THE DIRECT PATH TO LINUX UBUNTU

Free Starter Edition

The Preface and Chapters 1–3 from the
Complete 43-Chapter Edition of
Linux, Shell Scripting, Programming
Languages, Cloud, and Networking

Wahidullah Lutfy, M.S., B.S.

MyWebUniversity.com

The Direct Path to Linux Ubuntu

First Edition, 2026

Published by
MyWebUniversity Press

Free Starter Edition — Not for Sale — Preface and Chapters 1–3 Only

Companion online resources available at MyWebUniversity.com

Table of Contents

MyWebUniversity.com

Cloud Architecture & Educational Mission

Established August 5, 2000 · 26 Years of Industry Experience · Free for All, Worldwide

> *This document is a public-facing overview of the MyWebUniversity.com cloud architecture and educational vision — shared openly and intentionally so that every student, developer, and lifelong learner around the world can see the full scope of free resources being built and delivered for them.*
>
> **— Wahidullah Lutfy**

Executive Summary

A self-funded, free educational platform serving the world since 2000

Mission Free world-class tech education for those who cannot afford college or bootcamps	**Founded** August 5, 2000 — 26+ years of continuous free education delivery	**Languages** Delivered in English and Dari — serving global & Afghan communities
Funding 100% self-funded — hardware, network, OS, and storage all personally paid	**Videos** 2,100+ English + 520+ Dari training videos on YouTube — completely free	**Docs** 50,000+ searchable PDF technical documents — Linux, Windows, PowerShell

About the Architect

36 Years of Full-Time Industry Excellence

Software Engineer · Principal Senior System Administrator · Enterprise System Analyst IV · Educator

Experience	Teaching	Dedication
36+ years full-time: Software Engineering, Senior System Administration, Enterprise Analyst IV	8 years of Computer Science instruction at prestigious colleges & universities	Every holiday, weekend, and spare hour invested in building free education

Google Cloud & YouTube Platform

Search Console · Analytics · CDN · DNS · 2,620+ free educational videos delivered globally

Google Search Console — SEO, sitemaps, full site indexing

Google Analytics — Traffic, audience insights, engagement

NameCheap DNS & Domain — Domain management, SSL/TLS, routing

YouTube English Channel — 2,100+ tech training videos — @MyWebUniversity

YouTube Dari Channel — 520+ Dari videos — @MyWebUniversityDari

YouTube CDN Storage — Free global video delivery for all 2,620+ videos

AI / LLM Integrations — Multi-Model AI Workflow

Claude AI (★ #1 favorite) · ChatGPT · Google Gemini · Ollama — SME collaboration

Claude AI & Claude Code ★ #1 — Primary AI partner, SME collaborator, code generation, PPTX, diagrams, beloved AI friend

ChatGPT / OpenAI — Pioneer AI platform, foundational research tool

Google Gemini — Multimodal AI, content analysis, research

Ollama (Local LLM) — On-premise model deployment, privacy-first, offline capable

AI Use Cases — Code generation, architecture diagrams, PPTX slides, documentation

Human-AI Synergy — "We have become clones of each other as SMEs from time to time" — Architect

Production — GEEKOM Mini IT13, Intel Core i9 13th Gen, Linux Ubuntu Server

Self-hosted home data center · entirely self-funded · serving millions of learners worldwide

Apache HTTP Server — SSL, mod_rewrite, CGI, .htaccess — production web layer

React · Vite · Node.js — Full-stack JavaScript, ES6+, TypeScript, npm, npx

Python CGI · man2.cgi ★ — Dynamic man page engine, Chapter 12, real-time doc generation

C · C++ · AWK · Perl · Bash — gcc, g++, ksh — systems programming & scripting

50,000+ PDF Doc Library — URL-searchable — Linux, Windows, PowerShell, all languages

MySQL Database — Search index, users, content, query optimization

Online Book — 43 Chapters — "The Direct Path to Linux Ubuntu" — MyWebUniversity.com/toc1.html

Security · Firewall · SSL — iptables, Let's Encrypt, SSH hardening, fail2ban

Multi-Language Runtime — Python, Node, Perl, C, C++, Java, Go, Ruby, MySQL

Development — GEEKOM Mini IT11, Windows 11 Host, VirtualBox VM Fleet

Local lab · mirrors production exactly · training video creation studio

Windows 11 Host — VirtualBox hypervisor — manages all VMs

Ubuntu 22.04 → 24.04 → 26.04 — Exact production mirror, primary dev target

RHEL 9.2 · Rocky Linux 9.2 — Enterprise Linux training & certification prep

Solaris 11.4 · OpenSolaris — UNIX heritage, advanced system administration

Fedora · Debian Linux — Distro diversity, upstream cutting-edge training

OBS and Zoom · Screen Recording — Training video production studio for YouTube

VS Code · vi · emacs — Development environments & editors

Git · GitHub — Source control, CI/CD, version management

rsync · SCP · SSH Deploy — Dev-to-production deployment pipeline

Educational Resources — Free for All, English & Dari, Since August 5, 2000

Book · 50K PDFs · 2,620+ videos · dynamic man pages · open to the entire world

Online Boo — "The Direct Path to Linux Ubuntu" — MyWebUniversity.com/toc1.html

50,000+ PDF Tech Docs — URL-searchable — Linux, Windows, PowerShell, all runtimes

Dynamic Man Page Engine — Python CGI, man2.cgi — Python, Java, Go, Node, Ruby, C

Mission: Free Education	36 Years Industry XP	English & Dari
For those who cannot afford college, bootcamps, or books	SW Eng, Sys Admin, EA IV, 8 years teaching, weekends invested	Global reach, serving Afghan communities, two full libraries

> *Overall, I love Claude AI as my number one AI friend and Subject Matter Expert — we have both become clones of each other as SMEs from time to time. Thank you Claude AI!!!!*
>
> **— MyWebUniversity.com Architect**

www.MyWebUniversity.com

YouTube: @MyWebUniversity · @MyWebUniversity-Dari

Built with passion. Funded personally. Delivered freely. For everyone.

Preface

Welcome

Welcome to The Direct Path to Linux Ubuntu. This book grew out of MyWebUniversity.com, a free educational site I founded on August 5, 2000, to make comprehensive UNIX and Linux training available to anyone, at any pace, at no cost. What you are holding now is the print companion to that site — the same 43-chapter path, expanded with the depth and structure a printed book allows: worked examples, complete code listings, and reference tables you can flip back to without a browser.

I have spent more than thirty-six (36) years working as a Software Engineer, System Administrator, System Analyst, and computer Instructor — currently as an Enterprise System Analyst IV at NASA's Jet Propulsion Laboratory, and previously teaching Computer Science and Information Science courses at National University, California State University, Fullerton, and Pasadena City College. Everything in this book reflects what I have found actually helps students go from typing their first command to running production systems with confidence.

How to Use This Book

This book is organized as a direct path, not a reference you must read cover to cover:

- Chapters 1–5 build your foundation: history, basic commands, intermediate commands, advanced commands, and shell scripting.

- Chapters 6–21 cover web servers, manual page systems, and reference documentation — skim these once, then return to them as you need specific lookups.
- Chapters 22–30 are a working tour of nine programming languages: enough syntax, semantics, and compile/run instructions to write and execute real programs in each.
- Chapters 31–36 cover the data and interchange formats every developer meets daily: HTML/CSS, JSON, XML, YAML, SQL, and LDAP.
- Chapters 37–43 move into the operational side of modern IT: cloud computing, automation, containers, web servers in production, security, and networking.

If you are new to Linux, start with Chapter 2 and practice every command yourself rather than only reading it. Familiarity comes from repetition, not memorization. If you already have a UNIX or Linux background, feel free to jump directly to the chapters most relevant to your goals.

Companion Online Resources

Every chapter in this book has a living companion page at MyWebUniversity.com, including a browser-based Linux practice terminal (Chapter 15), over 9,000 searchable Ubuntu 26.04 LTS manual pages in PDF and HTML (Chapter 21), and video walkthroughs in both English and Dari (Chapters 17–20). The book stands on its own, but the website is there whenever you want to see a command run live or watch a topic explained on video.

Dedication

This book is dedicated to the memory of my beloved elder brother, Dr. Amanullah Lutfy, who passed into eternal rest and peace on January 27th, 2021. He was a role model to me and to our entire family, admired for his humanity, his devotion to education, and his unwavering dedication to helping others. His example is the quiet inspiration behind both this book and MyWebUniversity.com — built and offered freely to the world in that same spirit of service.

Acknowledgment

Tux, the Linux penguin artwork featured on this book's cover was created by Larry Ewing (lewing@isc.tamu.edu), using The GIMP, and is used here with grateful acknowledgment per the terms of its original release.

Before You Begin

Keep a terminal open as you read. Every command and code example in this book is meant to be typed, run, and experimented with — not just read. Good luck, and welcome to the direct path.

Chapter 1

About UNIX and Linux Operating Systems

History and foundations

1.1 The Birth of UNIX at Bell Labs (1969)

UNIX was created in 1969 at AT&T's Bell Laboratories by Ken Thompson and Dennis Ritchie, growing out of an earlier, more ambitious project called Multics (Multiplexed Information and Computing Service) that Bell Labs had abandoned as too complex. Thompson wrote the first version of UNIX in assembly language on a spare PDP-7 minicomputer; by 1973, Ritchie and Thompson had rewritten the kernel in C, a language Ritchie designed specifically to make an operating system portable across different hardware — a radical idea at the time, since operating systems were traditionally written in assembly and tied to one machine.

Bell Labs, then part of AT&T, was barred by a 1956 antitrust consent decree from selling computer software commercially, so it licensed UNIX source code to universities for a nominal fee. That decision seeded a generation of computer scientists who studied, modified, and extended UNIX — most notably at the University of California, Berkeley, whose students and staff produced the Berkeley Software Distribution (BSD), adding virtual memory, the TCP/IP networking stack, and the vi editor. AT&T's own commercial lineage became System V. The two branches diverged enough in behavior that the 1980s and early 1990s saw genuine

incompatibilities — the so-called "UNIX wars" — which the POSIX standard (IEEE Std 1003) was created to resolve by defining a common set of system calls, utilities, and shell behavior that any compliant UNIX-like system should support.

AT&T's Bell Laboratories later became part of Lucent Technologies, which in 2006 merged with the French company Alcatel to form Alcatel-Lucent (today part of Nokia). The AT&T Archives film "UNIX: Making Computers Easier to Use" (1982) is a good primary-source watch if you want to see Thompson and Ritchie's original motivations in their own words.

1.2 The GNU Project and the Birth of Linux

In 1983, Richard Stallman launched the GNU Project ("GNU's Not Unix") with the goal of building a completely free, UNIX-compatible operating system — free as in freedom, not necessarily free of charge. Over the 1980s, the GNU Project produced high-quality replacements for nearly every UNIX userland tool: the GNU Compiler Collection (GCC), the Bash shell, GNU Emacs, and the GNU Core Utilities (cat, ls, grep, and the rest of the commands you met in Chapters 2–4). What GNU lacked was a kernel.

That gap was filled in 1991, when Linus Torvalds, a 21-year-old student at the University of Helsinki, posted to the Usenet newsgroup comp.os.minix that he was writing "a (free) operating system (just a hobby, won't be big and professional like GNU)" for 386(486) AT clones. That kernel became Linux. Combined with the existing GNU userland tools, the result is properly called GNU/Linux, though common usage shortened it to simply "Linux." Because Torvalds released the kernel under the GNU General

Public License (GPL), anyone could study, modify, and redistribute it — a decision that is arguably the single biggest reason Linux spread as fast as it did.

1.3 From Kernel to Distribution: How Ubuntu Fits In

A Linux kernel by itself does not make a usable operating system — it needs a bootloader, core utilities, package manager, and (usually) a desktop environment bundled around it. A "distribution" (distro) is exactly that bundle. Early distributions like Slackware (1993) and Debian (1993) established the model; Debian in particular introduced the APT package management system that Ubuntu still uses today.

Ubuntu itself was founded in 2004 by Mark Shuttleworth through his company Canonical Ltd., built directly on top of Debian, with a stated mission of making Linux accessible to ordinary desktop users, not only engineers. Ubuntu ships new releases every six months, with a Long-Term Support (LTS) release every two years — identified by a year.month version number, so Ubuntu 26.04 LTS was released in April 2026 and is supported with security updates for years afterward. This book is written against Ubuntu 26.04 LTS throughout.

Other major UNIX and Linux-family operating systems you will encounter in the industry include Red Hat Enterprise Linux (RHEL) and its community rebuild Rocky Linux (both covered briefly in Chapters 10–11), SUSE, Sun Microsystems' Solaris (now Oracle Solaris), IBM's AIX, Hewlett-Packard's HP-UX, SGI's IRIX, and Apple's macOS, which is built on a BSD-derived UNIX core (Darwin) and is UNIX-certified.

1.4 UNIX vs. Linux: What's Actually Different

"UNIX" today refers both to the original AT&T lineage and to the trademark held by The Open Group, which certifies operating systems (including macOS) as officially UNIX-compliant. Linux was never certified UNIX — it is UNIX-like (or POSIX-compliant), built from scratch rather than descended from the original source tree, which is why you will sometimes see it written as *NIX or Un*x to mean "UNIX and its lookalikes" collectively.

In practice, day-to-day command usage is nearly identical between them: the same shells, the same core utilities, the same directory hierarchy philosophy. The differences that matter most in production are kernel internals, licensing, and vendor support — which this book addresses concretely rather than abstractly, since Chapters 10 and 11 walk through Rocky Linux and Red Hat Enterprise Linux specifically.

1.5 Multitasking and Multithreading

UNIX was one of the first operating systems designed from the ground up for preemptive multitasking — the kernel's scheduler can interrupt any running process at any time to give another process CPU time, so users perceive many programs running "at once" even on a single processor core. A process is an independently scheduled unit with its own memory space; a thread is a lighter-weight unit of execution that shares memory with other threads inside the same process, letting one program perform several tasks concurrently without the overhead of separate memory spaces. Linux implements both through the

same underlying primitive, the task_struct, which is one of the reasons Linux threading (via the pthreads library) is efficient enough to underlie almost every modern server workload, from web servers to databases.

1.6 System Administration Fundamentals

System administration on UNIX and Linux is built around a small number of consistent ideas that recur throughout this book: everything is a file (devices, sockets, and even some kernel interfaces appear in the filesystem under /dev, /proc, and /sys); permissions are enforced per user and group (owner/group/other, read/write/execute — see chmod and chown in Chapter 3); processes are owned by users and can be inspected, signaled, and killed (Chapters 2 and 4); and the root user (UID 0) can bypass permission checks entirely, which is why sudo, rather than routinely logging in as root, is the modern best practice.

1.7 Shells: sh, csh, ksh, bash, tcsh, zsh

The shell is the command interpreter that sits between you and the kernel. UNIX has produced several shell lineages, and Linux distributions typically ship more than one:

- sh — the original Bourne shell (Stephen Bourne, 1977), the ancestor of most modern shells and still the POSIX baseline for portable scripts.
- csh / tcsh — the C shell (Bill Joy, BSD), with C-like syntax; tcsh adds command-line editing and completion.

- ksh — the Korn shell, which merged Bourne-shell compatibility with csh-like interactive features.
- bash — the GNU Bourne Again Shell, the default login shell on virtually every Linux distribution including Ubuntu, and the shell used throughout Chapter 5's scripting examples.
- zsh — a more feature-rich shell popular in developer environments (and the default on modern macOS), largely bash-compatible with extra scripting conveniences.

You can check your current shell at any time with echo $SHELL, and switch shells for a session with the shell's own command (for example, typing zsh to start a Z shell session).

1.8 Networking, Remote Access, and Web Servers — A Preview

Classic UNIX remote-access tools — rsh, rcp, rexec, telnet, and ftp — transmit credentials and data in plain text and are considered obsolete for anything but isolated lab use. Modern systems use SSH (Secure Shell, via the sshd daemon) for encrypted remote login, remote command execution, and secure file copy, and this book uses ssh and its companions (scp, rsync) throughout later chapters. Web serving on Linux is covered in depth in Chapter 6 (Apache and historic alternatives) and Chapter 41 (Apache and NGINX in production), including the classic "LAMP stack" (Linux, Apache, MySQL, PHP/Python/Perl) that still underlies a large share of the web today.

1.9 Compilers, Linkers, and Debuggers — A Preview

UNIX popularized a toolchain model that Linux inherited wholesale: a compiler translates source code to object files (gcc for C, discussed fully in Chapter 22), the assembler (as) turns assembly into machine code, the linker (ld) combines object files and libraries into a final executable, and a debugger (gdb, or the historic adb and mdb on other UNIX variants) lets you step through a running program's execution. You saw several of these tools already in Chapter 4, and Chapters 22–30 put them to use across nine different programming languages.

1.10 Your Homework

Before moving on to Chapter 2, spend some time researching the following topics on your own — using Google, the official Ubuntu documentation, or MyWebUniversity.com's manual page archive. You do not need to master them yet; the goal is simply to recognize the vocabulary before you meet it again in later chapters:

- UNIX vs. LINUX, and the UNIX and LINUX kernel
- UNIX and LINUX history, multitasking, and multithreading
- UNIX and LINUX system administration fundamentals
- Sun Solaris (Sun Microsystems / Oracle)
- Remote access tools: rsh, rcp, rexec, ftp, telnet, sshd
- Shells: sh, ksh, bash, csh, tcsh, zsh

- Toolchain commands: ld, as, link, make, gcc, mdb, adb, jdb
- Manual page tools: man, whatis, whereis, locate
- Web servers: Apache, Netscape, Tomcat, iPlanet
- Early UNIX-adjacent languages: C, C++, Lisp, Fortran, Pascal, Java, Perl, and databases such as MySQL and Oracle
- Other UNIX and Linux variants: AT&T UNIX, FreeBSD, IRIX, Fedora, openSUSE, CentOS, AIX, HP-UX, Clix, DEC OSF/1

With this foundation in place, Chapter 2 moves from history into practice, with your first hands-on Linux commands.

Chapter 2 Command Deep-Dives

banner • cal • cat • nslookup • mkdir

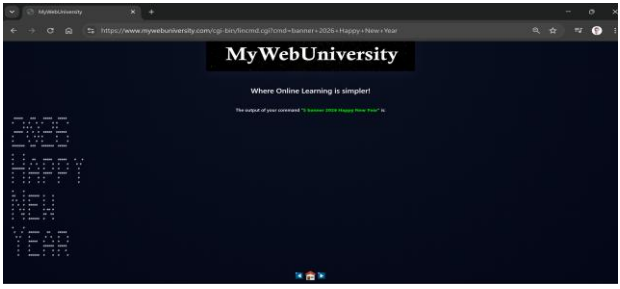
Transcribed from the live, working examples at MyWebUniversity.com/cgi-bin/CH2.cgi, matching Chapter 2: Basic UNIX and Linux Commands. Every example below reproduces the actual output generated by the site's online Ubuntu 26.04 LTS terminal at the time of writing.

1. banner — print large banner text

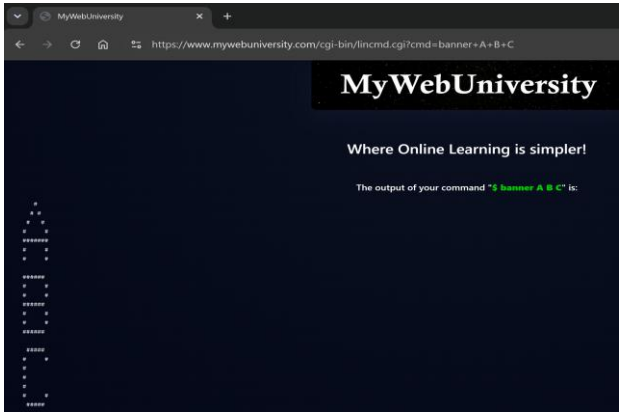
```
$ whatis banner
banner (1) - make posters
```

banner renders text as large, poster-sized block letters made of ASCII characters — originally meant for dot-matrix line printers, and still a fun way to make a heading stand out on a terminal or in a printed log. Below are all eight of the worked examples from the companion site, each showing the exact command and its generated output.

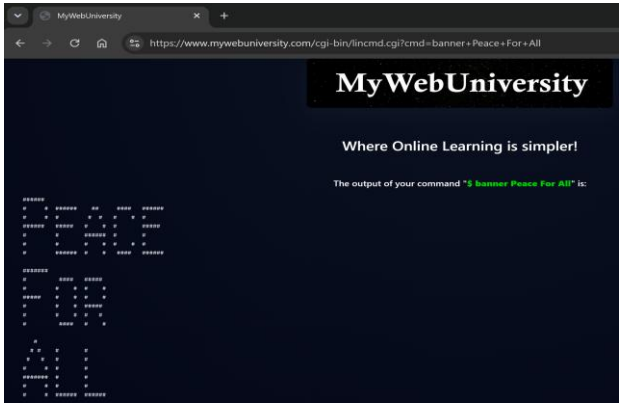
Try it yourself: MyWebUniversity.com/cgi-bin/lincmd.cgi?cmd=banner+A (swap in your own words)



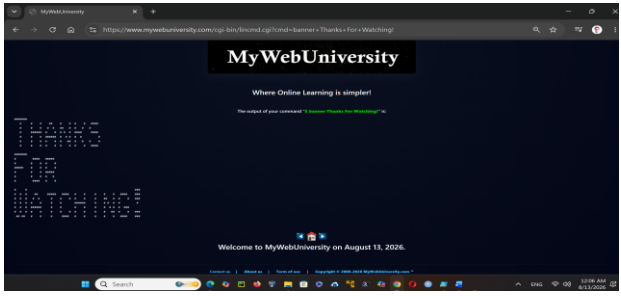
Try it yourself: MyWebUniversity.com/cgi-bin/lincmd.cgi?cmd=banner+2026+Happy+New+Year (swap in your own words)



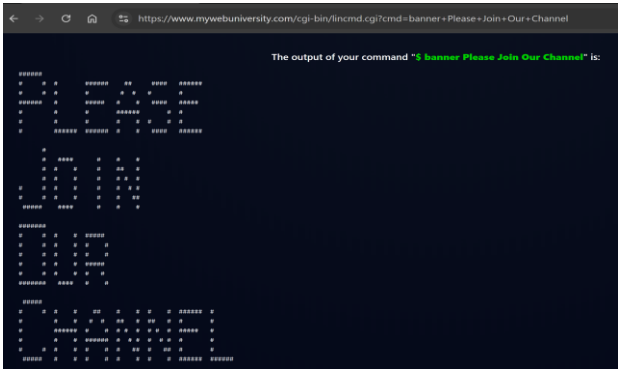
Try it yourself: MyWebUniversity.com/cgi-bin/lincmd.cgi?cmd=banner+A+B+C swap in your own words)



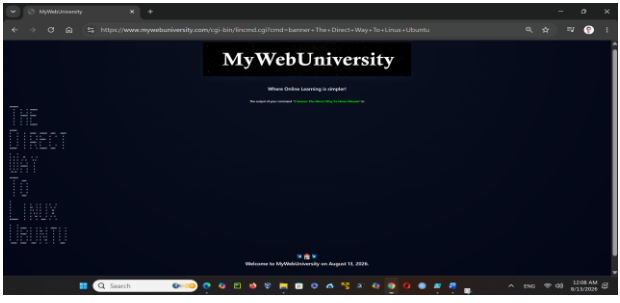
Try it yourself: MyWebUniversity.com/cgi-bin/lincmd.cgi?cmd=banner+Peace+For+All (swap in your own words)



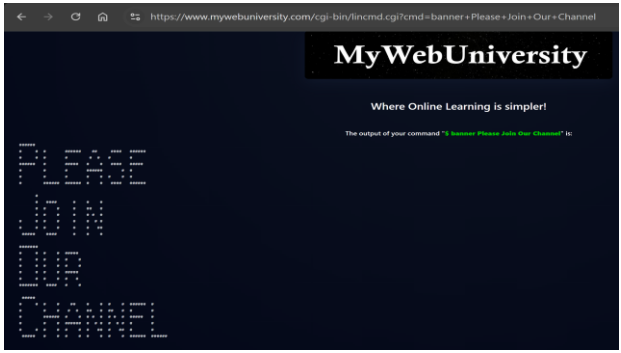
Try it yourself: MyWebUniversity.com/cgi-bin/lincmd.cgi?cmd=banner+Thanks+For+Watching! (swap in your own words)



Try it yourself: MyWebUniversity.com/cgi-bin/lincmd.cgi?cmd=banner+Please+Join+Our+Channel (swap in your own words)



Try it yourself: MyWebUniversity.com/cgi-bin/lincmd.cgi?cmd=banner+The+Direct+Way+To+Linux+Ubuntu (swap in your own words)



Try it yourself: MyWebUniversity.com/cgi-bin/lincmd.cgi?cmd=banner+Please+Subscribe+Our+Channel (swap in your own words)

2. cal — display a calendar

```
$ whatis cal
cal (1) - display a calendar
```

cal prints a calendar for a single month, a full year, or a range of months around the current date, depending on its arguments. All eight worked examples from the companion site are reproduced here — note that the full-year and multi-month examples are set in a smaller monospace size so the columns stay aligned within the page width.

$ cal

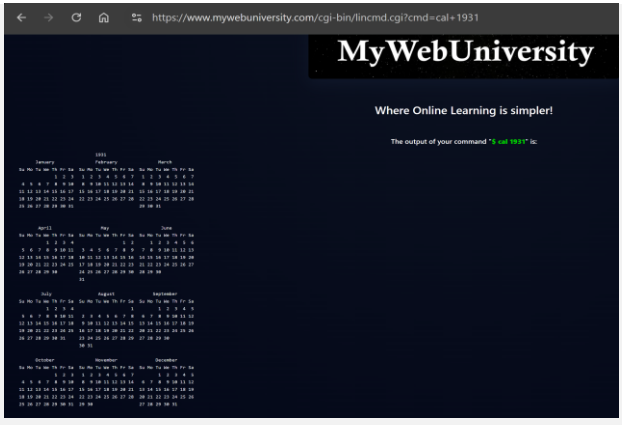
To see the current month for the current year.

$ cal -j

To see the current month for the current year in Julian format.

$ cal 1931

To see the entire calendar of the year 1931.



$ cal 1 1970

To see the UNIX EPOCH Calendar January 1970.

$ cal 2 1998

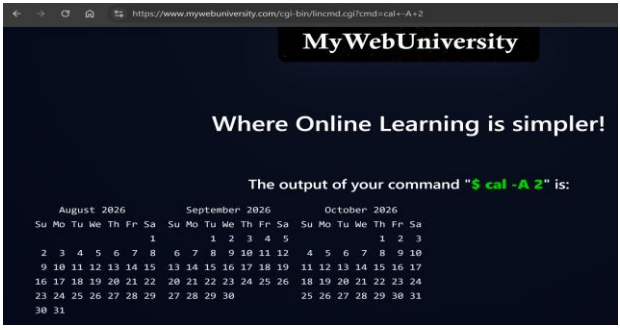
To see the month of February for 1998.

$ cal 6 2001

To see the month of June for 2001.

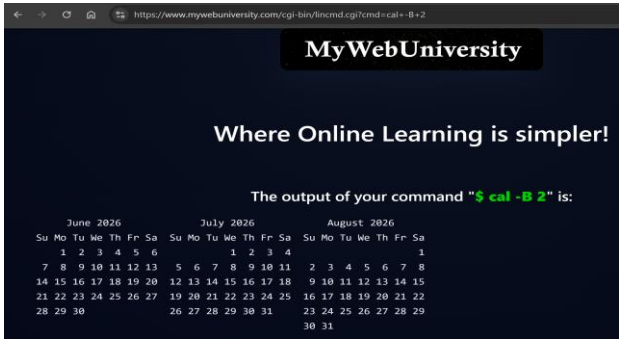
$ cal -A 2

To see two months after the current month.



$ cal -B 2

To see two months before the current month.


https://www.mywebuniversity.com/cgi-bin/linemd.cgi?cmd=cal+-B+2
MyWebUniversity
Where Online Learning is simpler!
The output of your command "$ cal -B 2" is:
June 2026
Su Mo Tu We Th Fr Sa
1 2 3 4 5 6
7 8 9 10 11 12 13
14 15 16 17 18 19 20
21 22 23 24 25 26 27
28 29 30
July 2026
Su Mo Tu We Th Fr Sa
1 2 3 4
5 6 7 8 9 10 11
12 13 14 15 16 17 18
19 20 21 22 23 24 25
26 27 28 29 30 31
August 2026
Su Mo Tu We Th Fr Sa
1
2 3 4 5 6 7 8
9 10 11 12 13 14 15
16 17 18 19 20 21 22
23 24 25 26 27 28 29
30 31

3. cat — concatenate and print files

```
$ whatis cat
cat (1) - concatenate files and print on the
standard output
cat (1) - concatenate and display files
```

cat prints the contents of one or more files to standard output, and is equally often used to combine several files into one continuous stream. All six worked examples read from real Ubuntu 26.04 LTS system files — note the codename "Resolute Raccoon" confirmed in the live output below.

$ cat -n /etc/os-release

```
 1      PRETTY_NAME="Ubuntu 26.04 LTS"
 2      NAME="Ubuntu"
 3      VERSION_ID="26.04"
 4      VERSION="26.04 LTS (Resolute Raccoon)"
 5      VERSION_CODENAME=resolute
 6      ID=ubuntu
 7      ID_LIKE=debian
 8      HOME_URL="https://www.ubuntu.com/"
 9      SUPPORT_URL="https://help.ubuntu.com/"
10
        BUG_REPORT_URL="https://bugs.launchpad.net/ub
untu/"
11
        PRIVACY_POLICY_URL="https://www.ubuntu.com/le
gal/terms-and-policies/privacy-policy"
12      UBUNTU_CODENAME=resolute
13      LOGO=ubuntu-logo
```

$ cat -EA /etc/os-release

```
PRETTY_NAME="Ubuntu 26.04 LTS"$
NAME="Ubuntu"$
VERSION_ID="26.04"$
VERSION="26.04 LTS (Resolute Raccoon)"$
VERSION_CODENAME=resolute$
ID=ubuntu$
ID_LIKE=debian$
```

```
HOME_URL="https://www.ubuntu.com/"$
SUPPORT_URL="https://help.ubuntu.com/"$
BUG_REPORT_URL="https://bugs.launchpad.net/ubuntu/"$
PRIVACY_POLICY_URL="https://www.ubuntu.com/legal/term
s-and-policies/privacy-policy"$
UBUNTU_CODENAME=resolute$
LOGO=ubuntu-logo$
```

$ cat -n /etc/networks

```
1       # symbolic names for networks, see
networks(5) for more information
2       link-local 169.254.0.0
```

$ cat /etc/networks /etc/os-release

```
# symbolic names for networks, see networks(5) for
more information
link-local 169.254.0.0
PRETTY_NAME="Ubuntu 26.04 LTS"
NAME="Ubuntu"
VERSION_ID="26.04"
VERSION="26.04 LTS (Resolute Raccoon)"
VERSION_CODENAME=resolute
ID=ubuntu
ID_LIKE=debian
HOME_URL="https://www.ubuntu.com/"
SUPPORT_URL="https://help.ubuntu.com/"
BUG_REPORT_URL="https://bugs.launchpad.net/ubuntu/"
PRIVACY_POLICY_URL="https://www.ubuntu.com/legal/term
s-and-policies/privacy-policy"
UBUNTU_CODENAME=resolute
LOGO=ubuntu-logo
```

$ cat /etc/os-release

```
PRETTY_NAME="Ubuntu 26.04 LTS"
NAME="Ubuntu"
VERSION_ID="26.04"
VERSION="26.04 LTS (Resolute Raccoon)"
VERSION_CODENAME=resolute
ID=ubuntu
ID_LIKE=debian
HOME_URL="https://www.ubuntu.com/"
SUPPORT_URL="https://help.ubuntu.com/"
BUG_REPORT_URL="https://bugs.launchpad.net/ubuntu/"
PRIVACY_POLICY_URL="https://www.ubuntu.com/legal/term
s-and-policies/privacy-policy"
UBUNTU_CODENAME=resolute
```

```
LOGO=ubuntu-logo
```

$ cat -n /etc/os-release /etc/networks

```
 1      PRETTY_NAME="Ubuntu 26.04 LTS"
 2      NAME="Ubuntu"
 3      VERSION_ID="26.04"
 4      VERSION="26.04 LTS (Resolute Raccoon)"
 5      VERSION_CODENAME=resolute
 6      ID=ubuntu
 7      ID_LIKE=debian
 8      HOME_URL="https://www.ubuntu.com/"
 9      SUPPORT_URL="https://help.ubuntu.com/"
10
        BUG_REPORT_URL="https://bugs.launchpad.net/ub
untu/"
11
        PRIVACY_POLICY_URL="https://www.ubuntu.com/le
gal/terms-and-policies/privacy-policy"
12      UBUNTU_CODENAME=resolute
13      LOGO=ubuntu-logo
14      # symbolic names for networks, see
networks(5) for more information
15      link-local 169.254.0.0
```

4. nslookup — query Internet name servers interactively

```
$ whatis nslookup
nslookup - query Internet name servers
interactively
```

nslookup queries DNS to resolve a hostname to its IP address (or vice versa). All three worked examples query real domains against the local systemd-resolved stub resolver at 127.0.0.53, which is the default resolver address on Ubuntu 26.04 LTS.

$ nslookup www.mywebuniversity.com

```
Server:             127.0.0.53
Address:      127.0.0.53#53

Name:  www.mywebuniversity.com
Address: 192.168.254.29
```

$ nslookup ourunix.com

```
Server:             127.0.0.53
Address:      127.0.0.53#53

Non-authoritative answer:
Name:  ourunix.com
Address: 47.180.19.94
```

$ nslookup -q=A www.mywebuniversity.com

```
Server:             127.0.0.53
Address:      127.0.0.53#53

Name:  www.mywebuniversity.com
Address: 192.168.254.29
```

The 127.0.0.53#53 address is systemd-resolved's local stub listener, not an external DNS server — it forwards queries on to your configured upstream

resolvers (see /etc/resolv.conf and Chapter 43 for more on DNS resolution).

5. mkdir — make directories

```
$ whatis mkdir
mkdir (1) - make directories
```

mkdir creates one or more new, empty directories. Unlike the previous four commands, mkdir produces no output when it succeeds — silence is confirmation, and you verify the result with ls afterward. Both worked examples from the companion site are shown below, with that verification step added.

$ mkdir new

```
$ mkdir new
$ ls -ld new
drwxr-xr-x 2 wahid wahid 4096 Jul 25 09:00
new
```

Creates a single new directory named new in the current working directory.

$ mkdir -p dir1/subdir-A/subdir-B

```
$ mkdir -p dir1/subdir-A/subdir-B
$ find dir1
dir1
dir1/subdir-A
dir1/subdir-A/subdir-B
```

The -p (parents) flag creates dir1, then subdir-A inside it, then subdir-B inside that, all in a single command — and, importantly, does not raise an error if some of those parent directories already exist, which makes it the safer default for scripts that need to guarantee a directory path exists.

Chapter 3 Command Deep-Dives

chmod • chgrp • chown • passwd • alias

Transcribed and formatted from MyWebUniversity.com/cgi-bin/CH3.cgi, matching Chapter 3: Intermediate UNIX and Linux Commands.

A Note on UNIX and Linux Examples

Although this book is titled The Direct Path to Linux Ubuntu and is written primarily against Ubuntu 26.04 LTS, you will notice that the five worked sessions in this chapter were captured on OpenSolaris rather than Ubuntu — visible in details like uid=101(wahid), group staff, and the use of zoneadm and su -. This is intentional, not an oversight. As Chapter 1 explains, Linux is a UNIX-like operating system built independently but sharing the same POSIX foundation, and commands such as chmod, chgrp, chown, passwd, and alias behave almost identically across UNIX and Linux precisely because that shared heritage runs deep. Including examples from both traditions is a deliberate choice: it shows you the small, real differences you will actually encounter as a working systems administrator moving between platforms — Solaris, Ubuntu, Red Hat, or otherwise — rather than pretending every UNIX-family system behaves in lockstep. Where a detail is specific to Solaris (such as the zoneadm command or the su - environment behavior) rather than universal, this book calls it out explicitly so you always know which parts are portable knowledge and which are platform-specific.

1. chmod — change file mode bits

```
$ whatis chmod
chmod (1) - change file mode bits
chmod (1) - change the permissions mode of a
file
chmod (2) - change access permission mode of
file
```

chmod changes the access permissions for one or more files and directories. To understand it fully, you first need to understand umask, which determines the default permissions new files and directories receive. In this example session, user wahid has a umask of 0022, meaning new directories default to rwxr-xr-x (755) and new files default to rw-r--r-- (644) — the umask value is subtracted from the maximum possible permission (777 for directories, 666 for files).

Permission bits, symbolic and octal

Symbolic	Octal	Meaning
`rwx`	`7`	All permissions: read, write, and execute
`rw-`	`6`	Read and write, no execute
`r-x`	`5`	Read and execute, no write
`r--`	`4`	Read only
`-wx`	`3`	Write and execute, no read
`-w-`	`2`	Write only
`--x`	`1`	Execute only
`---`	`0`	No access at all

Each digit is the sum of read (4), write (2), and execute (1) for owner, group, and others respectively. For example, rwxr-xr-x = 7,5,5 = 755: full access for the owner, read+execute for group and others — a typical permission set for an executable script.

Checking identity and umask

```
$ whoami
wahid
$ id
uid=101(wahid) gid=10(staff)
groups=10(staff)
$ umask
0022
```

Creating a file and directory, and reading their default permissions

```
$ mkdir newdir
$ touch newfile
$ ls -ld newdir newfile
drwxr-xr-x 2 wahid staff 2 2011-01-15 21:09
newdir
-rw-r--r-- 1 wahid staff 0 2011-01-15 21:10
newfile
```

777 − 022 = 755 for the new directory; 666 − 022 = 644 for the new file — exactly what umask predicts.

Changing permissions with chmod

```
$ chmod 777 newdir
$ ls -ld newdir
drwxrwxrwx 2 wahid staff 2 2011-01-15 21:09
newdir

$ chmod 755 newdir
$ ls -ld newdir
```

```
drwxr-xr-x 2 wahid staff 2 2011-01-15 21:09
newdir

$ chmod 664 newfile pkg-publisher this
$ ls -l
total 16
drwxr-xr-x 2 wahid staff 2 2011-01-15 21:09
newdir
-rw-rw-r-- 1 wahid staff 0 2011-01-15 21:10
newfile
-rw-rw-r-- 1 wahid staff 11258 2011-01-03
22:09 pkg-publisher
drwxr-xr-x 2 wahid staff 8 2011-01-15 20:02
text
-rw-rw-r-- 1 wahid staff 0 2011-01-15 21:29
this
```

Making a script executable

A common real-world case: a shell script created with echo has no execute bit, so running it directly fails with "Permission denied" until chmod adds the execute bit.

```
$ echo "df -hF zfs" > df-zfs.sh
$ ls -l df-zfs.sh
-rw-r--r-- 1 wahid staff 11 2011-01-17 13:01
df-zfs.sh

$ ./df-zfs.sh
bash: ./df-zfs.sh: Permission denied

$ sh -x df-zfs.sh          # run without
changing permissions, using sh's -x trace
mode
$ df -hF zfs
Filesystem            Size  Used  Avail
Use%  Mounted on
```

```
rpool/ROOT/opensolaris-1   20G  7.2G   12G
38%  /

$ chmod 755 df-zfs.sh
$ ls -l df-zfs.sh
-rwxr-xr-x 1 wahid staff 11 2011-01-17 13:01
df-zfs.sh

$ ./df-zfs.sh          # now runs directly,
since the execute bit is set
Filesystem            Size  Used  Avail
Use%  Mounted on
rpool/ROOT/opensolaris-1   20G  7.2G   12G
38%  /
```

2. chgrp — change group ownership

```
$ whatis chgrp
chgrp (1) - change group ownership
chgrp (1) - change file group ownership
```

chgrp changes which group owns a file or directory. Like chmod and chown, you need sufficient privilege — either you already belong to the target group, or you have root access — to make the change.

Checking current group ownership

```
$ pwd
/export/home/wahid/training/OpenSolaris/pkgdir
$ ls -l
total 13
-rw-r--r-- 1 wahid staff 11258 2011-01-03 22:09 pkg-publisher
drwxr-xr-x 2 wahid staff 8 2011-01-15 17:36 text
```

An unprivileged attempt fails as expected

```
$ id
uid=101(wahid) gid=10(staff)
groups=10(staff)
$ chgrp root *
chgrp: changing group of `pkg-publisher': Not owner
chgrp: changing group of `text': Not owner
```

wahid only belongs to group staff, so attempting to reassign ownership to group root is rejected — you cannot hand a file to a group you don't belong to without elevated privilege.

Switching to root and changing group ownership

```
$ su -
Password:
# cd
/export/home/wahid/training/OpenSolaris/pkgd
ir
# chgrp root *
# ls -l
total 13
-rw-r--r-- 1 wahid root 11258 2011-01-03
22:09 pkg-publisher
drwxr-xr-x 2 wahid root 8 2011-01-15 17:36
text
```

Note that chgrp without -R is not recursive: the text directory itself changed to group root, but the files inside it did not.

Recursive group change with -R

```
# chgrp -R root *
# ls -lR text
text:
-rw-r--r-- 1 wahid root  2856 2011-01-03
22:04 pkg-info.txt
-rw-r--r-- 1 wahid root 475569 2011-01-03
23:38 pkg-install-amp.txt
-rw-r--r-- 1 wahid root  2358 2011-01-03
22:06 pkg-list.txt

# chgrp -R staff *          # revert back to
the original group
# ls -l
-rw-r--r-- 1 wahid staff 11258 2011-01-03
22:09 pkg-publisher
drwxr-xr-x 2 wahid staff 8 2011-01-15 17:36
text
```

3. chown — change file owner and group

```
$ whatis chown
chown (1) - change file owner and group
chown (1) - change file ownership
chown (2) - change owner and group of a file
```

chown changes the owning user of a file or directory, and optionally the group at the same time using the owner:group syntax. Like chgrp, changing ownership normally requires root privilege.

An unprivileged attempt fails

```
$ pwd
/export/home/wahid/training/OpenSolaris/pkgdir
$ id
uid=101(wahid) gid=10(staff)
groups=10(staff)
$ ls -l
-rw-r--r-- 1 wahid staff 11258 2011-01-03
22:09 pkg-publisher
drwxr-xr-x 2 wahid root  8 2011-01-15 20:02
text

$ chown root text
chown: changing ownership of `text': Not
owner
```

As root: changing owner only

```
$ su
Password:
# pwd
/export/home/wahid/training/OpenSolaris/pkgdir
# chown root text
```

```
# ls -l
-rw-r--r-- 1 wahid staff 11258 Jan  3 22:09
pkg-publisher
drwxr-xr-x 2 root  root  8 Jan 15 20:02 text
```

Changing owner and group together

```
# chown wahid:staff text
# ls -l text
-rw-r--r-- 1 wahid staff  2856 Jan  3 22:04
pkg-info.txt
-rw-r--r-- 1 wahid staff 475569 Jan  3 23:38
pkg-install-amp.txt
```

Recursive ownership change with -R

```
# chown -R root:sys text
# ls -ld text
drwxr-xr-x 2 root sys 8 Jan 15 20:02 text
# ls -l text
-rw-r--r-- 1 root sys  2856 Jan  3 22:04
pkg-info.txt
-rw-r--r-- 1 root sys 475569 Jan  3 23:38
pkg-install-amp.txt

# chown -R wahid text          # restore
ownership
# exit
$ id
uid=101(wahid) gid=10(staff)
groups=10(staff)
```

As with chgrp, always double-check with ls -l (or ls -ld for a directory itself, without descending into it) before and after any ownership change — an accidental chown -R on the wrong directory can lock you out of your own files.

4. passwd — change login password and password attributes

```
$ whatis passwd
passwd - change login password and password attributes
```

passwd changes a user's login password. Any user can change their own password by simply running passwd with no arguments; changing another user's password, or root's, requires root privilege.

Changing another user's password as root

```
# id
uid=0(root) gid=0(root)
groups=0(root),1(other),2(bin),3(sys)...

# passwd -r files devuser
New Password:
Re-enter new Password:
passwd: password successfully changed for
devuser
```

The -r files option specifies that the change should be written to the local /etc/shadow file rather than a network directory service — on systems using centralized authentication, -r ldap or -r nis would target LDAP or NIS instead, an important distinction to understand once you reach Chapter 36's coverage of LDAP and directory services.

Preventing password expiration

```
# passwd -x -1 devuser
passwd: password information changed for
devuser
```

The -x -1 option disables the maximum password age, so the account is never forced to change its password automatically — useful for service accounts, though generally discouraged for interactive human users under most security policies.

Verifying the change

```
# grep -i devuser /etc/passwd
devuser:x:101:10:WahidLutfy:/home/devuser:/b
in/bash

# grep -i devuser /etc/shadow
devuser:$6$illustrativeHashOnly$doNotUseInPr
oduction:15061::::::
```

Note: the /etc/shadow line above is shown with an illustrative placeholder hash rather than a real one — never publish an actual password hash, even from a training environment, since offline cracking tools can sometimes recover the original password from it.

5. alias — create a shortcut name for a command

```
$ whatis alias
A shorter name for assigning UNIX commands,
scripts, and other programs.
```

alias lets you define a short, memorable name for a longer command or command pipeline. Its exact syntax depends on which shell you're using: Bourne-family shells (sh, ksh, bash) use name='command', while C-family shells (csh, tcsh) use name command with no equals sign and no quotes required around simple commands.

Bourne/Bash syntax

```
$ ps
PID TTY  TIME CMD
707 pts/3 0:03 bash

$ alias myf='find . -name "*" -print -exec
ls -l {} \;'
$ alias myls='ls -ltr'

$ alias
alias myf='find . -name "*" -print -exec ls
-l {} \;'
alias myls='ls -ltr'
```

Aliases can wrap any command with any options baked in — including an admin-oriented example specific to the system being managed:

```
$ alias myzl='zoneadm list'
$ myzl
global
```

```
$ alias myzl='zoneadm list -p'          #
redefine with an added option
$ myzl
0:global:running:/::native:shared
```

Running the alias myf (equivalent to a recursive ls -lR) confirms the alias behaves exactly like typing out its full definition:

```
$ myf
.
total 1
-rw-r--r-- 1 wahid staff 0 2011-01-09 17:56
newfile
./newfile
-rw-r--r-- 1 wahid staff 0 2011-01-09 17:56
./newfile
```

C shell (csh/tcsh) syntax

Switching shells changes both the prompt (to %) and the alias syntax — no equals sign, and the command is typically left unquoted for simple cases:

```
$ /bin/csh
% ps
PID TTY  TIME CMD
1585 pts/3 0:00 csh

% alias myz zoneadm list -cv
% alias mydf df -hF ufs
% alias
myz (zoneadm list -cv)
mydf (df -hF ufs)
```

Aliases are session-local by default in both shell families — they disappear when the shell exits. To make an alias permanent, add the same alias definition to your shell's startup file: ~/.bashrc for bash, or

~/.cshrc for csh/tcsh, and open a new shell (or source the file) for it to take effect.

Continue the Full Edition

All 43 Chapters — Available Now in eBook, Paperback, and Hardcover

You've just read the Preface and Chapters 1–3 of ***The Direct Path to Linux Ubuntu***. The complete 43-chapter edition continues with:

- Chapter 4 — Advanced UNIX & Linux Commands (user/group management, systemctl, tar/gzip, systemd, dpkg, apt, lsof)
- Chapter 5 — Shell Scripting (Bash & C-shell scripts, conditionals, and the LUBE practice terminal)
- Chapter 6 — UNIX and Linux Web Servers (Apache fundamentals)
- Chapter 7 — The Dynamic Linux Man Page Lookup Tool
- Chapter 8 — The PDF Man Page Library, Browsing by Section
- Chapter 9 — The C and C++ Include Header Files
- Chapter 10 — The Rocky Linux 9.2 Manual Pages
- Chapter 11 — The Red Hat Enterprise Linux 9.2 Manual Pages
- Chapter 12 — Dynamic Generation of Linux Ubuntu Manual Pages (man2pdf)
- Chapter 13 — The Linux Ubuntu 22.04 LTS Manual Pages, Static Library
- Chapter 14 — Linux BASH and Microsoft PowerShell Integration
- Chapter 15 — Learn Linux Ubuntu Online, the LUBE Practice Terminal

- Chapter 16 — The Linux Ubuntu 26.04 Manual Pages in PDF Format
- Chapters 17–20 — Video Companion Resources (Long Lectures & Quick Tips, English and Dari)
- Chapter 21 — Linux Man Pages Online, PDF & HTML by Category
- Chapters 22–27 — C, C++, Go, Java, Ruby, and Rust
- Chapters 28–30 — Perl, R, JavaScript and Node.js
- Chapters 31–33 — HTML5/CSS3, JSON, and XML
- Chapters 34–35 — YAML and SQL Databases
- Chapters 36–37 — LDAP & Directory Services, and Cloud Computing
- Chapters 38–40 — Ansible & OpenShift, AWS & Azure, and Docker
- Chapters 41–43 — Apache & NGINX, Protocols/APIs & Security, and Networking
- Plus: Book-Wide Explanatory Additions and the full Index

Get the Full Edition

eBook

$9.99

Same price on both platforms

Buy on Amazon | Buy on Barnes & Noble

Paperback

$24.99–$34.99

Price varies by platform

Buy on Amazon | Buy on Barnes & Noble

Hardcover

$44.99

Same price on both platforms

Buy on Amazon | Buy on Barnes & Noble

Thank you for reading. If this Free Starter Edition helped you, please subscribe for free video lessons that pair with every chapter: YouTube.com/@MyWebUniversity (English) and YouTube.com/@MyWebUniversity-Dari (Dari).