extremely childish and simple definition of the process, and I suggest you read this wikipedia page for a deeper understanding:

Wikipedia: Booting : Modern Boot Loaders

What RAM does vs what the proccesor does:
A computer is just a piece of silicon that we have, with our black magic, made into a pice of silicon that can do math. A CPU, or a processor can only solve math problems. The operation of a CPU is to fetch, decode and execute. But the math problems also have to be remembered in order for the calculations to make sense, hence all the data to be processed is stored in the RAM (with the exception of cpu cache) billions of times. So RAM just stores the data, and a cpu is a math problem devouring monster that constantly keep chomping on numbers and spitting them out.

Standard Applications

Browsers: These are the “internet” clients which you use to access webpages on the internet. Remember, chrome is a broswer, google is a search engine and a giant evil corporation after your data. The modern web is so broken and inefficiently written in inefficient languages, that you need a senseless amount of hardware resources to load a webpage, and this has led to browsers becoming “bloated”. There is no best browser, some just suck less than others.

File Managers: Applications designed to let you view and perform operations on files present on your system or even on another machine (remember, it’s not a cloud, it’s just someone else’s computer). This is the “This PC” for windowsfolk and “Macintosh HD” for applefolk.
Fun fact: You can even use your internet browser to browse local files.

To view different types of files in your computer, you will use image viewers, media players, document viewers, text editors and a plethora of different programs, but don't worry, they’re all just “apps” to you ;}.

Document handling/Office software: These are suites of applications made to edit/create/view documents of various types, ya know, work stuff. Examples include LibreOffice, WPS Office and our lord and saviour Microsoft Office 2025 MAX PRO ELITE.

If you’re a gnu/linux user, this page will help you find and install any application you need. If you’re a Windows user, install gnu/linux. MacOS users might find some programs to use in the list as well, because it is an evil cousin of gnu/linux distributions, being a derivative of BSD.

And we’re done!!

Congratulations!! If your purpose was to get a deeper knowledge of what computers are and how they work, you’ve done it, everything that follows in the book is practical learning, and can only be done with a computer (unless you want to see white balding men do this boring stuff on youtube), so I would suggest you stop reading unless you’re really curious about computers or have a computer to do these “crazy” expermints on, so long……

WAIT, you might be interested in these external links if you liked what you read:

Youtube : Linus Tech Tips (nice hardware stuff)

Arch Wiki

ComputerHope

Install Gentoo Wiki

So, what is this loonix you speak of?

Obligatory Stallman Interjection:

I'd just like to interject for a moment. What you're referring to as Linux, is in fact, GNU/Linux, or as I've recently taken to calling it, GNU plus Linux. Linux is not an operating system unto itself, but rather another free component of a fully functioning GNU system made useful by the GNU corelibs, shell utilities and vital system components comprising a full OS as defined by POSIX.

Many computer users run a modified version of the GNU system every day, without realizing it. Through a peculiar turn of events, the version of GNU which is widely used today is often called "Linux", and many of its users are not aware that it is basically the GNU system, developed by the GNU Project.

There really is a Linux, and these people are using it, but it is just a part of the system they use. Linux is the kernel: the program in the system that allocates the machine's resources to the other programs that you run. The kernel is an essential part of an operating system, but useless by itself; it can only function in the context of a complete operating system. Linux is normally used in combination with the GNU operating system: the whole system is basically GNU with Linux added, or GNU/Linux. All the so-called “Linux” distributions are really distributions of GNU/Linux.

Alright, lets linux. A gnu/linux operating system is one which uses the gnu system+the linux kernel. And they work well together. So well that the entire web(hyperbole) runs it.

And why is it so good?

There’s several reasons behind the success of linux based operating systems:

- Opennes - Linux based distros are usually community driven, which means all code is pulicly audited and any bugs/security vulnerabilities/improvements can be easily brought to the attention of the original author. This is made possible by git (git is not github, just like news is not New York Times). Git was created by Linus Torvalds, for a better way of version control for linux.

- Efficient Code – There have been several instances of Linus being an unpleasant person, but the code thanks to him, other kernel contributors, and the open source community has been extremely efficient in getting things done.

- Customizability – The open nature of the system means there is complete access to the user, at all times. This leads to some very insane levels of customizability. And having access to the sourcecode means you can theoretically make the software do literally anything you want with it. Trust me, if you want your OS to boot with an animation of dora the explorer beating Julius Caeser over the head with a hockey stick, then log into your email account and fetch new mail, resume music from where it left off previously, download all available updates, fetch the top voted post from your favourite subreddit, and post a tweet complaining about your ineternet speed to your ISP, without even having to press a button, you can do that.

- Versatility – The beauty of free and open source software, is that it can be forked and made into a different thing altogether, which has led to this mammoth family tree of gnu/linux distributions. Every sigle one of these distros offers something entirely different, and a linux user will never run out of options and alternatives.

- Security – If you're a linux user, you're far less likely to be pwnd by some cheap malware. This is both due to statistics and user awareness.

- Impressive Resource Management – Linux brings back old computers to life, with there being such a versatile and enormous plethora of distros available, there are bound to be some very minimal ones. The most user-friendly ones from this category have proven to be a suitable replacement for old Windows computers.

- Privacy – All proprietary code is controlled by the proprietor, here's a fun read for Windows users, and for apple users.

How do I linux?

You can use gnu/linux based operating systems I virtual machines, dual boot setups or a standalone installation on a machine.

- ## Virtual Machines :

  Virtual machines are small emulated environments where you can install and/or run an operating system. You could think of them as a computer within a computer. Virtual machines allow you to enjoy the beneficial features of multiple operating systems without having to reboot, as a single processor can run only one operating system at a time, unless multiple operating systems are configured in a host/guest arrangement, which is what virtual machines help us achieve. The main OS you boot into is called the guest OS, and the OS you run on a virtual machine is called the guest OS. Notable virtaulization software are virtualbox by Oracle, and vmware.

- ## Dual Booting :

  Even though virtualization software has come a long way, and is a very noble endeavour, it has faailed quite noticably to deliver the full bare metal experience that an OS offers when run directly on hardware. Remember, virtual machines run on emulated software. To enjoy the full experience, people choose to install a gnu/linux OS on their computers, along with Windows/macOS. This is called dual booting, where there are two (and very commonly even more) operaating systems present on their computers, and the user can choose which operating system she wants to boot into at the initial stages of the boot process via a boot menu or a GRUB or GRUB-like interface.

- ## Standalone Installation :

  For users who do not need more than one operating system on their machine, they can choose to only keep one. Most computers and laptops today are sold with a standalone Windows installation because of Microsoft's aggressive market tactics. Apple ships all their computers with macOS and makes it very hard for a user to even install another OS because Steve Jobs' vision to make apple a cosed ecosystem where users are slaves to the proprietor i.e. apple. To all the people who think he was some technological pioneer who revolutionised tech, you have the wrong Steve (it was Wozniak) "Our first computers were not born out of greed or ego, but in the revolutionary spirit of helping common people rise above the most powerful institutions." What Microsoft and Apple are doing goes against everything their ancestors stood for. If you even credit Jobs for coming up with the smartphone, here you go.

Virtual Machine Installation

- ## Choosing your virtualization software
  The first step is to choose which software you would like to use to create a virtualized environment which you want to run your guest OS in. Common software are:
  - Virtualbox : https://www.virtualbox.org/
  - vmware (proprietary)
  - KVM : https://www.linux-kvm.org/page/Main_Page

  After you are done with your decision, you van download and install whichever VM software you want.

- ## Choosing what OS to use
  The second step is to choose which OS you would like to use as a guest OS. It is a very complex decision so please refer to my distro guide from later in the book. After you have made the decision, proceed to download the iso image of the operating system, provided on their website (keep in mind of whether the iso is for 32 bit [x86] or 64 bit [x64] processors, and download according to your preferences. Read more about processor types here).

- ## Installing the OS as a vm
  Now the installation for every OS and every vm host software will differ, and you will have to refer to an online guide for the first few installs but a general idea of the process is:
  - Creating a new virtual machine and giving it the hardware specifications that you preper (cpu cores, disk space, etc.)
  - Booting into the vm and choosing the iso file to boot from (choose the iso file which you downloaded earlier)
  - Going through the installation of the OS in the vm, (unless you only want to run it live without installing).
  - After the installation finishes, reboot the virtual machine and unmount the iso image to boot into the installation.
  - Congratulations, you are now the owner of a virtual machine, may there be many more to come.

Note : A user should always try out a new os on a virtual machine before installing it on their hardware, and only use it on hardware if it looks promising.

Dual Booting

If you're satisfied or interested in an OS after using it in a virtual environment, it is time to install it on bare metal (which means installing it directly on the hard-drive) along with Windows or macOS and having the option to boot into either OS for your computing tasks.

- ## Choosing your OS:
  Same as the virtual machine. Choose and obtain the iso image of the OS you want to install on your computer.

- ## Making a bootable medium:
  Most gnu/linux operating systems use a live bootable medium for installation, and it is usually a usb/cd/dvd. To make a bootable usb from windows, you need a program like rufus, etcher, or win32diskimager, linux users can use the dd command in their bash/zsh terminals.

- ## Partitioning your drives:
  An operting system needs an indepependant partition (or several) formatted in a suitable filesystem format to operate. Most gnu/linux distros will run on the extended filesystem family or ext* (ext2/ext3/ext4), Windows runs on NTFS, whereas macOS uses the APFS(Apple FileSystem). For cross compatibility with operating systems, a FAT16/32 were used but now have been replaced with exFAT due to limitations. Furthermore, EFI Partitions are formatted in FAT12/16/32. Swap partitions have a independent filesystem format. You can manage your partitions either via a disk management utility provided by windows, or via the disk partitioning tools provided in your live usb after you boot into it, or via terminal based disk managmenet programs like cfdisk and fdisk which are commonly found in gnu/linux installation mediums and installations. Also make sure to have a backup of all your important files and directories before you get to do major partition manipulation, in case Murphy's Law decides to have a little fun with you.

- ## Booting into the live usb:
  Some laptops and computers will directly boot into a bootable medium (usb/dvd) when they detect one. Otherwise, you can go into the BIOS of your machine to set the boot order (use the internet to find out how to enter the BIOS in the make and model of your machine). Or, some machines provide a shortcut you can press at the boot time when the manufacturer's

logo is displayed, to enter the boot menu and choose which OS to boot from.

- ## Installing the OS:
  If you've chosen a common gnu/linux distro, chances are that it comes with its own installer, and it will guide you through the installation process. Please, for the love of God, DON'T nuke your partitions with sensitive data. Remember, installation of any program is basically putting executable binaries in a directory from which they can be accessed and executed, and the installation of other programs required for its smooth functioning (depedancies). For example: most games require DirectX, the .Net framework and Visual C++ to run on Windows.

- ## Setting the boot order:
  After the installation is complete, you'll be asked to reboot and remove the usb/dvd from your machine. If the installer configured the boot menu entries to prefer your new installation, you'll boot directly into your new OS. Otherwise, you will have to edit the boot preference order in your BIOS. You can also configure which OS to boot into by defualt in your bootloader of choice in your gnu/linux distro (GRUB/rEFInd/syslinux).

Congratulations, you are now a gnu/linux user. Please proceed to grow a neckbeard and cuss out any Windows/macOS users and assert your dominance by showing off your rices on r/unixporn.

After some usage, users might also consider removing windows entirely from their systems, move their data elsewhere and only using the OS they installed. This step is only recommended if you are completely aware of what you are doing and what you will lose by letting go of windows. The OS then becomes a standalone installation.

Furthermore, people delve into LibreBooting and CoreBooting their machines, because some companies be sneaky beaky like.

So which Distro should I choose?

A note on the community:

GNU/Linux users have been notorious for incessantly asking (telling or even shaming) others to start using gnu/linux, be it for ethical reasons or just to paint an elite image of themselves. And this does not stop once you're a linux user, you will have Arch users bragging how difficult it is to install Arch Linux, because it does not come with an installer. You will have Gentoo users dunking on Arch users because installing Gentoo is even harder and more time-consuming, as users have to compile directly from the source code after editing flags according to their own preferences. Then linux-from-scrath users will claim to be the ultimate wizards, as it isn't even a distribution, it is just a book which tells you how to acquire sourcecode for the various programs you want, and build everything yourself. The point here is, gnu/linux users tend to be connected to the software that they use on an emotional level, and it is wisely thought that linux users are elitist tend to look down upon other people. While parts of this behaviour can be experienced on various forums on the internet, it is not entirely ture, and communities will mostly treat new users with care and help them with the learning curve. That being said, the distribution you choose should not be difficult enough to install and maintain that it makes computing dreadful, which means new users should think twice before they consider using one of the three distros mentioned above as their first ever venture away from Windows/macOS, unless they're out for adventure.

All information about any distro can be found at distrowatch.com, the distro pages on the website will link you to their official websites, along with providing some useful information about the project.

Your choice about your OS should depend on what you aim to extract from it. If you want something very similar to Windows macOS in terms of appearance, you could try Linux Mint and Deepin respectively. If you want to quickly set up your OS for gaming, try Pop! OS. If you want years of uptime and stability and very dependable and infrequent version upgrades, go for Debian or any of its derivatives built for stability. If you're looking for a regular desktop experience and have work to do, you could try openSUSE. If you're a budding penetration tester, look into Kali, Parrot OS and BlackArch. If you want a portable linux distro you can carry around on a usb stick, try Porteus or Puppy or MX Linux. Notable mentions in distros you should read about : Manjaro, Fedora, Centos, RHEL.

If you're wondering about Ubuntu, here's why I didn't mention it.

I've mentioned these operating systems along with the attributes that they are known for, but, gnu/linux being a very versatile and open system, allows you to configure any OS to suit your needs, so your choice does not matter as much as you think it does, as long as you make a reasonable choice. More here and here.

I use Arch BTW

Tinkering with your system

So you've installed a gnu/linux system and are having a fine time looking around and getting a feel of the new environment. What if something in the UI interrupts with your workflow? Change it. Do you see a program you do not want on your system? Remove it. Do you want an alternative to any software because the current one does not offer the functionality you want? There's tons! You can change/edit/remove/install anything you want, in any way you want. Anything? Anything. This is the essence of free software, it is free as in freedom, not free as in free food. Here are a few things you might want to play around with to get the aesthetic you want, or to provied you with the environment which suits your workflow the most:

Desktop Environments :

This is probably the first and the most impactful choice gnu/linux users make in their systems. A desktop environment controls how applications are displayed, how various workspaces/tags/virtual desktops are handled, and the look and feel of the desktop. Desktop environments also come bundled with a suite of applications, which provide a unique experience with different DE's. More advanced users may want to replace their desktop environment with a window manager, which only dictates how windows are displayed, and the rest of the functionality is configured by the user. Window managers tend to be keyboard-driven, faster and more efficient. If you're conflicted between two desktop environments, you might want to read their comparison on slant.co, which might help you decide. Notable desktop environments are : GNOME, KDE, Mate, XFCE, Budgie, Deepin, Cinnamon. Notable window managers are: i3, i3-blocks, openbox, awesome, dwm, xmonad.

Bootloader :

If you are dual-booting, you might want to configure your bootloader to achieve the functinaity that you want. You also may want to change your bootloader to one which you like for its features/visual appeal. Users also configure their bootloaders to ahcieve the fastest boot time. Notable bootloaders are : GRUB, rEFInd, syslinux, clover, LILO.

Display Manager :

A display manager is the piece of software responsible for offering a log-in screen to the user, offering some configuration options, choosing the Desktop environment to start up (yes, you can have as many DE's installed as you like, and switch between them by logging out), and starting up the display server. Advanced users may also wish to do away with their display managers (dm's), and logging into a cli environment, and then launching their display server via a command. Notable display managers are: entrance, gdm, lightdm, sddm and xdm.

Display Server :

The display server is the fundamental program which controls everything which happens in a GUI environment. There are mainly two display servers in popular use: Xorg (the old hat), and Wayland, the flashy new youngster. Xorg is based on a client/server model, and hence can be used by machines remotely. More here.

Furthermore…

These are the fundamental pieces of software which you will have to make choices about, if at all you choose to make changes to the default system provided to you. But, you also have the freedom of choosing any set of programs you want, like terminal emulators, file managers, archive managers, IDE's, disk management software, system monitors and a lot more. The list of applications page on the Arch Wiki is a very good resource for finding applications of your desire and needs, you can also refer to comparison pages by slant.co or even wikipedia.

Android

A brief history:

Initially aimed as an operating system for digital cameras, the project later turned into a poetantial rival of operating systems for mobile devices, with the vision of providing consumers with an alternative to the closed and orwellian system provided by carrier companies. These cellular companies controlled the operating systems, hardware and everything other decision related to mobile phones. On the other hand, Android sought to be the open alternative, following the open source philosophy. The company was acquired by google, which helped the project grab the enormous market share in mobile devices aand wearables, which it now has. Fun fact: android is based on linux. More here and here.

Mobile devices have become an extension to the soul of the user now. The sensitivity and value of data stored on and handled by these mobile computers is at an all time high, and is set to only increase. A mobile phone paired with an internet connection is enough to make someone an internet sensation and a worldwide celebrity. It is capable of swaying elections and starting political movements, trading millions in stocks and running companies into the ground. It can even let you click pictures of yourself and upload them on social platforms where you can garner likes and cooments from people who pretend to be your friends!

With such a potent device in your pocket, not realizing its potential or underestimating what it can do can turn out to be fatal, or worse. The freedom and privacy situation is even worse when it comes to mobile devices. Apple's iOS is just state-of-the-art spyware, and Android, which started out to be an open and free system has become the very thing it swore to destroy. The stock android code that Google releases is open source software, but mobile companies pair it with non-free proprietary software of their own, and ship their devices with loads of bloatware and tools meant for data collection. In the immortal words of Benjamin Franklin, "Those who would give up essential Liberty, to purchase a little temporary Safety, deserve neither Liberty nor Safety." I would say the same applies to convenience. (Yes, I know it's out of context.)

Here's an interesting perspective on mobile devices.

One way to decrease the amount of data big brother collects from you is to obtain root access and offloading or uninstalling some apps which you are not able to remove without root access. Root access is the ability to make administrative changes to your system. The term comes from Unix, where the Superuser is often called root. The root user can modify/read/write anything on the system. Do not confuse this with administrator priviledge in Windows, which is a joke. This access is obtained by "rooting" your phone if it is an android, and "jailbreaking" if you bit the apple. Hats off to this legend.

Rooting your phone

The best resource you will ever find for android modding/rooting/configuration is the XDA forums. This is the largest community of android ethusiasts who like to play around with various apps, skins, UI's and even kernels. The website has a different subsection for every handset model, and these subsections include user reviews, QnA's, news and discussions, accessories, guides to unlocking various capabilities, rooting tutorials and much more. The forum also has a strict policy about GPL v2, which means all software shared on the website must be licensed under the GNU General Public License, which ensures that anyone and everyone can view and edit the source code to their liking, and follows the 4 freedoms of the free software ideology. This ensures that malicious entities can't post their software on the forum, for innocent and clueless people to install and run, ultimately compromising their system, bricking their phone, mining cryptocurrency, or making the device a part of a botnet.

There is a copypasta which gets used a lot on the forum :

* Disclaimer
* I am not responsible for bricked devices, dead SD cards, thermonuclear war,
*or you getting fired because the alarm app failed. Please do some research
*if you have any concerns about features included in this ROM
* before flashing it! YOU are choosing to make these modifications, and if
* you blame me in any way for what happens to your device, I will laugh at you.
* BOOM! goes the dynamite

The same applies to the conents of this book.

To get a general idea of what root access is and how users may gain root access, this article on xda and this subsection of the wikipedia page might be helpful. The general idea is to escalate your privileges to the highest level and obtain access to hidden or denied parts of the device. Rooting can either be through exploits which take advantage of the vulnerabilities found in device firmware and flashing a custom firmware which allows us to access root privileges. Some companies choose to allow their users to unlock their bootloaders and installing their own firmware without exploiting a vulnerability, but the support provided is limited and you're basically on your own most of the time.

General steps in rooting your phone include:

1. Unlocking your bootloader

This step usually erases all the data on the device, and allows you to access the boootloader of the device, from where you can choose which ROM/recovery you want to boot into. More here.

2. Flashing a custom recovery

The stock android recovery environment provided by google is very limited in capabilities, as it can only flash official ROMs, restore the device to the factory version, wipe the cache partition etc. A custom recovery allows you to do advanced changes to your device, like installing a third party ROM, taking full device backups, etc. More here.

3. Actual Rooting

This step involves flashing a zip file from within a custom recovery, which “roots” your phone. A popular software for this is Magisk. After this step is complete, you are the owner of a rooted phone. You may now install a root manager which lets you control which application has root access, hiding the fact that the phone is rooted from certain apps, etc. Examples of root managers are SuperSu and Magisk Manager, with the latter being more popular with modern phones.

4. Custom ROM?

Users may wish to exercise their newfound freedom to install a custom ROM of their choice. As you know, ROM stands for read only memory, which is irrelevant to what an android custom ROM is. A custom ROM is a spinoff of android firmware, built to provide some desirable features advanced users may feel the need of but can’t find in the stock ROMs. They’re also used to achieve a performance benefit because they support overclocking or underclocking the cpu, and help extend the battery life by being a more open and configurable system. Popular ROMs include LineageOS (previously CyanogenMod), PixelExperience and AOSPEXTENDED.

If you’re not interested in rooting your phone, and are just looking to pay around with your android in a secure environment, you could tinker with themes, launchers and the like, to achieve a better workflow.

If you want to bring the desktop linux experience to your mobile device, you could look into VNC servers, or even installing a gnu/linux distro on your phone using linux deploy or something similar.

Afterthoughts

I hope that you enjoyed reading this book and learning as much as I did, and I also hope that I've contributed to your dedication towards learning new things and becoming a better version of yourself. I hope you bought into the philosophy of making the community that serves you with these amazing bits of technology grateful that they serve to you, by paying them back in kind.
That'll be all, grazie mille for reading this book, good luck, godspeed, and goodbye.