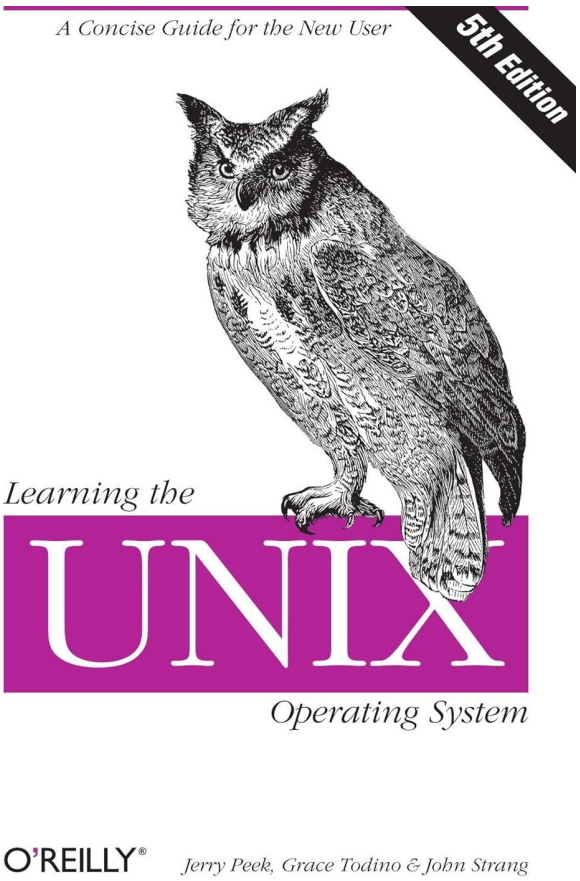



# Unix OS

---

**Learning the Unix OS - 5th Edition**

**Jerry Peek, Grace Todino & John Strang ( O'Reilly, 2002 )**

This book is now over 20 years old, yet it has aged gracefully.
Roughly 90% of the information inside is still relevant today and the other 10% can be treated as a history lesson.
The reason can be found in the book's preface.

> **This book teaches basic system utility commands to get you started with Unix... We cover a commands most useful features...**

And it more than lives up to its purpose. While Linux & Unix have changed a lot over the years the command line itself hasn't changed radically. The reason why the commands shown are interchangeable is that many of the core commands are identical in terms of syntax and functionality between Unix & Linux. There are of course some differences, but they are not enough to be a problem at our current level. We won't learn about Modern Filesystems or Containers, but we will learn enough to get even more confident with working through Linux.

## Book Overview

Our next step in the systems fundamentals course will be **Learning The Unix OS by O'Reilly**. You may treat this book as a **Practical Guide to Unix**. This book is 157 pages long and will take you 7 days to finish if you dedicate between 2-3 hours every day to work through it.

When you look at the title you will notice the word **Unix,** but don't let the word Unix discourage you. Much of what we will be going over will be regarding the Command Line Interface. Which for our purposes is practically interchangeable with Linux. Don't misunderstand, they are two different operating systems, with the key difference being that **Linux is Open-Source** while **Unix is Proprietary**

The main goal of this book is to serve as a **Practical Starter Guide**, covering the main essentials of the CLI and Unix OS. That is why this book is still remains relevant even today. While we did go through an introductory course already, that being the Linux Upskill Challenge, there are some nuances that that this book goes through that will be very helpful to us.
We'll explore the various types of commands, Windows Systems, basic File Management, I/O-Redirection and even some interesting pieces of Linux history. When I finished going through this book, I felt confident that I could start working on this roadmap and bring something interesting to the table. Although we still have a long way to go before we can call ourselves true adepts, but steadily, we are making progress.

## Personal Note

**( The following has been added to the overview page. But I am keeping this here as this was when I thought of the idea. )**
Throughout this journey I will do my best to be as objective as I can. However, I am still in the middle of my learning and I am only human. While technology itself is neutral, the way we use it, how we think about it, what we think is best moving forward and so on, are things that are human viewpoints. There are of course times when things are simply 0 or 1, but as you go through your own journey you will find that you will have to come up with your own definitions and reasons for doing things backed by personal preference or well researched analysis or simply that's the way things are done.
So, to carry on with that spirit I will place the **sparkling star icon** whenever certain notes or topics are things I am speaking my own mind on. I encourage you to come up with your own analysis or thoughts on those topics.

## Goals of this Book

- Gain a basic understanding of the various use cases for **Linux** as well as cover its **strengths and weaknesses**
- Learn how to find the **current shell** you're working in: **ps $$** & **echo $SHELL**
- Cover the formatting of Unix command arguments
- Go over Jobs and how to: show, restart, stop and kill them: **jobs** | **fg** | **ctrl-z** | **kill %**

- Cover Window Systems going over virtual consoles, terminal windows vs alphanumeric terminals
- Root vs relative path names
- 3 permission types of files and directories
- Understand group ownership and how to give specific groups access to files and directories
- How to remove files with spaces between them
- Cover the Primary Wildcards **( *, ?, [ ] )**
- Go over the main ways to find files in Unix **( find | locate )**
- I/O Redirection with: **( >, >>, <,  |, grep, sort )**
- Pager program commands **( less | more | pg )**
- Examine CLI web browsers and their use case

## Course Notes

I quite enjoyed making the notes for this book since it led me to making several web searches on various different topics about Linux both old and new.

- **Why should we use Linux?**
  - The FOSS nature of the OS allows groups to have a very powerful tool. Making it a great choice for smaller teams or groups without much money.
  - Linux is also made to be run on practically anything. From the weakest hardware to the most powerful, although Linux's best potential can be extracted on more powerful systems.
  - As another note, Linux supports both windowed and non-windowed systems.
- **Chapter 1 - Getting Started**
  - You can find the current shell you are working in with **ps $$** & **echo $SHELL**
  - **whoami** - Will show the username you are logged in as
  - **who** - Will show a list of users that are logged on
  - There are control characters you can use in Linux. The most common of which is **CTRL+C** & **CTRL+D**
    - **CTRL+C ( ^C )** : **Cancel the currently running process or command**
    - **CTRL+D ( ^D ) : Marks the End-Of-File. Will have no effect if the program isn't reading input from the terminal. If used in a shell terminal it will exit.**
  - **Unix Command Arguments**
    - **command** option(s) filename(s)
    - Options modify the way a command works
    - Can be letters **( -ltr )** or words **( --delete )**
    - At times there will be special exceptions for certain commands
  - **Daemon** - May also be called a background process. It is a Unix or Linux program that executes in the background. The main distinction between a daemon and a job is that a daemon operates autonomously, performing tasks without user intervention.
  - **Job** - Any programs that are started interactively by the user that don't detach in the background, ( ie. becoming a daemon ) are jobs.

        - **jobs -** shows running jobs in the current shell
        - **ps** - lists running processes
        - Jobs in the foreground can be stopped with **ctrl-z**
        - **job_name & -** starts a job and places it as a background process
        - **fg -** places a background process into the foreground
        - **kill % job_num** - sends a signal to kill a job or process
    - **Chapter 1 - Summary:** In this chapter we covered the basics of working in a Unix environment. We learned about **processes**, starting them and stopping them, We also briefly covered **control characters** and some **command syntax**
- **Chapter 2 - Using Window Systems**
    - Alphanumeric terminals handle a single session on a single screen
    - Virtual consoles exist as a way to get Fullscreen log-in sessions on the same OS. You may switch between consoles with **Ctrl-LeftAlt-[F1...F6]**
    - Unix Systems have windows managers. One of the most important types being terminal windows
        - A Unix session inside a shell prompt is a terminal window
    - Terminal windows allow us to interact with Unix from a shell prompt
    - Use **Ctrl+Alt+F6** to get a non-graphical log-in prompt
    - **Chapter 2 - Summary:** In this chapter we learned about the history of windows terminals. And we also covered various ways to deal with crashing windows
- **Chapter 3 - Using your Unix account**
    - A file is the unit of storage in Unix. Files can hold anything from Text to programs, digitally encoded data, etc.
    - Any time we log into a Linux/Unix machine we will start in the home directory which is also our working directory
    - All directories in the Unix system are organized into a tree-like hierarchy structure.
    - **Networked Filesystems** allow us to access a remote computer's files and have them appear in our computer's directory tree.
    - **Absolute** vs **Relative** Paths
        - **Root starts with a slash ( / )**
        - **Relative path names never start with a ( / )**
        - **Unix/Linux filesystems can also hold things that aren't directories or files.**
            - **Symbolic-links, FIFOs, & sockets**
    - **Sysadmins** are users that have full control over Linux files
    - There are 3 permission types to files and directories
        - **Read | Write | Execute**
    - Both files and directories have their own permissions
        - **File Access Permissions | Directory Access Permissions**
    - This book goes over **chmod** and **chattr** as well. You can find this information in the **Linux Upskills Challenge Notes** Section.
    - Protecting and Sharing Files
      - Group ownership is the way to give certain groups of users access to files or directories
        - List all groups: **groups; compgen -u; compgen -g;**

    - Add groups: **sudo addgroup local_group**
    - Add users to groups: **sudo usermod -a -G local_group user_name**
    - Change group owner: **sudo chgrp user_name target-file**
    - Change owner: **sudo chown user_name dirname**
    - Make Sub-Files and Sub-Directories have same group as parent: chmod **g+s dirname**
  - Profile customization can be done through the following: **( .login, .cshrc, .tcsrhc, .bashrc, .bash_profile, .bash_login )**
  - **PATH=** > Tells the shell which directories to search for Unix Programs
  - **umask** > Sets the default file permissions assigned to all files you create
  - **Chapter 3 - Summary:** We have now gone over **file management**, **profile management** and the **PATH** variable. And we are now aware about network file sharing
- **Chapter 4 - File Management**
  - In Linux everything can be viewed as a file. Directories themselves are simply a special type of file
  - Removing files with spaces between them
    - ~~**rm a strange name**~~
    - **rm "a strange name"**
  - The primary Wildcards
    - **Asterisk :** *
    - **Question Mark :** ? | Covers single characters
    - **Square Brackets :** [ ] | Covers a range of single characters
  - The most popular text editors are **vim**, **emacs** & **Pico**
    - Text editors are made to work with plaintext files.
    - Unix makes us of plain-text files in various locations, from the Input and Output of Linux programs to shell setup files & shell scripting
  - Commands to find files in Unix
    - **find** -> simple subdirectory search
    - **locate** -> GNU program that requires setup
  - **Chapter 4 - Summary:** We interacted in various different ways with the Unix file-system, covering popular text-editors, and the powerful use-cases for wildcards
- **Chapter 5 - Redirecting I/O**
  - Covered the I/O operations [ > , >> , < , | , grep ]
    - > - Output Redirection Operator
    - >> - Append Redirection Operator
    - < - input redirection
    - | - Pipe Operator
    - **grep - Looks for patterns**
      - ☐ **-v** : lines that don't match pattern
      - ☐ **-n** : matched line and line #
      - ☐ **-l** : names of files with matching lines
      - ☐ **-c** : count of matching lines
      - ☐ **-i** : match either upper/lower case
    - **sort - Arranges lines of text numerically or alphabetically**
      - ☐ **-n** : sort numerically

- -r : reverse sorting order
- -f : sort upper- and lowercase together
- +x : ignore first x fields when sorting
- (+4n ) -rw-rw-r-- 1 user group 1605
- +4n **- skips over 4 fields, sep. by blank spaces**

- "> filename" - Programs output is diverted from the standard output to the named file. **Overwriting the previous output.**
  - Careful about **overwriting** target file!
  - Also known as **clobbered**
- ">> filename" - Programs output is diverted from the standard output to the named file. And **appends to the file** as opposed to overwriting it.

- **Pager programs** - less, more, pg
- **Chapter 5 - Summary:** Covered the various redirection operators and learned about the power of **piping** & **filtering**. Pager programs and Filter operators like **grep** and **sort** are a powerful tool that allow us to organize and access information more effectively

- **Chapter 6 - Using the Internet and Other Networks**
  - There are various ways to access remote computers. You could use telnet, ssh, rlogin and rsh. However, the main standard today is **ssh**. Though there may be times when you will need to use older protocols.
    - It is also possible to open windows through remote connections. This is known as X-Forwarding
  - There are programs out there called CLI Web browsers. The most popular of which are w3m and Lynx. These tools allow you to browse online resources through the command line interface without needing a GUI.
  - There exist CLI tools to transfer files between systems, for remote systems you will need a networked filesystem location,
  - scp & rcp
    - **scp** ( secure copy )
    - **rcp** ( remote copy )
    - **Syntax:** scp||rcp hostname:pathname
  - FTP
    - More flexible and secure than rcp but much less secure than scp
    - **Syntax:** hostname:ftp
    - **Commands**
      - **Copies files Local -> Remote**
        - put filename | mput filename
      - **Copies files Remote -> Local**
        - get filename | mget filename
      - prompt
      - cd pathname - change working dir. on remote pc
      - lcd pathname - change working dir. on local pc
      - dir - list remote pc's directory
      - binary - tells ftp to copy files w/o translation. Preserves, [ picture, sound or other date ]

        - **ascii** - Transfers plaintext files, transferring data if needed Covers [ MS <-> Unix Transfer, etc. ]
    - I've noted in the book overview that **"Learning the Unix OS"** would also serve as a brief history lesson on some old topics. And this is where you'll see a lot of that. Don't get me wrong Usenet is still around and quite popular in many circles. The same can be said for **IRC**, we also have **Matrix** as a new option. However, **talk** can be considered to no longer be in use although some distributions still ship with it. In regard to **Usenet**, **IRC** and **Matrix** I do recommend you look more into it. It's certainly quite interesting and could be worth your time.
    - **Electronic Mail**
        - You're able to send E-Mails in Unix & Linux through the shell prompt
        - One of the most popular programs to do this was made by Berkley
    - **Usenet News**
        - WW distributed system available to computers.
        - In order to read Usenet groups, you will need a news client
            - Popular readers: **slrn**, **nn**, **trn**
            - Usenet Groups are what came before forums
    - **Interactive chat**
        - Instant Messaging, in the days when AOL and Jabber took off
        - **talk** & **IRC**
- **Chapter 7 - Multitasking**
    - Unix is able to do many different jobs at once
        - However, the reality is that each processor can only execute one command at a time. So, the OS divides the processor's time between tasks quickly, so it looks as is everything is running at the same time.
        - This is also known as **multi-tasking**
    - In order to run a program in the background, add an **&** character at the end
        - The shell then assigns and displays a **Process ID (PID) number** for the program
        - To put several processes in the background, use the following **syntax:**
            - **(process1; process2) &**
    - With Linux you can specify command-line options for windows
    - **Checking on a Process**
        - **ps** - allows you to see how long a process has been running
        - **tty** - shows the name of the terminal where a process is running
        - **ps**
            - **Process ID (PID)**
                - A unique number assigned by Unix to the process
            - **Terminal name (TTY)**
                - The Unix name for the terminal from which the process was started
            - **Run time (TIME)**
                - The amount of computer time **(in minutes and seconds)** that the process has used
            - **Command (CMD)**
                - The name of the process
    - The **kill** command aborts a process

- **kill PID(s)**
- **sleep n**

## Required Reading

Our required reading for this section will be a mix of reviewing old commands with learning new ones. We begin by covering some particularly useful List and System/Network monitoring commands. For those I recommend using the **man** page in order to get more comfortable with using the **man** command. After that we'll cover some articles on the Digital Ocean and RedHat community blogs that cover some more useful commands.

Finally we end with a more philosophical article, technical ability is important but I encourage you to consider the abstract side as well. Primarily questions like: "Why is Unix & Linux designed this way?" "What the point of FOSS?" and musings of that nature.

**( Self-Guided )** - Look into the following commands and go through them by using the **man** command. Read the **Name**, **Synopsis**, **Description** and go over some of the **Options**, then try the commands out. For system memory commands I recommend paying attention to the **-h** flag which shows displays the output in a human readable format. After you read the command's man page if it's use-case or meaning is unknown to you, do a web-search for more information.

- **List commands**: **ls, lsusb, lscpu, lshw, lsipc, lslocks, lslogins, lsmem, lsns, lsof, lspci, lsusb**
- **System Monitoring Commands: top/htop, free, df, ps, pkill**
- **Network Utilities: ping, traceroute, dig, nslookup, host, hostname, arp, ifconfig, iwconfig, route**

**( Article ) Digital Ocean Community: Top 50+ Linux Commands by e...f** - By this point we have gone over most of the commands listed in the Digital Ocean article. I recommend reviewing any commands you're not confident in and pick up any that you may not know.

**( Article ) RedHat - Enable Sysadmin: Linux Commands: du... by Tyler Carrigan** - The **du** command is an excellent tool that will give you insight into how your storage is being used. This short article will go over the most used flags and how to make the most of them.

**( Article ) RedHat - Enable Sysadmin: How to Manage Linux Permissions... by Damon Garn** - Managing user permissions is an essential task for any sysadmin. User access control is an essential security feature meant to protect data by giving only certain users access to data and resources based on certain contexts. If you still feel unconfident about managing permissions focus on this post and make sure to understand **absolute/octal mode.**

**( Article ) Linux-Databook: The Unix and Linux Philosophy** by **dboth** - Every Operating System has it's own philosophy. To steal the quote listed in the post.

> An operating system, by its nature, embodies the philosophy of its creators..." **- Mike Gancarz**

Essentially any OS from Linux to Windows to macOS has their own way of doing things. It's Linux's philosophy that makes it so powerful in a capable users hand. As always I recommend you take away your own conclusions from this article. You may agree with much of it now and maybe come to disagree with much of it later and vice versa, even I don't agree with every tenet.

**( Article ) itsfoss: 21 Terminal Shortcuts by Sagar Sharma** - Lastly, we cover **Terminal Shortcuts**. They might not seem like such a big deal but getting the hang of a couple shortcuts can make a big difference for your efficiency and speed leading to substantial quality of life improvements.

## Recommended Reading

**( Video ) Unix and Linux History by Jon "maddog" Hall** - Jon Hall gives a great rundown on Linux and Unix history in the 50 year period from 1969 to 2019. It's an informal recount mixed with personal experiences and that's exactly what makes it such a compelling watch.

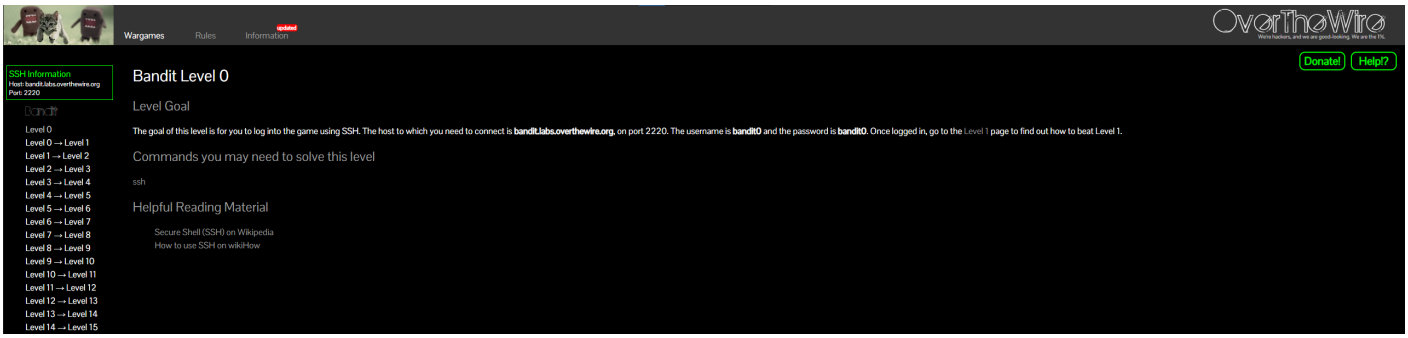
**( Video ) The Rise of Unix. The Seeds of its Fall. by Asianometry** - A great brief 16-minute video that covers the rise of Unix. Started in the legendary Bell Labs by Ken Thompson and Dennis Ritchie. As a personal note, if you're interested in video essays on technology, economics, business and politics I would highly recommend you explore more of Asianometry's videos.

**( Article ) The Early Days of Linux by Lars Wirzenius** - In keeping with the theme of historical reading the following article goes over some of the very first days of Linux. Lars witnessed the birth of Linux and he also helped fix some of its first issues.

**( Short Rant ) How does the Unix Philosophy matter in modern times? by dlarge65 -** There are many diverging opinions on the Unix philosophy. Some say it is over-idealized and irrelevant in the modern day, others argue that there was maybe a philosophy at one point but practicality won over and so on. While I'm still very much new to Linux I enjoyed this rant which takes on a more balanced stance and viewpoint on the Unix Philosophy. It is too early to give my thoughts on the matter but moving forward I plan on taking the balanced approach.

# Section Projects

### Project 2 - Your First Wargame -> Over The Wire: Bandit




**Wargame vs CTFs**
If you're like me, you probably thought that the term wargame and CTF were interchangeable but they are 2 different things although they are related.
To Summarize: **Capture The Flag (CTF)** - **Time Limited** / **Wargame** - **Not Time Limited**
**Capture The Flag (CTF)** - CTFs are exercises where participants try to find strings, called "flags" which are hidden inside purposefully vulnerable programs or websites,
**Wargame** - A cyber-security challenge where competitors must exploit or defend a vulnerability in a system or application to gain or prevent access to a computer system.

In essence CTF's are time-based competitions where people try to come out on top, whereas wargames aren't really time limited or competitions, although they do have CTF style challenges.

> For this project your job is to go through **The Beginner Wargame - Bandit**

**Why a wargame?**
First off, they are very fun! I was always a bit hesitant to read through man pages, but I found myself going through documentation and scouring man pages all to get that next flag.
Secondly, they can be very challenging. You'll be forced outside of your comfort zone quite often. Bandit will lead you to explore  concepts in networking, compression, decompression, hex dumps, the shell and more.

**Bandit**
Bandit is a wargame aimed at beginners. Wargames are usually split into levels. We start at Level 0 and try to reach the end.

**Why Bandit?**
I was pretty hesitant about doing CTFs, they tend to have a heavy cybersecurity focus, which is important, but at our stage can be a distraction.
However, **bandit is the ultimate Linux Lab.**
- You will use a variety of new flags for commands you commonly use
- You will get a better grasp on user and file permissions
- You will become more comfortable google searching for Linux information
- You will be forced to read the man pages
- You will cover Networking Topics, Data encoding, compression and decompression
- You will be introduced to git, and its commands
- You will go over shell and CLI behavior

After doing Bandit, I am much more comfortable reading documentation, and I have a better idea on the skills I'm missing as a Linux user. You'll learn a lot doing your first wargame and I hope you'll have as much fun working through it as I did.

## Bandit Level - 0

Revision #23
Created 25 November 2023 02:03:52 by 01Blu3
Updated 26 April 2024 16:37:10 by 01Blu3