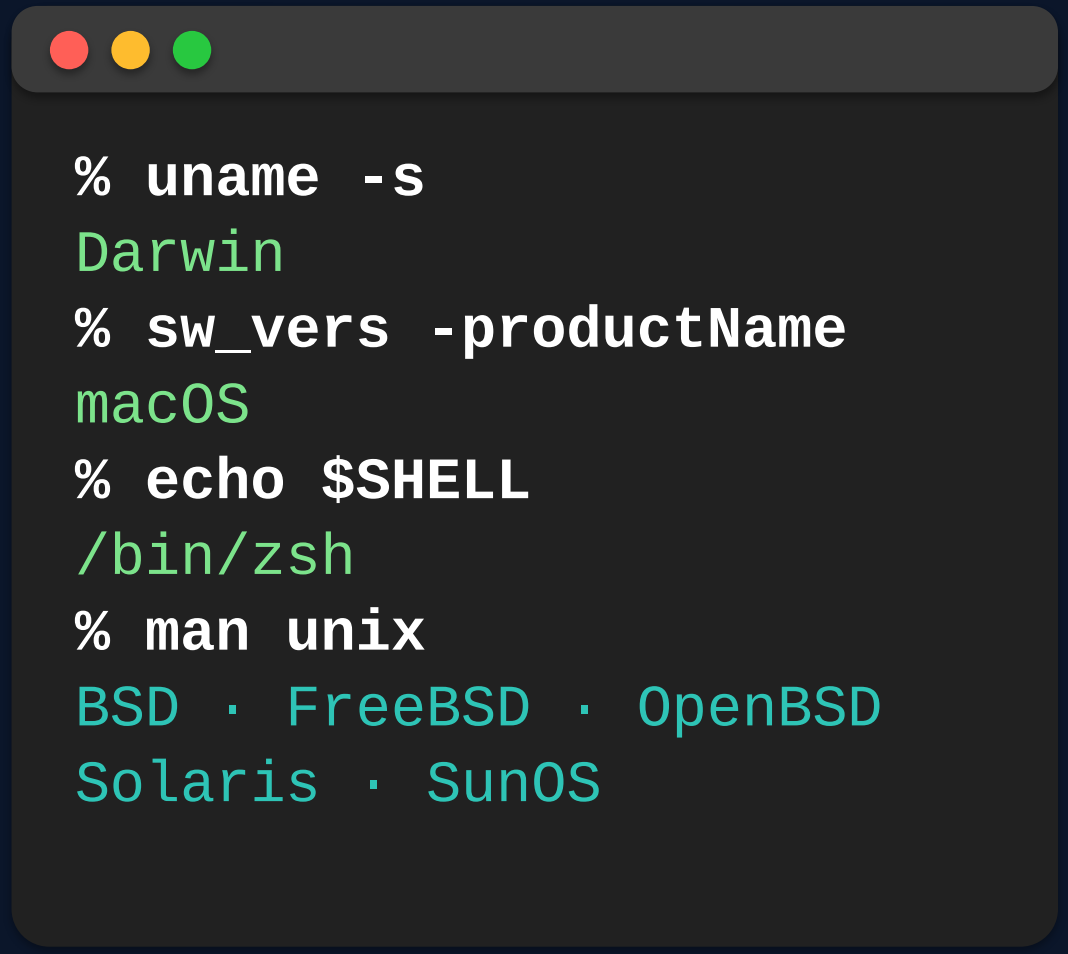


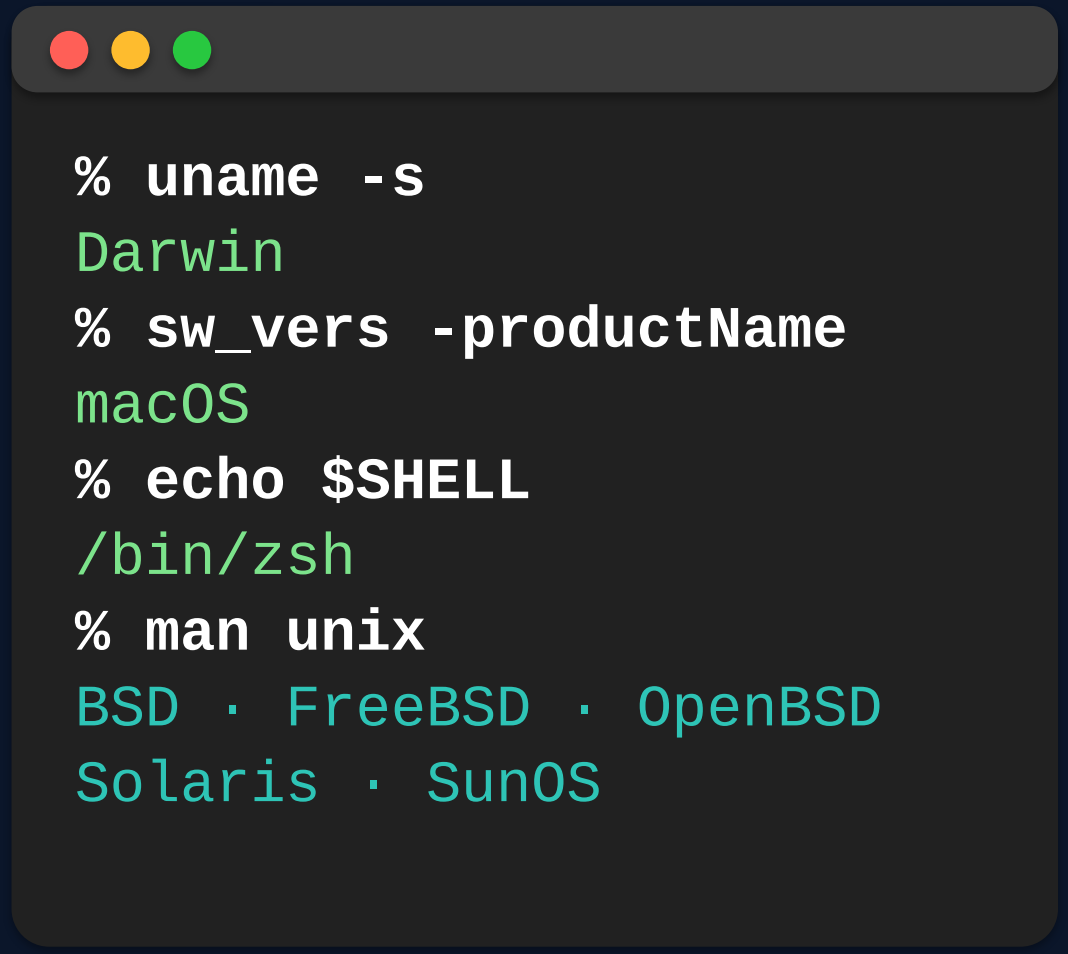
FULL COURSE · 12 LESSONS

macOS UNIX

Darwin & BSD

From UNIX History to Server Administration: GUI & Command Line

Full Course · macOS 27 Golden Gate



LESSON 1 OF 12

macOS UNIX

Darwin & BSD

UNIX Heritage, Darwin Architecture & Your macOS Lab

Full Course · macOS 27 Golden Gate

What You Will Learn in Lesson 1

- Where UNIX came from, and why it still matters
- The two UNIX families: AT&T System V and Berkeley BSD
- FreeBSD, NetBSD, OpenBSD and Sun SunOS / Solaris
- How NeXTSTEP became Mac OS X and then macOS
- Darwin and the XNU kernel: how macOS is built
- Build your lab: a macOS 27 virtual machine in UTM
- Your first UNIX commands on macOS

The 12-Lesson Course Map

#	Lesson	#	Lesson
1	UNIX heritage, Darwin & your lab	7	Databases: SQLite, PostgreSQL, MySQL
2	Install, upgrade & recover macOS	8	Users, groups, sudo, Directory Services
3	GUI, applications & utilities	9	Processes, services & launchd
4	Terminal, zsh & the file system	10	Networking, firewall & SSH
5	Packages: Homebrew, MacPorts, pkg	11	Storage, APFS, security & backup
6	Compilers & interpreters	12	Administering desktop vs server

Where UNIX Began

- 1969: Ken Thompson and Dennis Ritchie start UNIX at AT&T Bell Labs
- Early 1970s: Ritchie creates the C language; UNIX is rewritten in C
- Written in C = portable: UNIX could move to new hardware
- AT&T licensed UNIX source code to universities
- Ideas we still use: small tools, pipes, everything is a file

The UNIX Family Tree

Branch	Origin	Descendants you will meet
System V	AT&T Bell Labs	Sun Solaris (SVR4), IBM AIX, HP-UX
BSD	UC Berkeley (1977+)	FreeBSD, NetBSD, OpenBSD, NeXTSTEP → macOS
UNIX-like	Written from scratch	Linux (with GNU tools), not descended from UNIX code

The BSD Family Today

System	Since	Known for
FreeBSD	1993	Performance, ZFS, jails; servers and network appliances
NetBSD	1993	Portability: runs on a huge range of hardware
OpenBSD	1995	Security first; home of OpenSSH and the pf firewall
macOS (Darwin)	2001	BSD userland on the XNU kernel; desktop UNIX

Sun Microsystems: SunOS and Solaris

- SunOS 1.x–4.x (1982+): BSD-based UNIX on Sun workstations
- Solaris 2 (1992): rebuilt on AT&T System V Release 4 (SunOS 5.x)
- Gave the industry ZFS, DTrace, Zones and SMF
- DTrace also ships in macOS and FreeBSD
- Oracle acquired Sun in 2010; Oracle Solaris 11.4 continues today

From NeXTSTEP to macOS

1985	Steve Jobs founds NeXT
1989	NeXTSTEP: Mach kernel + BSD UNIX + object-oriented GUI
1997	Apple acquires NeXT; NeXTSTEP becomes the base of the next Mac OS
2000	Darwin, the open source core, is released
2001	Mac OS X 10.0 ships
2007	Mac OS X 10.5 Leopard: certified UNIX 03
2012 / 2016	Renamed OS X, then macOS
2019	zsh becomes the default shell (Catalina)
2020	Apple silicon Macs (Big Sur)
2025 / 2026	Tahoe 26: last for Intel · Golden Gate 27

How macOS Is Built: The Layers

Layer	What lives there
Applications	Finder, Safari, Terminal, your apps
Frameworks	SwiftUI, AppKit (Cocoa), Core Data, Metal, Core Audio
Darwin: userland	libSystem, launchd, zsh/bash, BSD commands (ls, ps, grep)
Darwin: XNU kernel	Mach + BSD layer + I/O Kit / DriverKit
Hardware	Apple silicon (M-series) or Intel (older Macs)

Inside XNU: "X is Not Unix"

- Mach: tasks, threads, virtual memory, message passing (IPC)
- BSD layer: processes, users, permissions, POSIX APIs, sockets, file systems
- I/O Kit and DriverKit: device drivers
- Kernel extensions (kexts) are being replaced by system extensions
- Darwin source code is published at github.com/apple-oss-distributions

macOS vs FreeBSD vs OpenBSD vs Solaris

	macOS 27	FreeBSD	OpenBSD	Solaris 11.4
Kernel	XNU	FreeBSD	OpenBSD	SunOS 5.11
Default shell	zsh	sh	ksh	bash
Packages	Homebrew, MacPorts*	pkg, ports	pkg_add, ports	pkg (IPS)
Services	launchd / launchctl	rc.d / service	rc.d / rcctl	SMF / svcadm
Firewall	pf + App Firewall	pf, ipfw	pf	PF, IP Filter
File system	APFS	ZFS, UFS	FFS	ZFS

* third-party package managers; Apple also provides the App Store and .pkg installers

macOS Is Certified UNIX — With BSD Flavor

- Apple has registered macOS with The Open Group's UNIX 03 standard since 10.5
- Means: POSIX APIs, a standard shell and standard utilities
- macOS tools are BSD versions, not GNU (Linux) versions

```
Linux: sed -i 's/a/b/' file      macOS: sed -i '' 's/a/b/' file
Linux: ls --color      macOS: ls -G
```

- Want GNU tools too? Homebrew coreutils (Lesson 5)

Lab Setup: Why a macOS Virtual Machine?

- Practice admin commands safely: break it, restore it
- Snapshots before risky changes
- Needs an Apple silicon Mac (M1 or newer)
- Apple's license allows up to 2 extra macOS VMs on Apple hardware
- Guest macOS should not be newer than the host macOS
- Suggested VM: 4 CPU cores · 8 GB RAM · 80 GB disk

Lab: Create the macOS 27 VM in UTM

- 1. Install UTM (free) from mac.getutm.app or the Mac App Store
- 2. Create a New Virtual Machine → Virtualize → macOS 12+
- 3. Choose a restore image (.ipsw) or let UTM download the latest
- 4. Set CPU cores, memory and disk size → Save
- 5. Start the VM: macOS installs, then Setup Assistant opens
- 6. Create your admin account; take a clean snapshot / copy
- Parallels Desktop users: File → New → install macOS from the restore image

Your First UNIX Commands on macOS

```
% sw_vers                          # macOS version and build
ProductName:      macOS
ProductVersion:   27.2.1
BuildVersion:     26B131

% uname -srm                       # kernel name, release, hardware
Darwin 26.x.x arm64

% echo $SHELL                      # your login shell
/bin/zsh

% csrutil status                   # System Integrity Protection
System Integrity Protection status: enabled.

% system_profiler SPHardwareDataType   # hardware summary
```

Example output — your version numbers will differ.

Summary & Homework

- UNIX: Bell Labs 1969 → System V and BSD families
- macOS = Darwin (XNU kernel + BSD userland) + Apple frameworks
- FreeBSD, OpenBSD and Solaris share the same core ideas
- Homework: build the UTM VM and run the 5 commands
- Homework: write down your sw_vers and uname output
- Next: Lesson 2 — Installing, Upgrading & Recovering macOS

Thank You! 🙏

Free videos & PDFs: www.MyWebUniversity.com

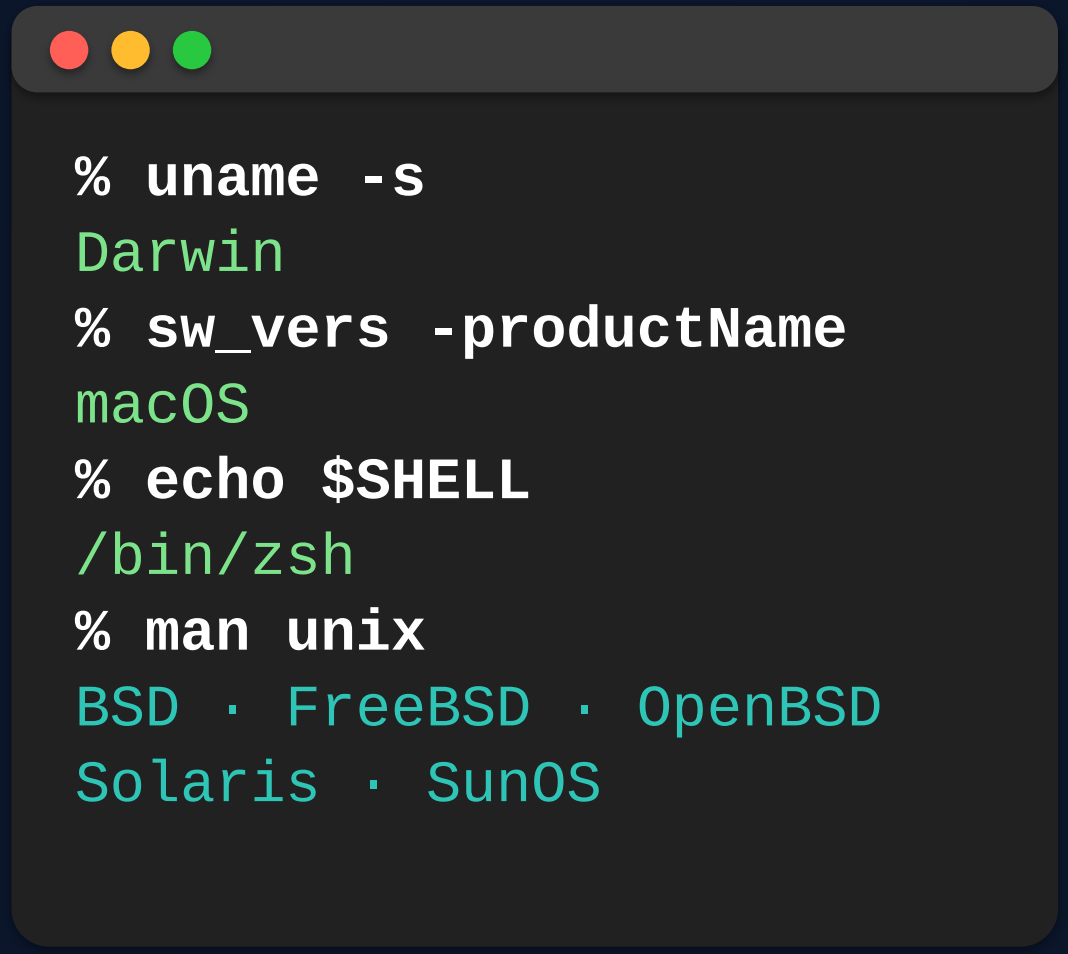
Slides for this lesson: MyWebUniversity.com/presentations

YouTube: @MyWebUniversity · @MyWebUniversity-Dari

Wahidullah Lutfy — Founder and Director of IT of MyWebUniversity, and CS teacher

Book: The Direct Path to Linux Ubuntu (Amazon · Barnes & Noble)

Learn it, then pass it on.



LESSON 2 OF 12

macOS UNIX

Darwin & BSD

Installing, Upgrading & Recovering macOS

Full Course · macOS 27 Golden Gate

What You Will Learn in Lesson 2

- Prepare: compatibility, disk space and a backup first
- Start an Apple silicon Mac in macOS Recovery
- Recovery tools: reinstall, restore, Disk Utility, Terminal
- Clean install, Erase All Content and Settings, bootable USB installer
- Setup Assistant: the first account and security choices
- Updates vs upgrades in the GUI and with softwareupdate
- Rollback plans, and how FreeBSD, OpenBSD and Solaris do it

Before You Install or Upgrade

- macOS 27 Golden Gate runs on Apple silicon Macs only
- Check your model in Apple menu → About This Mac
- Free disk space: plan for 30–40 GB for a major upgrade
- Back up first: Time Machine, or clone/snapshot your VM
- Know your passwords: Apple Account, admin, FileVault recovery key
- Check important apps support the new version

Starting macOS Recovery

Mac	How to start Recovery
Apple silicon	Shut down → press and hold the power button until "Loading startup options" → Options → Continue
Apple silicon (fallback)	Shut down, then press-release-press-and-hold the power button
Intel (older Macs)	Hold Command (⌘) + R at startup; Option + ⌘ + R = Internet Recovery
UTM virtual machine	Right-click the VM → "Boot into Recovery Mode" (or hold power in the VM)

The Tools Inside macOS Recovery

Tool	Use it to
Restore from Time Machine	Bring back the whole system from a backup
Reinstall macOS	Install a fresh copy of macOS (keeps your data if you don't erase)
Safari	Read Apple support articles while the Mac is broken
Disk Utility	First Aid, erase and format disks (APFS)
Utilities → Terminal	Run commands such as csrutil, diskutil, resetpassword
Utilities → Startup Security Utility	Set Full or Reduced Security for the startup disk

Three Ways to Do a Clean Install

- 1. Recovery: Disk Utility → erase the internal disk (APFS) → Reinstall macOS
- 2. System Settings → General → Transfer or Reset → Erase All Content and Settings
- 3. A bootable USB installer (next slide): best for many Macs or slow internet
- Apple silicon also supports a full restore from an IPSW file with Apple Configurator
- In the lab: UTM installs from an IPSW automatically

Create a Bootable USB Installer

```
% softwareupdate --list-full-installers        # which full installers exist
% softwareupdate --fetch-full-installer \
      --full-installer-version 27.2.1           # downloads to /Applications

# Erase a 32 GB+ USB drive as Mac OS Extended, named MyUSB, then:
% sudo /Applications/Install\ macOS*.app/Contents/Resources/createinstallmedia \
      --volume /Volumes/MyUSB

# Boot from it: hold the power button → choose the installer
```

Replace 27.2.1 with a version shown by --list-full-installers.

Setup Assistant: First Boot Choices

- Country, language, accessibility and network
- Migration Assistant: move data from another Mac or a backup
- Apple Account: optional; you can skip it in a lab
- Create the computer account: the FIRST account is an administrator
- FileVault disk encryption: keep the recovery key safe
- Location, analytics, Screen Time, Siri, Touch ID, appearance

Updates vs Upgrades

	Update	Upgrade
Example	27.2 → 27.2.1	26 Tahoe → 27 Golden Gate
Contains	Security and bug fixes	New features, new APIs, changed tools
Risk	Low	Medium: test apps and scripts first
Where (GUI)	System Settings → General → Software Update	Same place: "Upgrade Now"
Tip	Install promptly	Wait for x.1, back up, test in a VM

softwareupdate: Updates From the Command Line

```
% softwareupdate --list                          # what is available
% sudo softwareupdate --install --all            # install everything
% sudo softwareupdate -i -a --restart            # install and restart if needed
% softwareupdate --history                       # what was installed, and when
% sudo softwareupdate --install-rosetta --agree-to-license

% sw_vers                                        # confirm the new version
% tail -f /var/log/install.log                   # watch the installer log
```

App Compatibility and Rosetta 2

- Native apps are built for Apple silicon (arm64)
- Rosetta 2 translates Intel (x86_64) apps
- Apple has said Rosetta 2 stays broadly available through macOS 27
- Check an app: Get Info in Finder, or the Kind column in Activity Monitor
- Command line: file /Applications/App.app/Contents/MacOS/App
- Plan now to replace Intel-only apps

Your Rollback Plan

- macOS cannot simply 'uninstall' an upgrade
- Going back = erase + install the older version + restore your data
- Real Mac: Time Machine backup made BEFORE the upgrade
- Lab: duplicate the VM in UTM before every risky change
- Keep the older full installer or IPSW you may need
- Test the plan once, before you need it

Updating Other UNIX Systems

System	Security fixes	Major upgrade	Packages
macOS	softwareupdate -i -a	softwareupdate / installer app	brew upgrade (Lesson 5)
FreeBSD	freebsd-update fetch install	freebsd-update upgrade -r <ver>	pkg upgrade
OpenBSD	syspatch	sysupgrade	pkg_add -u
Solaris 11.4	pkg update	pkg update (new boot environment)	pkg update

Lab: Practice Updates in Your VM

```
# 1. In UTM: right-click the VM → Clone (your rollback copy)
# 2. In the VM, open Terminal:
% sw_vers
% softwareupdate --list
% softwareupdate --history
% softwareupdate --list-full-installers | head
# 3. Start the VM in Recovery Mode and open Utilities → Terminal:
-bash-3.2# csrutil status
-bash-3.2# reboot
```

Keep the clone until you finish Lesson 3.

Summary & Homework

- Prepare: compatibility, space, backup, passwords
- Recovery: hold the power button → Options
- Clean install: Recovery, Erase All Content and Settings, USB, IPSW
- softwareupdate --list / --install --all / --history
- Homework: clone the VM, run the lab, boot into Recovery once
- Next: Lesson 3 — The macOS GUI, Applications & Utilities

Thank You! 🙏

Free videos & PDFs: www.MyWebUniversity.com

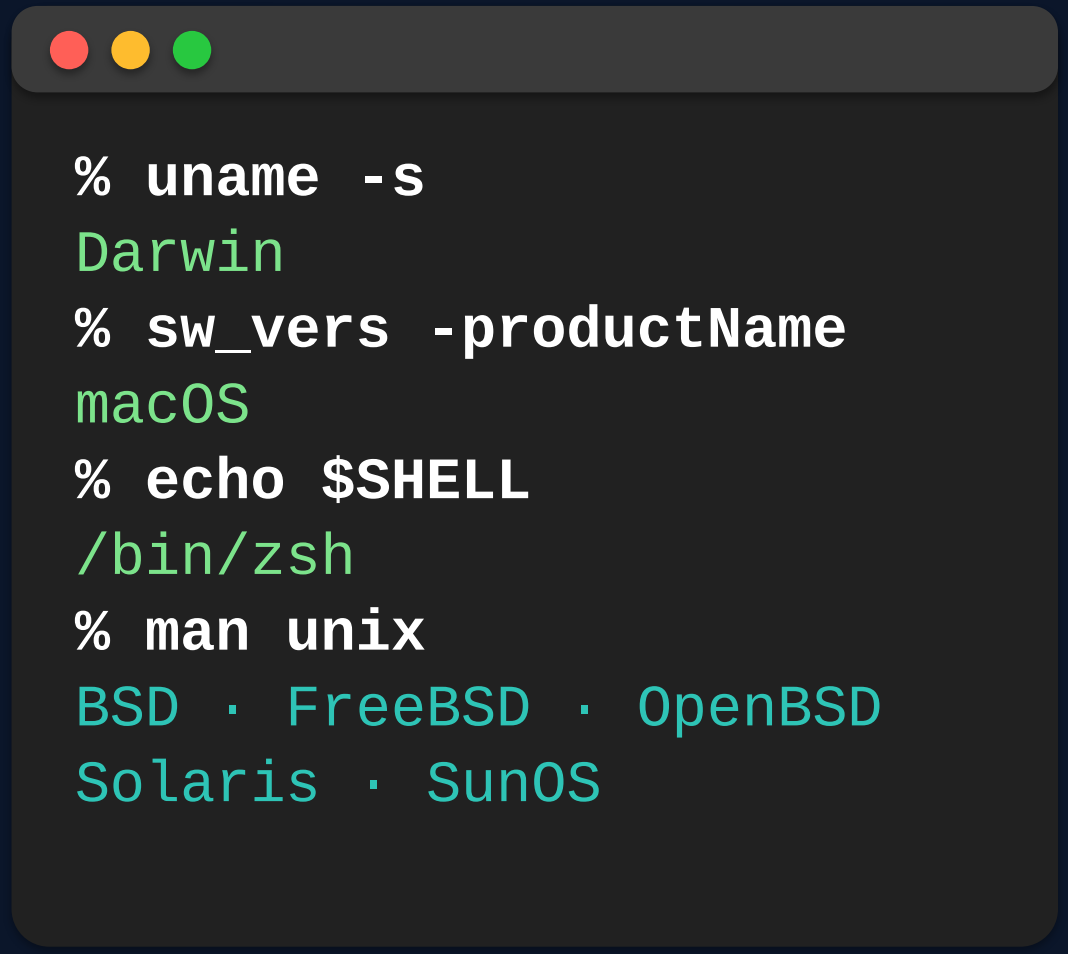
Slides for this lesson: MyWebUniversity.com/presentations

YouTube: @MyWebUniversity · @MyWebUniversity-Dari

Wahidullah Lutfy — Founder and Director of IT of MyWebUniversity, and CS teacher

Book: The Direct Path to Linux Ubuntu (Amazon · Barnes & Noble)

Learn it, then pass it on.



LESSON 3 OF 12

macOS UNIX

Darwin & BSD

The macOS GUI, Running Applications & Built-in Utilities

Full Course · macOS 27 Golden Gate

What You Will Learn in Lesson 3

- The desktop: menu bar, Dock, Finder, Spotlight, Control Center
- Finder power skills and hidden files
- System Settings: where the admin panels live
- Where apps live, and what an .app bundle really is
- Installing apps safely: App Store, .dmg, .pkg and Gatekeeper
- Opening, quitting and force-quitting apps from Terminal
- Utilities: Activity Monitor, Console, Disk Utility and automation

A Tour of the Desktop

Part	What it does	Shortcut
Apple menu	About This Mac, System Settings, Restart, Shut Down, Lock Screen	—
Menu bar	The active app's menus; status icons and Control Center on the right	—
Dock	Launch and switch apps; minimized windows; Trash	⌥⌘D hide/show
Finder	The file manager; always running	⌘N new window
Spotlight	Search files, apps, settings; launch anything	⌘Space
Mission Control	See all windows and desktops (Spaces)	^↑

Finder Power Skills

- Go to any folder: Go → Go to Folder (⇧⌘G), type /etc or ~/Library
- Show hidden files: ⇧⌘. (Shift-Command-Period)
- Show the path bar: View → Show Path Bar (⌥⌘P)
- Quick Look: select a file and press Space
- Get Info (⌘I): size, permissions, 'Open with', kind (Apple silicon / Intel)
- Drag a folder from Finder into Terminal to paste its full path

System Settings for Administrators

Panel	Why an admin goes there
General → About / Software Update / Storage	Version, updates, disk usage
General → Login Items & Extensions	What starts at login; system extensions
General → Sharing	Computer name, File, Screen and Remote Login (SSH) sharing
Users & Groups	Create users, admin rights, login options (Lesson 8)
Network	Wi-Fi, Ethernet, DNS, firewall (Lesson 10)
Privacy & Security	App permissions, Gatekeeper 'Open Anyway', FileVault (Lesson 11)

Where Apps Live: The .app Bundle

```
/Applications            # apps for all users
/System/Applications     # Apple's built-in apps (read-only system volume)
~/Applications           # apps for one user only

% ls /Applications/Safari.app/Contents
Info.plist  MacOS  Resources  _CodeSignature  ...

% ls /System/Applications/Calculator.app/Contents/MacOS
Calculator                    # the real executable

% plutil -p /System/Applications/Calculator.app/Contents/Info.plist | head
```

Installing Apps Safely

- App Store: reviewed, sandboxed, updates automatically
- .dmg disk image: open it, drag the app to Applications, eject
- .pkg installer: runs scripts and can install anywhere (needs admin)
- Homebrew casks: brew install --cask <app> (Lesson 5)
- Gatekeeper checks the developer signature and Apple notarization
- Blocked app? System Settings → Privacy & Security → Open Anyway

Control Apps From Terminal

```
% open -a Safari                          # launch an app by name
% open -a TextEdit notes.txt              # open a file with a chosen app
% open .                                  # open this folder in Finder
% open https://www.MyWebUniversity.com # default browser
% open -R /etc/hosts                      # reveal a file in Finder

% osascript -e 'quit app "Safari"'      # ask an app to quit nicely
% killall Safari                         # force it to quit
% mdfind -name .app | head               # find apps with Spotlight
```

When an App Misbehaves

- Force Quit window: ⌥⌘Esc
- Activity Monitor: select the process → Stop (✕) → Quit or Force Quit
- Terminal: pkill -x Safari or kill <PID> (Lesson 9)
- Remove login items you don't need: General → Login Items
- Safe Mode (Apple silicon): hold power → select disk → hold Shift → Continue in Safe Mode
- Read the crash report in Console → Crash Reports

The Built-in Utilities (/Applications/Utilities)

Utility	What it is for
Activity Monitor	CPU, memory, energy, disk and network per process
Console	System logs, crash reports, live log streaming
Disk Utility	Volumes, First Aid, erase, disk images (Lesson 11)
Terminal	The UNIX shell (Lesson 4)
System Information	Full hardware and software report (system_profiler)
Script Editor	AppleScript and JavaScript for Automation
Screenshot · Migration Assistant · Audio MIDI Setup	Capture screen · move users and data · audio devices

Reading Activity Monitor Like an Admin

- CPU tab: % CPU per process; 'Kind' shows Apple silicon or Intel
- Memory tab: watch Memory Pressure (green, yellow, red), not 'free' memory
- Energy tab: apps that drain the battery or prevent sleep
- Disk and Network tabs: who reads, writes and sends the most
- Double-click a process: open files, ports, parent process
- Same data in Terminal: top, ps, lsof, vm_stat (Lesson 9)

Automation: Shortcuts, AppleScript and say

```
% shortcuts list                            # Shortcuts you have created
% shortcuts run "Resize Image"              # run one from Terminal

% osascript -e 'display notification "Backup done" with title "MWU"'
% osascript -e 'tell app "Finder" to get name of every disk'

% say "Your build has finished"
% defaults read com.apple.dock orientation   # read an app's setting
```

The Desktop on Other UNIX Systems

System	Default desktop	How you get a GUI
macOS	Aqua (built in)	Always there; WindowServer draws it
FreeBSD	None (console)	pkg install xorg + KDE Plasma, GNOME or Xfce
OpenBSD	X11 included (Xenocara)	Install the X sets; cwm or fvwm, or pkg_add a desktop
Solaris 11.4	GNOME (optional)	pkg install solaris-desktop

Lab: GUI Meets UNIX

```
% open -a "Activity Monitor"
% open /System/Applications/Utilities
% ls /System/Applications/Calculator.app/Contents
% plutil -p /System/Applications/Calculator.app/Contents/Info.plist | grep -i version

# Show hidden files in every Finder window, then undo it:
% defaults write com.apple.finder AppleShowAllFiles -bool true;  killall Finder
% defaults delete com.apple.finder AppleShowAllFiles;            killall Finder
```

Summary & Homework

- Menu bar, Dock, Finder, Spotlight (⌘Space), Mission Control
- Finder: ⇧⌘G, ⇧⌘., path bar, Quick Look, drag to Terminal
- An .app is a bundle (a folder); Gatekeeper checks signatures
- open, osascript, killall, mdfind, shortcuts, defaults
- Homework: open every utility once; do the lab; learn 5 shortcuts
- Next: Lesson 4 — Terminal, zsh & the macOS File System

Thank You! 🙏

Free videos & PDFs: www.MyWebUniversity.com

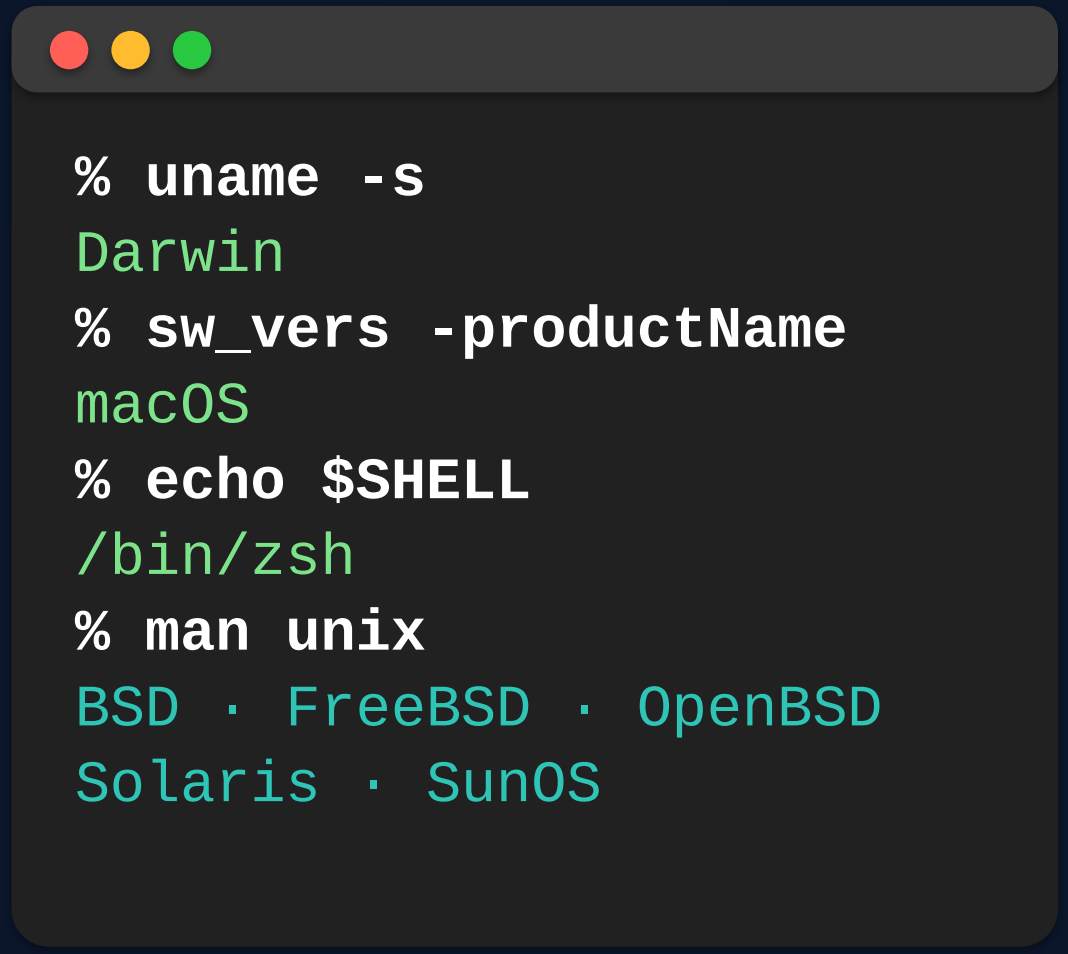
Slides for this lesson: MyWebUniversity.com/presentations

YouTube: @MyWebUniversity · @MyWebUniversity-Dari

Wahidullah Lutfy — Founder and Director of IT of MyWebUniversity, and CS teacher

Book: The Direct Path to Linux Ubuntu (Amazon · Barnes & Noble)

Learn it, then pass it on.



LESSON 4 OF 12

macOS UNIX

Darwin & BSD

Terminal, zsh & the macOS File System

Full Course · macOS 27 Golden Gate

What You Will Learn in Lesson 4

- Terminal, the shells macOS ships, and how to switch
- zsh startup files and the PATH
- Getting help: man, apropos, which, type
- The file system layout and the sealed system volume
- Files, text, permissions, pipes and redirection
- BSD vs GNU differences and macOS-only commands
- zsh power features, and the shells of BSD and Solaris

Terminal and Your Shells

```
% echo $SHELL              # your login shell
/bin/zsh
% cat /etc/shells          # shells you are allowed to use
/bin/bash  /bin/csh  /bin/dash  /bin/ksh  /bin/sh  /bin/tcsh  /bin/zsh
% bash --version | head -1
GNU bash, version 3.2.57 ...    # old: newer bash uses the GPLv3 license
% chsh -s /bin/bash        # change your login shell (asks for password)
% chsh -s /bin/zsh         # change it back
```

Example output. Newer bash: brew install bash (Lesson 5).

zsh Startup Files and the PATH

File	Read when	Put here
~/.zshenv	Every zsh, even scripts	Rarely needed; keep it tiny
~/.zprofile	Login shells (each new Terminal window)	PATH changes, e.g. Homebrew
~/.zshrc	Interactive shells	Aliases, functions, prompt, options
/etc/paths, /etc/paths.d/	By path_helper at login	System-wide PATH entries
~/.bash_profile	bash login shells	If you switch to bash

Getting Help Without Leaving Terminal

```
% man ls                    # the manual page (q to quit, /word to search)
% man -k password           # search all manual pages (same as apropos)
% man 5 crontab             # section 5 = file formats
% which -a python3          # every python3 on your PATH
% type ls                   # alias, builtin, function or file?
% whatis grep               # one-line description
% help                      # zsh has run-help for builtins:  run-help cd
```

The macOS File System Layout

Path	What lives there
/System	macOS itself, on the sealed, read-only system volume
/Library · ~/Library	System-wide and per-user settings, caches, fonts, LaunchDaemons
/Applications · /Users	Apps for all users · home folders (/Users/<name>)
/etc → /private/etc	UNIX configuration files (hosts, shells, paths, sudoers)
/var → /private/var · /tmp	Logs, databases, spools · temporary files
/usr/bin · /bin · /sbin	Apple's UNIX commands (read-only)
/usr/local · /opt/homebrew	Your own software; Homebrew on Apple silicon
/Volumes	Mounted disks, USB drives and disk images

The Sealed System Volume and Firmlinks

```
% ls -l / | grep -E 'etc|var|tmp'
lrwxr-xr-x  ... etc -> private/etc
lrwxr-xr-x  ... tmp -> private/tmp
lrwxr-xr-x  ... var -> private/var

% mount | head -2
/dev/disk3s1s1 on / (apfs, sealed, local, read-only, journaled)
/dev/disk3s5 on /System/Volumes/Data (apfs, local, journaled, nobrowse)

% sudo touch /usr/bin/test     # fails: Read-only file system
```

Example output; disk numbers vary.

Files and Folders: The Daily Commands

```
% pwd;  cd ~;  cd -;  cd ..          # where am I, home, back, up
% ls -laG                           # long list, hidden files, color
% mkdir -p ~/lab/a/b                # make nested folders
% cp -R ~/lab ~/lab-copy            # copy a folder
% mv old.txt new.txt;  rm -i file   # rename; delete with a prompt
% ln -s /etc/hosts ~/lab/hosts      # symbolic link
% file /bin/ls;  stat -x /etc/hosts # what is it; details
% find ~ -name '*.log' -mtime -7    # logs changed in 7 days
% du -sh ~/Library;  df -h /        # folder size; free space
```

Reading and Editing Text

```
% cat /etc/shells                # print a short file
% less /var/log/system.log       # page through (space, b, /search, q)
% head -20 file;  tail -f file   # first lines; follow a growing log
% vim ~/.zshrc                   # vi/vim: i insert, Esc, :wq save+quit
% nano ~/.zshrc                  # simple editor (^O save, ^X exit)
% open -e ~/.zshrc               # edit in TextEdit
% echo 'alias ll="ls -laG"' >> ~/.zshrc;  source ~/.zshrc
```

Permissions in One Slide

```
% ls -l ~/lab/script.sh
-rwxr-xr--@ 1 wahid  staff  120 Nov 17 10:00 script.sh
# type | owner rwx | group r-x | others r-- | @ = extended attributes

% chmod 755 script.sh       # rwx r-x r-x   (programs, folders)
% chmod 644 notes.txt       # rw- r-- r--   (normal files)
% chmod u+x script.sh       # add execute for the owner
% sudo chown root:wheel file
% ls -le@ file              # also show ACLs (+) and xattrs (@)
```

Pipes, Redirection and Text Tools

```
% ls /usr/bin > list.txt          # write output to a file
% date >> list.txt                # append
% ls /nothere 2> errors.txt       # errors only
% ls /usr/bin | wc -l             # count commands
% ps aux | sort -nrk 3 | head -5  # top 5 by CPU
% grep -i error /var/log/install.log | tail
% cut -d: -f1 /etc/passwd | sort | uniq | head
% awk '{print $1}' list.txt | sort | uniq -c | sort -nr | head
```

BSD (macOS) vs GNU (Linux) Tools

Task	macOS / BSD	Linux / GNU
Edit file in place	sed -i '' 's/a/b/' f	sed -i 's/a/b/' f
Yesterday's date	date -v-1d	date -d yesterday
File size with stat	stat -f %z f	stat -c %s f
Colored ls	ls -G	ls --color
Checksum	shasum -a 256 f · md5 f	sha256sum f · md5sum f
Extended regex in find/sed	find -E . ; sed -E	find -regextype … ; sed -E

Commands Only macOS Has

Command	What it does
open · pbcopy · pbpaste	Open files/apps · copy to and paste from the clipboard
mdfind · mdls	Spotlight search · show a file's metadata
defaults · plutil	Read/write app settings · check and convert .plist files
diskutil · hdiutil	Disks and APFS · disk images (Lesson 11)
networksetup · scutil	Network settings · DNS, hostname (Lesson 10)
softwareupdate · sw_vers · system_profiler	Updates · version · hardware report
caffeinate · pmset · screencapture · say	Stay awake · power settings · screenshot · speech

zsh Power Features (~/.zshrc)

```
# ~/.zshrc  — MyWebUniversity lab settings
PROMPT='%n@%m %1~ %# '          # user@host folder %
alias ll='ls -laG'
alias ..='cd ..'
setopt AUTO_CD HIST_IGNORE_DUPS SHARE_HISTORY
autoload -Uz compinit && compinit  # smart Tab completion
mkcd() { mkdir -p "$1" && cd "$1"; }

% ls **/*.log      # recursive glob
% ^R               # Ctrl-R: search your history
```

Shells and Layout on Other UNIX Systems

	macOS	FreeBSD	OpenBSD	Solaris 11.4
Default user shell	zsh	sh	ksh	bash
Root shell	zsh (root is disabled)	sh	ksh	bash (root is a role)
Add-on software	/opt/homebrew	/usr/local	/usr/local	/usr, /opt (IPS)
System config	/etc + plists	/etc/rc.conf	/etc/*.conf, rcctl	SMF properties
Tools flavor	BSD	BSD	BSD	System V + GNU

Lab: Live in Terminal

```
% echo $SHELL; cat /etc/shells; bash --version | head -1
% ls -l / | grep private;  mount | head -2
% mkdir -p ~/lab && cd ~/lab
% ls /usr/bin > cmds.txt;  wc -l cmds.txt
% grep -c '^z' cmds.txt              # commands starting with z
% date -v-1d;  stat -f '%z bytes' cmds.txt
% echo 'alias ll="ls -laG"' >> ~/.zshrc && source ~/.zshrc && ll
% man -k disk | head
```

Summary & Homework

- zsh is the default; /etc/shells lists the others; chsh switches
- .zprofile for PATH, .zshrc for aliases and prompt
- Sealed read-only system volume + Data volume; /etc → /private/etc
- Pipes + grep, sort, uniq, cut, awk answer most questions
- Homework: finish the lab; build your own .zshrc with 3 aliases
- Next: Lesson 5 — Package Management: Homebrew, MacPorts, pkg

Thank You! 🙏

Free videos & PDFs: www.MyWebUniversity.com

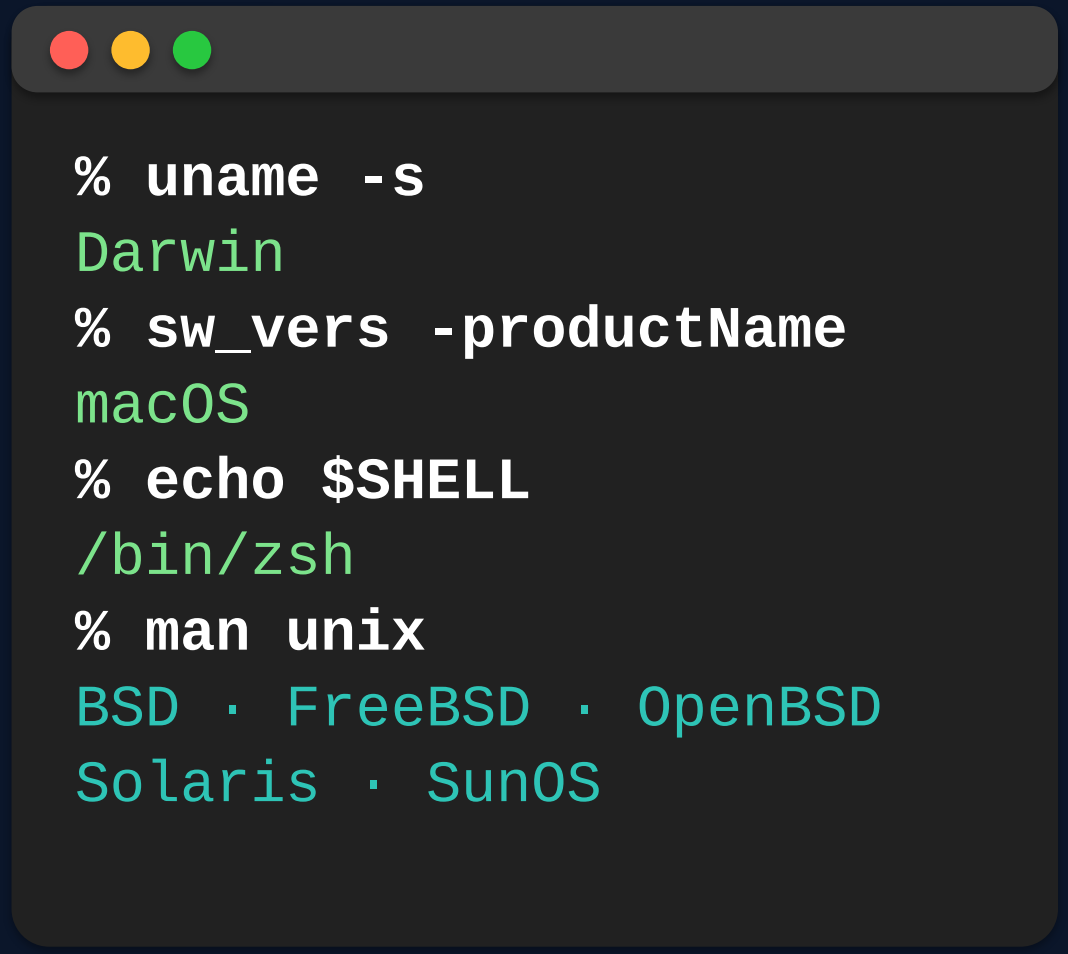
Slides for this lesson: MyWebUniversity.com/presentations

YouTube: @MyWebUniversity · @MyWebUniversity-Dari

Wahidullah Lutfy — Founder and Director of IT of MyWebUniversity, and CS teacher

Book: The Direct Path to Linux Ubuntu (Amazon · Barnes & Noble)

Learn it, then pass it on.



LESSON 5 OF 12

macOS UNIX

Darwin & BSD

Package Management: Homebrew, MacPorts, pkg & App Store

Full Course · macOS 27 Golden Gate

What You Will Learn in Lesson 5

- The ways software arrives on a Mac
- Install Homebrew and learn its vocabulary
- Install, upgrade, clean up, and run services with brew
- Brewfiles: rebuild a whole machine with one command
- GNU tools side by side with BSD tools
- MacPorts, .pkg installers with pkgutil, and the App Store CLI
- pkg on FreeBSD, pkg_add on OpenBSD, and IPS on Solaris

How Software Arrives on a Mac

Method	Where it installs	Best for
App Store	/Applications	Sandboxed desktop apps; automatic updates
.dmg (drag and drop)	/Applications	Desktop apps from a developer's website
.pkg installer	Anywhere (needs admin)	Drivers, system tools, enterprise software
Homebrew formula / cask	/opt/homebrew · /Applications	UNIX tools and developer apps
MacPorts	/opt/local	UNIX tools, built in a BSD ports style
Source (./configure && make)	/usr/local	Learning, or software with no package

Install Homebrew

```
# 1. Run the official installer (from brew.sh):
% /bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
#    It installs the Xcode Command Line Tools if they are missing.

# 2. Add brew to your PATH (Apple silicon):
% echo 'eval "$(/opt/homebrew/bin/brew shellenv)"' >> ~/.zprofile
% eval "$(/opt/homebrew/bin/brew shellenv)"

% brew --version;  brew doctor     # "Your system is ready to brew."
```

Always copy the install command from brew.sh. A .pkg installer is also offered there.

Homebrew Vocabulary

Word	Meaning	Example
Formula	A command-line package, built from a recipe	brew install wget
Cask	A desktop app (.app) or binary installer	brew install --cask firefox
Bottle	A pre-built binary of a formula (fast install)	Downloaded automatically
Tap	An extra repository of formulas and casks	brew tap <user>/<repo>
Cellar	Where installed versions live	/opt/homebrew/Cellar
Keg	One installed version inside the Cellar	/opt/homebrew/Cellar/wget/1.x

brew: The Daily Commands

```
% brew search nginx            # find packages
% brew info wget               # version, dependencies, caveats
% brew install wget tree htop  # install formulas
% brew install --cask visual-studio-code
% brew list;  brew leaves      # everything · only what YOU asked for
% brew outdated                # what has a newer version
% brew upgrade                 # upgrade everything (or: brew upgrade wget)
% brew uninstall htop;  brew autoremove   # remove + unused dependencies
% brew cleanup                 # delete old versions and downloads
```

Services and Brewfiles

```
% brew install postgresql@17
% brew services start postgresql@17   # starts now AND at login (launchd)
% brew services list                  # name, status, user, plist file
% brew services stop postgresql@17

% brew bundle dump --file=~/Brewfile  # save everything installed
% cat ~/Brewfile
brew "wget"
cask "visual-studio-code"
% brew bundle install --file=~/Brewfile   # rebuild on a new Mac
```

PostgreSQL version is an example: use brew search postgresql for current ones.

GNU Tools Side by Side

```
% brew install coreutils gnu-sed grep findutils gawk bash

% gls --color=auto                # GNU ls   (BSD ls stays as ls)
% gsed -i 's/old/new/' file.txt   # GNU sed  (no '' needed)
% gdate -d yesterday              # GNU date
% ggrep -P '\d+' file.txt         # GNU grep with Perl regex
% $(brew --prefix)/bin/bash --version   # modern bash 5.x
```

MacPorts: The BSD Ports Style

```
# Install the MacPorts .pkg for your macOS version from macports.org
% sudo port selfupdate                  # update the ports tree
% port search nmap
% sudo port install nmap
% port installed
% sudo port upgrade outdated
% sudo port uninstall nmap
# Everything lives under /opt/local (add /opt/local/bin to PATH)
```

.pkg Installers: installer and pkgutil

```
% pkgutil --check-signature MyTool.pkg     # who signed it?
% pkgutil --expand-full MyTool.pkg /tmp/x  # look inside before installing
% sudo installer -pkg MyTool.pkg -target / # install from Terminal

% pkgutil --pkgs | grep -v com.apple       # installed (non-Apple) packages
% pkgutil --pkg-info com.example.mytool
% pkgutil --files com.example.mytool       # every file it installed
% sudo pkgutil --forget com.example.mytool # remove the receipt only
```

The App Store From Terminal: mas

```
% brew install mas
% mas list                          # App Store apps installed (with IDs)
% mas outdated                      # apps with updates
% mas upgrade                       # update them all
% mas search Xcode
% mas install <app-id>              # needs you signed in to the App Store

# In a Brewfile:   mas "Xcode", id: <app-id>
```

mas is a community tool, not made by Apple.

Uninstalling Cleanly

- App Store or .dmg apps: drag to Trash (or delete in Launchpad/Apps)
- Leftovers: ~/Library/Application Support, Preferences, Caches, Containers
- Homebrew: brew uninstall <name> (casks: --zap removes settings too)
- MacPorts: sudo port uninstall <name>
- .pkg: pkgutil --files <id> → remove files → pkgutil --forget <id>
- Check for login items, LaunchAgents and LaunchDaemons it added

Package Tools on Other UNIX Systems

Task	FreeBSD	OpenBSD	Solaris 11.4
Search	pkg search nginx	pkg_info -Q nginx	pkg search nginx
Install	pkg install nginx	pkg_add nginx	pkg install nginx
List installed	pkg info	pkg_info	pkg list
Upgrade all	pkg upgrade	pkg_add -u	pkg update
Remove	pkg delete nginx	pkg_delete nginx	pkg uninstall nginx
Build from source	/usr/ports: make install clean	/usr/ports: make install	(source builds are rare)

Which Tool Should You Use?

- Desktop apps: App Store first, then the developer's .dmg, or brew --cask
- UNIX and developer tools: Homebrew (most popular, fastest)
- Need BSD-ports-style builds: MacPorts
- Enterprise or driver software: the vendor's .pkg
- Pick ONE of Homebrew or MacPorts as your main tool
- Write it down: a Brewfile is your machine's recipe

Lab: Build Your Toolbox

```
% /bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
% eval "$(/opt/homebrew/bin/brew shellenv)"
% brew doctor
% brew install wget tree htop coreutils mas
% tree -L 1 /opt/homebrew;  brew leaves
% gls --color=auto;  htop     # q to quit htop
% brew bundle dump --file=~/Brewfile && cat ~/Brewfile
% pkgutil --pkgs | grep -v com.apple
```

Summary & Homework

- App Store, .dmg, .pkg, Homebrew, MacPorts, source
- brew install / upgrade / leaves / autoremove / cleanup / services
- brew bundle dump + install: rebuild a Mac from a Brewfile
- pkgutil inspects .pkg files and their receipts; mas manages App Store apps
- Homework: do the lab; uninstall one cask cleanly with --zap
- Next: Lesson 6 — Compilers & Interpreters

Thank You! 🙏

Free videos & PDFs: www.MyWebUniversity.com

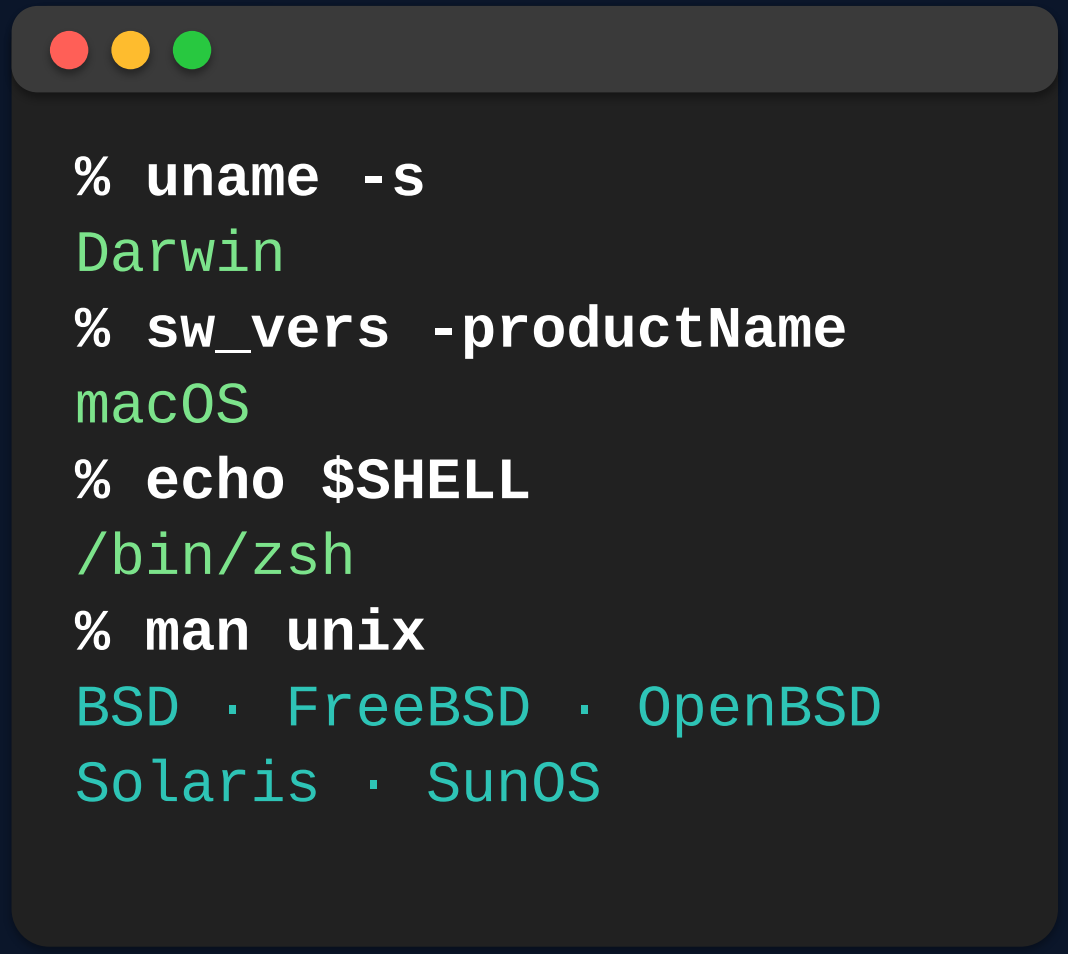
Slides for this lesson: MyWebUniversity.com/presentations

YouTube: @MyWebUniversity · @MyWebUniversity-Dari

Wahidullah Lutfy — Founder and Director of IT of MyWebUniversity, and CS teacher

Book: The Direct Path to Linux Ubuntu (Amazon · Barnes & Noble)

Learn it, then pass it on.



LESSON 6 OF 12

macOS UNIX

Darwin & BSD

Compilers & Interpreters: Xcode Tools, clang, Python, Swift & More

Full Course · macOS 27 Golden Gate

What You Will Learn in Lesson 6

- Compiler vs interpreter, and why it matters
- Xcode vs the Command Line Tools
- C and C++ with clang, make and Makefiles
- Mach-O executables, libraries and universal binaries
- Swift, Python with virtual environments, Perl and shell scripts
- Java, Go, Rust and Node.js with Homebrew; building from source
- Debugging with lldb, and the toolchains of BSD and Solaris

Compiled vs Interpreted

	Compiled	Interpreted / VM
How it runs	Translated to machine code before running	Read and executed by another program at run time
Examples	C, C++, Swift, Go, Rust	Python, Perl, Ruby, JavaScript, zsh; Java (JVM)
Tools on macOS	clang, clang++, swiftc, go, rustc	python3, perl, node, java, zsh
Strength	Speed; a single executable file	Fast to write and test; portable scripts
Check a file	file hello → Mach-O 64-bit executable arm64	head -1 script.py → #!/usr/bin/env python3

Xcode or the Command Line Tools?

```
% xcode-select --install          # Command Line Tools only (about 1 GB)
% xcode-select -p                 # which toolchain is active
/Library/Developer/CommandLineTools

% clang --version;  gcc --version # 'gcc' is really Apple clang
% ls /Library/Developer/CommandLineTools/usr/bin | head

# Full Xcode (App Store): IDE, Simulator, Instruments, iOS SDKs
% sudo xcode-select -s /Applications/Xcode.app/Contents/Developer
% sudo xcodebuild -license accept
```

C: Write, Compile, Run

```
% cat > hello.c <<'EOF'
#include <stdio.h>
#include <sys/utsname.h>
int main(void) {
    struct utsname u;  uname(&u);
    printf("Hello from %s %s on %s\n", u.sysname, u.release, u.machine);
    return 0;
}
EOF
% clang -Wall -O2 -o hello hello.c
% ./hello
Hello from Darwin 26.x.x on arm64
```

C++ and make

```
# Makefile   (recipe lines must start with a TAB)
CXX      = clang++
CXXFLAGS = -std=c++20 -Wall -O2
app: main.o util.o
    $(CXX) $(CXXFLAGS) -o app main.o util.o
%.o: %.cpp
    $(CXX) $(CXXFLAGS) -c $<
clean:
    rm -f *.o app

% make           # builds only what changed
% make clean
```

Inside an Executable: Mach-O

```
% file hello
hello: Mach-O 64-bit executable arm64
% otool -L hello                  # shared libraries it needs
    /usr/lib/libSystem.B.dylib ...

% clang -arch arm64 -arch x86_64 -o hello-uni hello.c
% lipo -info hello-uni            # universal binary
Architectures in the fat file: hello-uni are: x86_64 arm64

% codesign -dv hello 2>&1 | head -3   # Apple silicon needs a signature
```

Swift: Apple's Modern Language

```
% swift --version
% echo 'print("Hello, MyWebUniversity!")' > hello.swift
% swiftc -O hello.swift -o hello-swift && ./hello-swift

% mkdir Tool && cd Tool
% swift package init --type executable
% swift run                          # build and run the package
% swift build -c release             # optimized build in .build/release
```

Python the Right Way

```
% which -a python3
/opt/homebrew/bin/python3      # Homebrew (current version)
/usr/bin/python3               # Apple's, from the Command Line Tools

% mkdir proj && cd proj
% python3 -m venv .venv          # a private environment per project
% source .venv/bin/activate
(.venv) % pip install requests
(.venv) % python -c 'import platform; print(platform.platform())'
(.venv) % deactivate
```

Scripting: zsh, bash, Perl

```
#!/bin/zsh
# disk-alert.zsh — warn when / is over 80% full
use=$(df -h / | awk 'NR==2 {gsub("%","",$5); print $5}')
if (( use > 80 )); then
  osascript -e "display notification \"Disk ${use}% full\" with title \"Alert\""
fi

% chmod +x disk-alert.zsh && ./disk-alert.zsh
% perl -ne 'print if /error/i' /var/log/install.log | head
% /usr/bin/env python3 --version   # portable shebang: #!/usr/bin/env python3
```

More Languages With Homebrew

Language	Install	Check	Hello world
Java (OpenJDK)	brew install openjdk	java -version	java Hello.java
Go	brew install go	go version	go run hello.go
Rust	curl https://sh.rustup.rs -sSf | sh	rustc --version	cargo new app && cargo run
Node.js	brew install node	node -v; npm -v	node hello.js
Ruby / Perl (current)	brew install ruby perl	ruby -v; perl -v	ruby -e 'puts 1'

Homebrew prints 'caveats' (e.g. PATH or symlink steps for openjdk): read them.

Build From Source: The Classic Way

```
% curl -LO https://ftp.gnu.org/gnu/hello/hello-2.12.1.tar.gz
% tar xzf hello-2.12.1.tar.gz && cd hello-2.12.1
% ./configure --prefix=$HOME/local    # check the system, write Makefiles
% make -j4                            # compile with 4 jobs
% make check                          # run the tests
% make install                        # copy to ~/local (no sudo needed)
% ~/local/bin/hello
Hello, world!
```

Debugging With lldb

```
% clang -g -O0 -o hello hello.c   # -g = debug information
% lldb ./hello
(lldb) breakpoint set --name main  # or: b main
(lldb) run
(lldb) next                        # step over a line
(lldb) frame variable              # show local variables
(lldb) bt                          # backtrace: who called whom
(lldb) quit
```

Toolchains on Other UNIX Systems

	macOS	FreeBSD	OpenBSD	Solaris 11.4
C compiler	Apple clang (CLT)	clang in base	clang in base	gcc (pkg) / Developer Studio
make	GNU make 3.81 (+ brew gmake)	BSD make; gmake from pkg	BSD make; gmake from pkg	make; gmake
Debugger	lldb	lldb, gdb (pkg)	lldb / gdb (pkg)	dbx, gdb
Executable format	Mach-O	ELF	ELF	ELF
Tracing	DTrace	DTrace	btrace, ktrace	DTrace

Lab: Your Developer Workstation

```
% xcode-select --install;  xcode-select -p
% clang -Wall -O2 -o hello hello.c && ./hello
% file hello;  otool -L hello
% clang -arch arm64 -arch x86_64 -o hello-uni hello.c && lipo -info hello-uni
% echo 'print("Hello")' > h.swift && swiftc h.swift -o h && ./h
% python3 -m venv ~/proj/.venv && source ~/proj/.venv/bin/activate
% pip install requests && python -c 'import requests; print(requests.__version__)'
% deactivate;  brew install go && go version
```

Summary & Homework

- Compiled (C, C++, Swift, Go, Rust) vs interpreted (Python, Perl, shell)
- xcode-select --install gives clang, make, git, swift, python3
- clang -Wall -O2; Makefiles; Mach-O, otool, lipo, codesign
- Python: always use a venv; Homebrew for Java, Go, Rust, Node
- Homework: do the lab; write disk-alert.zsh; debug hello with lldb
- Next: Lesson 7 — Databases: SQLite, PostgreSQL & MySQL

Thank You! 🙏

Free videos & PDFs: www.MyWebUniversity.com

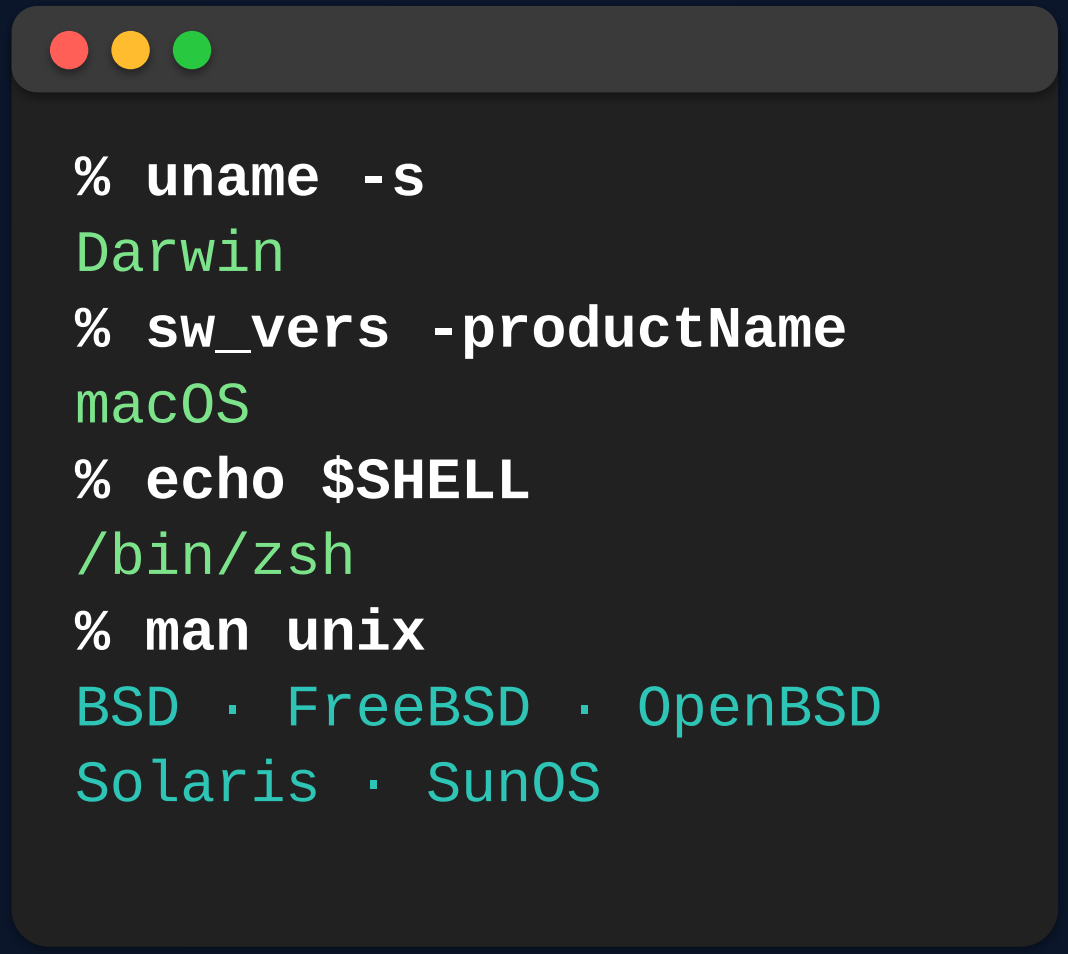
Slides for this lesson: MyWebUniversity.com/presentations

YouTube: @MyWebUniversity · @MyWebUniversity-Dari

Wahidullah Lutfy — Founder and Director of IT of MyWebUniversity, and CS teacher

Book: The Direct Path to Linux Ubuntu (Amazon · Barnes & Noble)

Learn it, then pass it on.



LESSON 7 OF 12

macOS UNIX

Darwin & BSD

Databases on macOS: SQLite, PostgreSQL & MySQL

Full Course · macOS 27 Golden Gate

What You Will Learn in Lesson 7

- Embedded vs client-server databases
- SQLite: built into macOS, used by macOS itself
- SQL essentials: create, insert, select, update, join
- PostgreSQL with Homebrew: install, users, databases, psql
- MySQL with Homebrew: install, secure, users, the mysql client
- Services, ports, backups, restores and GUI clients
- Databases on FreeBSD, OpenBSD and Solaris

Two Kinds of Databases

	Embedded (SQLite)	Client-server (PostgreSQL, MySQL)
Runs as	A library inside your program	A separate server process (daemon)
Data lives in	One file, e.g. school.db	A data directory managed by the server
Users & network	No users; file permissions only	Users, passwords, roles; TCP port 5432 / 3306
Best for	Apps, tools, small sites, learning	Web apps, many users at once, big data
On macOS	Built in: /usr/bin/sqlite3	Install with Homebrew

SQLite: Already on Your Mac

- sqlite3 ships with macOS in /usr/bin
- Many macOS apps store data in SQLite files
- A database is a single, portable file: copy it, email it, back it up
- Same file format on macOS, Linux, BSD, Windows, iPhone, Android
- Python includes it too: import sqlite3
- Never edit a live app's database; work on a copy

SQLite Hands-On

```
% sqlite3 ~/lab/school.db
sqlite> CREATE TABLE students (id INTEGER PRIMARY KEY, name TEXT, city TEXT);
sqlite> INSERT INTO students (name, city) VALUES ('Maryam','Kabul'), ('Ahmad','Herat');
sqlite> .headers on
sqlite> .mode column
sqlite> SELECT * FROM students;
id  name    city
1   Maryam  Kabul
2   Ahmad   Herat
sqlite> .tables      .schema students      .quit
```

SQL Essentials

```
CREATE TABLE courses (id INTEGER PRIMARY KEY, title TEXT);
CREATE TABLE enroll  (student_id INT, course_id INT);
INSERT INTO courses (title) VALUES ('macOS UNIX'), ('Python');
INSERT INTO enroll VALUES (1,1), (1,2), (2,1);

SELECT name, city FROM students WHERE city = 'Kabul' ORDER BY name;
UPDATE students SET city = 'Mazar' WHERE id = 2;
DELETE FROM enroll WHERE student_id = 2 AND course_id = 1;

SELECT s.name, c.title FROM students s
  JOIN enroll e  ON e.student_id = s.id
  JOIN courses c ON c.id = e.course_id;
```

PostgreSQL With Homebrew

```
% brew search postgresql           # choose a current version
% brew install postgresql@17
% brew services start postgresql@17 # starts now and at login
% echo 'export PATH="/opt/homebrew/opt/postgresql@17/bin:$PATH"' >> ~/.zprofile
% source ~/.zprofile

% psql postgres                    # your macOS user is the superuser
postgres=# SELECT version();
postgres=# \q
```

Version 17 is an example. Read the caveats brew prints after installing.

PostgreSQL: Users and Databases

```
% createuser --pwprompt webapp        # a login role with a password
% createdb --owner=webapp school
% psql -U webapp -d school -h localhost
school=> CREATE TABLE students (id SERIAL PRIMARY KEY, name TEXT);
school=> \dt                            -- list tables
school=> \l                             -- list databases
school=> \du                            -- list roles (users)
school=> \conninfo                      -- who and where am I?
school=> \q
```

MySQL With Homebrew

```
% brew install mysql                    # or: brew install mariadb (pick one)
% brew services start mysql
% mysql_secure_installation             # set root password, remove test users
% mysql -u root -p
mysql> SELECT VERSION();
mysql> CREATE DATABASE school;
mysql> CREATE USER 'webapp'@'localhost' IDENTIFIED BY 'Strong#Pass1';
mysql> GRANT ALL PRIVILEGES ON school.* TO 'webapp'@'localhost';
mysql> SHOW DATABASES;   exit
```

Services, Ports and Data Folders

```
% brew services list
Name           Status  User   File
mysql          started wahid  ~/Library/LaunchAgents/homebrew.mxcl.mysql.plist
postgresql@17  started wahid  ~/Library/LaunchAgents/homebrew.mxcl.postgresql@17.plist

% lsof -iTCP -sTCP:LISTEN -n -P | grep -E 'postgres|mysqld'
postgres ... TCP 127.0.0.1:5432 (LISTEN)
mysqld   ... TCP 127.0.0.1:3306 (LISTEN)

% ls /opt/homebrew/var            # data: postgresql@17/, mysql/
```

Example output.

Backup and Restore

```
# SQLite
% sqlite3 ~/lab/school.db .dump > school-sqlite.sql
% sqlite3 restored.db < school-sqlite.sql

# PostgreSQL
% pg_dump -U webapp -h localhost school > school-pg.sql
% createdb school_test && psql -d school_test -f school-pg.sql

# MySQL
% mysqldump -u root -p --databases school > school-my.sql
% mysql -u root -p < school-my.sql
```

Graphical Database Tools

Tool	Databases	Install
DB Browser for SQLite	SQLite	brew install --cask db-browser-for-sqlite
pgAdmin 4	PostgreSQL	brew install --cask pgadmin4
MySQL Workbench	MySQL	brew install --cask mysqlworkbench
DBeaver Community	All of them (and more)	brew install --cask dbeaver-community

Database Security Checklist

- Listen on localhost only unless other machines truly need access
- Run mysql_secure_installation; never leave root without a password
- One user per application, with rights to its own database only
- Strong passwords; never put them in scripts you share (use ~/.pgpass, ~/.my.cnf with chmod 600)
- Automated, tested backups — copied off the Mac
- Update regularly: brew upgrade, and read the release notes

Databases on Other UNIX Systems

Step	FreeBSD	OpenBSD	Solaris 11.4
Install	pkg install postgresql17-server	pkg_add postgresql-server	pkg install a database/postgres or mysql package
Initialize	service postgresql initdb	su _postgresql -c 'initdb ...'	per package instructions
Enable at boot	sysrc postgresql_enable=YES	rcctl enable postgresql	svcadm enable <service>
Start / status	service postgresql start / status	rcctl start / check postgresql	svcs <service>

Package names and versions change: use pkg search / pkg_info -Q / pkg search first.

Lab: A Student Database Three Ways

```
% mkdir -p ~/lab && sqlite3 ~/lab/school.db < students.sql
% sqlite3 -header -column ~/lab/school.db 'SELECT * FROM students;'

% brew install postgresql@17 && brew services start postgresql@17
% createuser --pwprompt webapp && createdb --owner=webapp school
% psql -U webapp -h localhost school -c 'CREATE TABLE t(x int); \dt'
% pg_dump -U webapp -h localhost school > ~/lab/school-pg.sql

% lsof -iTCP -sTCP:LISTEN -n -P;   brew services list
```

students.sql = the CREATE and INSERT lines from the SQLite slides.

Summary & Homework

- SQLite = one file, built in; PostgreSQL and MySQL = servers on 5432 / 3306
- SQL: CREATE, INSERT, SELECT … WHERE, UPDATE, DELETE, JOIN
- brew install + brew services start; one user per application
- Backups: .dump, pg_dump, mysqldump — and test the restore
- Homework: do the lab; add a courses table and write a JOIN
- Next: Lesson 8 — Users, Groups, sudo & Directory Services

Thank You! 🙏

Free videos & PDFs: www.MyWebUniversity.com

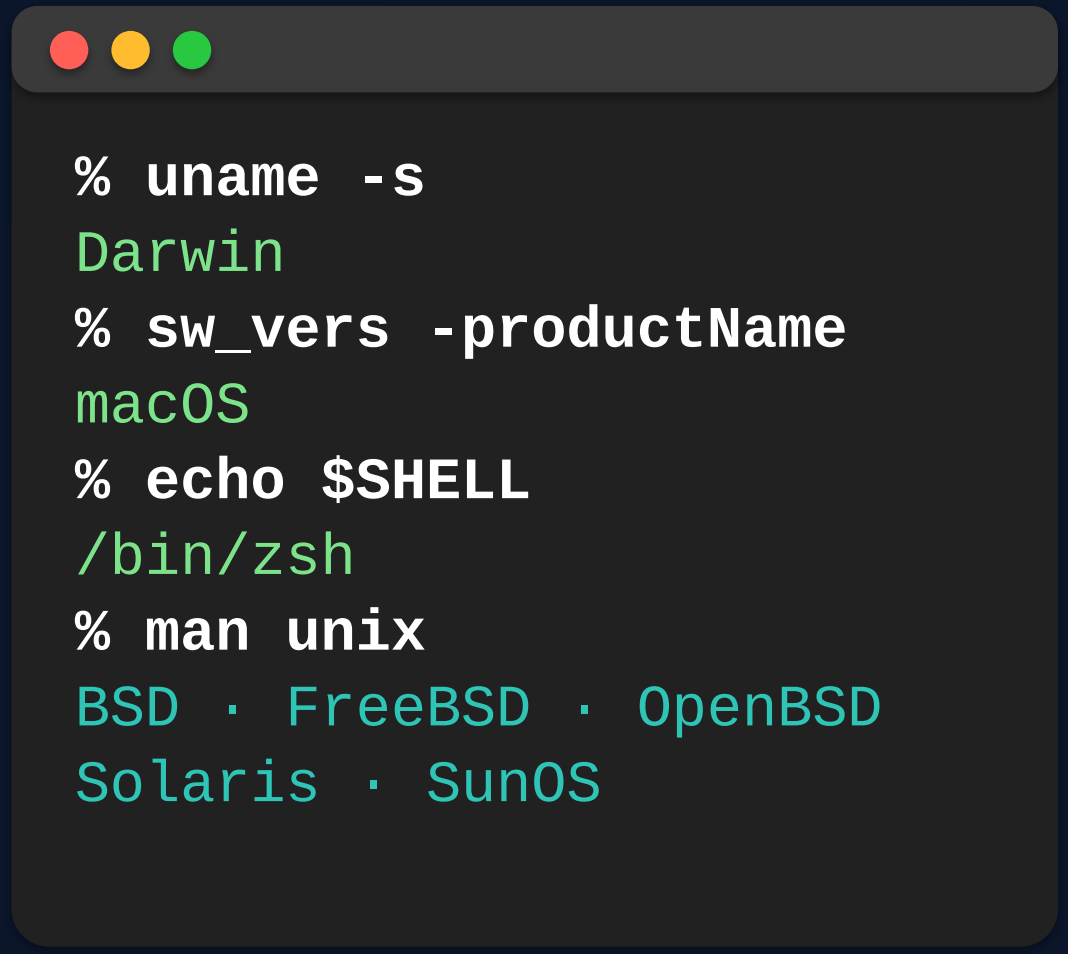
Slides for this lesson: MyWebUniversity.com/presentations

YouTube: @MyWebUniversity · @MyWebUniversity-Dari

Wahidullah Lutfy — Founder and Director of IT of MyWebUniversity, and CS teacher

Book: The Direct Path to Linux Ubuntu (Amazon · Barnes & Noble)

Learn it, then pass it on.



LESSON 8 OF 12

macOS UNIX

Darwin & BSD

Account Management: Users, Groups, sudo & Directory Services

Full Course · macOS 27 Golden Gate

What You Will Learn in Lesson 8

- Account types: administrator, standard, sharing only, guest, root
- Users & Groups in System Settings
- Directory Services: where macOS really keeps accounts
- dscl, sysadminctl and dseditgroup from Terminal
- sudo, /etc/sudoers.d and Touch ID for sudo
- Secure Tokens, volume owners and FileVault
- Accounts on FreeBSD, OpenBSD and Solaris

macOS Account Types

Type	Can do	Use it for
Administrator	Install software, change settings, use sudo	You — but not for daily browsing on shared Macs
Standard	Use apps, change own settings	Daily work, students, family
Sharing Only	Connect to shared files; no login at the Mac	File server access
Guest	Temporary login; files erased at logout	Visitors (off by default)
root	Everything; disabled by default	Use sudo instead

Users & Groups in System Settings

- System Settings → Users & Groups → Add User (admin password required)
- Choose Standard or Administrator; set name, account name, password
- Click ⓘ on a user: change type, reset password, Advanced Options
- Advanced Options: UID, group, account name, login shell, home folder
- Add Group: create groups and tick their members
- Login options: automatic login off, Guest User off

Directory Services: Where Accounts Live

```
% id
uid=501(wahid) gid=20(staff) groups=20(staff),80(admin),12(everyone),...

% dscl . -list /Users UniqueID | awk '$2 >= 500'   # real people
wahid      501
% dscl . -list /Users | head -3                    # system accounts start with _
_amavisd  _analyticsd  _appleevents
% dscl . -read /Users/wahid NFSHomeDirectory UserShell RealName

% sudo ls /var/db/dslocal/nodes/Default/users | head   # one .plist per user
```

/etc/passwd exists but macOS does not use it for normal logins.

Create, Change and Delete Users: sysadminctl

```
% sudo sysadminctl -addUser student1 -fullName "Student One" -password -
   (type the new password when asked)
% sudo sysadminctl -addUser labadmin -fullName "Lab Admin" -password - -admin

% sudo sysadminctl -resetPasswordFor student1 -newPassword -
% sudo dscl . -create /Users/student1 UserShell /bin/bash   # change shell
% sysadminctl -secureTokenStatus student1

% sudo sysadminctl -deleteUser student1 -keepHome     # keep files
% sudo sysadminctl -deleteUser student1 -secure       # erase home
```

Groups: dseditgroup

```
% dscl . -list /Groups | grep -v '^_' | head
admin  everyone  staff  wheel ...

% sudo dseditgroup -o create -r "Lab Students" labstudents
% sudo dseditgroup -o edit -a student1 -t user labstudents
% dseditgroup -o checkmember -m student1 labstudents
yes student1 is a member of labstudents

% sudo dseditgroup -o edit -a student1 -t user admin    # make admin
% sudo dseditgroup -o edit -d student1 -t user admin    # remove admin
% groups student1;  id student1
```

sudo: Borrowing Root's Power Safely

```
% sudo -l                              # what may I run?
% sudo -k                              # forget my cached password now
% sudo cat /etc/sudoers | grep -v '^#' | grep -v '^$'
root    ALL = (ALL) ALL
%admin  ALL = (ALL) ALL
#includedir /private/etc/sudoers.d

# Let labstudents restart Apache only — always edit with visudo:
% sudo visudo -f /etc/sudoers.d/labstudents
%labstudents ALL = (root) /usr/sbin/apachectl restart
```

Touch ID for sudo and Account Auditing

```
# Touch ID for sudo (survives macOS updates):
% sudo cp /etc/pam.d/sudo_local.template /etc/pam.d/sudo_local
% sudo vim /etc/pam.d/sudo_local      # remove the # before:
auth       sufficient     pam_tid.so

# Who is here, and who was here?
% who;  w                             # logged-in users
% last | head                         # recent logins and reboots
% dscl . -read /Groups/admin GroupMembership   # who is an admin?
```

Secure Tokens, Volume Owners and FileVault

- Secure Token: lets an account unlock FileVault and create other token users
- The first admin created in Setup Assistant gets a Secure Token
- Apple silicon: Volume Owners may change startup security and install updates
- Check: sysadminctl -secureTokenStatus <user>
- Check: diskutil apfs listUsers /
- Create users from a token-holding admin, or they may not unlock the disk

Home Folders and Account Hygiene

- Home folder: /Users/<shortname> with Desktop, Documents, Library …
- Default permissions keep other users out (drwxr-x---+)
- /Users/Shared: a folder every user can use
- Disable, don't delete, while you investigate: sudo pwpolicy -u <user> -disableuser
- Remove admin rights you no longer need
- Enterprise Macs: accounts and password rules come from MDM

Accounts on Other UNIX Systems

Task	FreeBSD	OpenBSD	Solaris 11.4
Add user	adduser or pw useradd -n bob -m	adduser or useradd -m bob	useradd -m bob
Set password	passwd bob	passwd bob	passwd bob
Admin group	pw groupmod wheel -m bob	usermod -G wheel bob	RBAC profiles / roles
Root tool	sudo (pkg) or su	doas (/etc/doas.conf)	su - (root is a role), pfexec, sudo
Account files	/etc/master.passwd (vipw)	/etc/master.passwd (vipw)	/etc/passwd, /etc/shadow

Lab: A Classroom of Users

```
% sudo dseditgroup -o create -r "Lab Students" labstudents
% for n in 1 2 3; do
>   sudo sysadminctl -addUser student$n -fullName "Student $n" -password "Temp#Pass$n"
>   sudo dseditgroup -o edit -a student$n -t user labstudents
> done
% dscl . -read /Groups/labstudents GroupMembership
% dscl . -list /Users UniqueID | awk '$2 >= 500'
% sysadminctl -secureTokenStatus student1
% sudo sysadminctl -deleteUser student3 -secure
```

Lab only: never put real passwords in a script.

Summary & Homework

- Admin, Standard, Sharing Only, Guest; root is disabled — use sudo
- Accounts live in Directory Services: dscl . -list / -read
- sysadminctl creates, resets, deletes; dseditgroup manages groups
- visudo + /etc/sudoers.d for least-privilege rules; Touch ID via sudo_local
- Homework: do the lab; give labstudents one safe sudo command
- Next: Lesson 9 — Processes, Services & launchd

Thank You! 🙏

Free videos & PDFs: www.MyWebUniversity.com

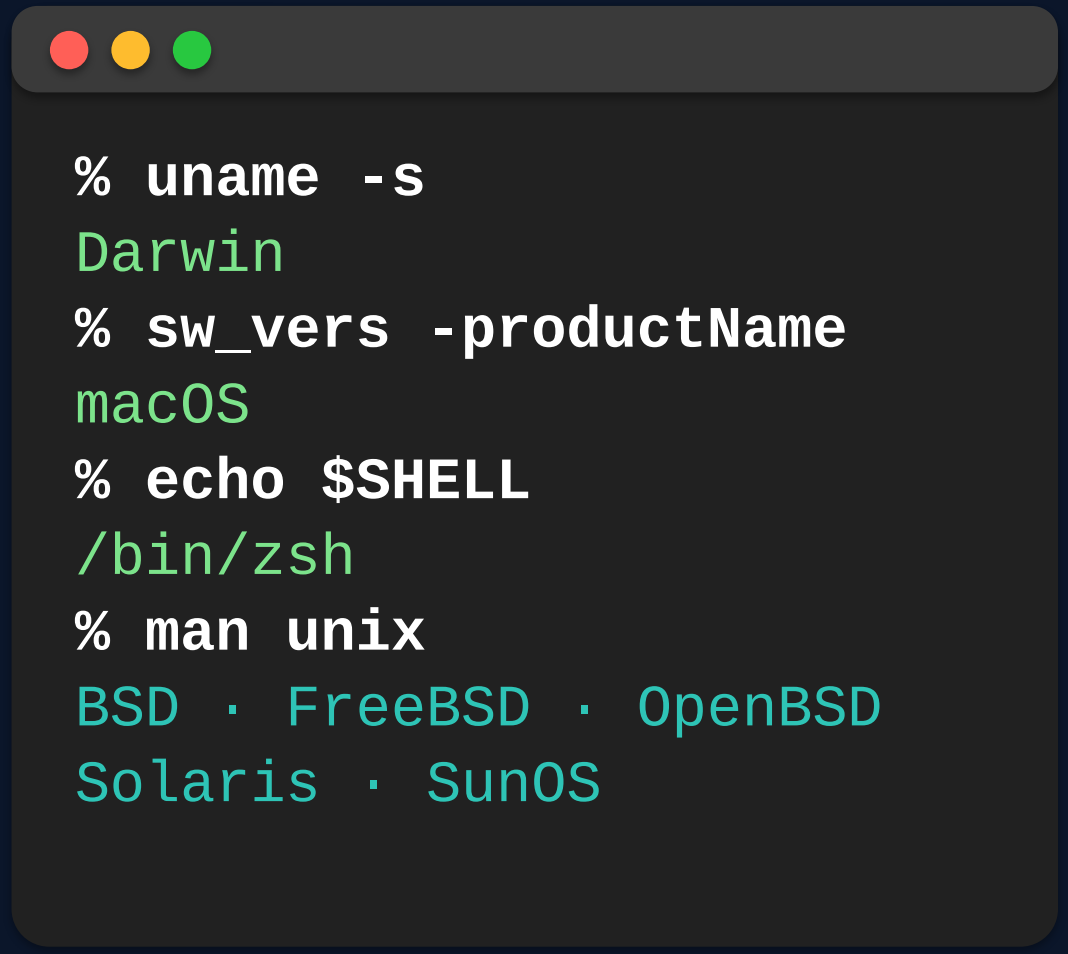
Slides for this lesson: MyWebUniversity.com/presentations

YouTube: @MyWebUniversity · @MyWebUniversity-Dari

Wahidullah Lutfy — Founder and Director of IT of MyWebUniversity, and CS teacher

Book: The Direct Path to Linux Ubuntu (Amazon · Barnes & Noble)

Learn it, then pass it on.



LESSON 9 OF 12

macOS UNIX

Darwin & BSD

Process & Service Management: ps, top, kill & launchd

Full Course · macOS 27 Golden Gate

What You Will Learn in Lesson 9

- Processes, PIDs, parents and launchd (PID 1)
- ps, top, pgrep and Activity Monitor
- Signals: kill, pkill, killall; jobs, nice and nohup
- Who uses what: lsof, vm_stat, fs_usage
- launchd: daemons vs agents, and the plist format
- launchctl: bootstrap, bootout, print, kickstart; launchd vs cron
- Unified logging, and services on FreeBSD, OpenBSD and Solaris

Processes and launchd, PID 1

```
% ps -o pid,ppid,user,command -p 1
  PID  PPID USER  COMMAND
    1     0 root  /sbin/launchd

% ps aux | head -5                    # every process, BSD style
% ps -ef | grep -i finder              # System V style also works
% ps -o pid,ppid,%cpu,%mem,command -U $USER | head
% pgrep -lf Safari                     # PIDs by name
% sysctl hw.ncpu hw.memsize            # how many CPUs, how much RAM
```

top: Live View

```
% top -o cpu                    # sort by CPU (q to quit)
% top -o mem -n 10              # top 10 by memory
% top -l 1 | head -12          # one snapshot, good for scripts
Processes: 512 total, 3 running, 509 sleeping, 2201 threads
Load Avg: 1.52, 1.31, 1.20  CPU usage: 6.1% user, 4.0% sys, 89.8% idle
PhysMem: 7210M used (1302M wired), 802M unused.

% vm_stat;  memory_pressure  # memory details
% uptime                     # load averages: 1, 5, 15 minutes
```

Example output.

Signals: Talking to Processes

Signal	Number	Meaning	Example
TERM	15	Please exit (default; can clean up)	kill 4242
INT	2	Interrupt (what Ctrl-C sends)	kill -INT 4242
HUP	1	Hang up; many daemons reload config	kill -HUP 4242
KILL	9	Stop immediately (no clean-up!)	kill -9 4242
STOP / CONT	17 / 19	Pause / resume	kill -STOP 4242

Name-based: pkill -x Safari · killall Safari · pkill -u student1 · list all: kill -l

Jobs, Priority and Background Work

```
% sleep 300 &                    # run in the background
[1] 5120
% jobs;  fg %1;  ^Z;  bg %1     # list, foreground, suspend, background

% nice -n 10 ./backup.sh        # start with lower priority (-20..20)
% sudo renice -n 5 -p 5120      # change a running process

% nohup ./long-job.sh > job.log 2>&1 &   # survives closing Terminal
% caffeinate -i ./long-job.sh            # keep the Mac awake meanwhile
```

Who Uses What? lsof, fs_usage, sample

```
% lsof -p 5120 | head              # files and sockets a process has open
% lsof -i :8080                   # who uses port 8080?
% lsof +D ~/lab                   # who has files open in a folder
% sudo lsof /Volumes/MyUSB        # why can't I eject this disk?

% sudo fs_usage -w -f filesys Safari | head   # live file activity
% sample Safari 5 -file /tmp/safari.txt      # what is it doing? (5 s)
% sudo spindump                              # report on hung apps
```

launchd: Daemons vs Agents

Type	Runs as	Starts	Folder
LaunchDaemon (Apple)	root or a service user	At boot, no login needed	/System/Library/LaunchDaemons
LaunchDaemon (yours)	root or UserName	At boot	/Library/LaunchDaemons
LaunchAgent (all users)	Each logged-in user	At login	/Library/LaunchAgents
LaunchAgent (one user)	That user	At that user's login	~/Library/LaunchAgents

Never edit /System: it is on the sealed volume. Your own jobs go in /Library or ~/Library.

A launchd Job: ~/Library/LaunchAgents/com.mwu.diskalert.plist

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-
1.0.dtd">
<plist version="1.0"><dict>
  <key>Label</key>             <string>com.mwu.diskalert</string>
  <key>ProgramArguments</key>  <array><string>/Users/wahid/bin/disk-alert.zsh</string></array>
  <key>StartCalendarInterval</key> <dict><key>Hour</key><integer>9</integer>
                                   <key>Minute</key><integer>0</integer></dict>
  <key>RunAtLoad</key>         <true/>
  <key>StandardOutPath</key>   <string>/tmp/diskalert.out</string>
  <key>StandardErrorPath</key> <string>/tmp/diskalert.err</string>
</dict></plist>
```

Other keys: StartInterval (seconds) · KeepAlive (restart if it exits) · WorkingDirectory · UserName (daemons)

launchctl: Load, Inspect, Run, Unload

```
% plutil -lint ~/Library/LaunchAgents/com.mwu.diskalert.plist   # check syntax
% launchctl bootstrap gui/$(id -u) ~/Library/LaunchAgents/com.mwu.diskalert.plist
% launchctl list | grep mwu                      # PID, last exit status, label
% launchctl print gui/$(id -u)/com.mwu.diskalert | head -20
% launchctl kickstart -k gui/$(id -u)/com.mwu.diskalert   # run (restart) now
% launchctl bootout gui/$(id -u)/com.mwu.diskalert         # unload

# Daemons use the system domain (plist owned by root:wheel, mode 644):
% sudo launchctl bootstrap system /Library/LaunchDaemons/com.mwu.web.plist
```

launchd vs cron

- cron still works on macOS: crontab -e (min hour day month weekday)
- Apple recommends launchd for new jobs
- launchd runs a missed calendar job when the Mac wakes; cron skips it
- launchd can restart crashed programs (KeepAlive) and start on demand
- Output, environment and user are set per job in the plist
- brew services writes launchd plists for you (Lesson 5, 7)

Unified Logging: log and Console

```
% logger "MWU test message"                    # write to the system log
% log show --last 5m --predicate 'eventMessage CONTAINS "MWU"'
% log show --last 1h --predicate 'process == "sshd"' --style compact
% log stream --predicate 'process == "launchd"' --level info   # live
% log show --last boot | grep -i error | head

% ls /var/log;  tail -f /tmp/diskalert.err      # classic text logs
```

Services on Other UNIX Systems

Task	macOS	FreeBSD	OpenBSD	Solaris 11.4
PID 1	launchd	init	init	init + SMF
Enable at boot	plist in LaunchDaemons	sysrc nginx_enable=YES	rcctl enable nginx	svcadm enable nginx
Start / stop	launchctl bootstrap / bootout	service nginx start / stop	rcctl start / stop nginx	svcadm enable -t / disable
Status	launchctl print	service nginx status	rcctl check nginx	svcs -x nginx
Process tools	ps, top, lsof	ps, top, procstat	ps, top, fstat	ps, prstat, ptree, pfiles

Lab: Your First launchd Service

```
% mkdir -p ~/bin && cp ~/lab/disk-alert.zsh ~/bin/ && chmod +x ~/bin/disk-alert.zsh
% vim ~/Library/LaunchAgents/com.mwu.diskalert.plist    # from the slide
% plutil -lint ~/Library/LaunchAgents/com.mwu.diskalert.plist
% launchctl bootstrap gui/$(id -u) ~/Library/LaunchAgents/com.mwu.diskalert.plist
% launchctl list | grep mwu;  cat /tmp/diskalert.err
% sleep 600 &
% pgrep -lf sleep;  kill %1
% top -l 1 -o cpu | head -15;  lsof -iTCP -sTCP:LISTEN -n -P
% launchctl bootout gui/$(id -u)/com.mwu.diskalert
```

Summary & Homework

- launchd is PID 1; ps, top, pgrep, lsof show what runs
- Signals: TERM first, KILL last; pkill / killall by name
- Daemons (boot, root) vs agents (login, user) — choose the folder
- launchctl bootstrap / print / kickstart / bootout; plutil -lint first
- Homework: do the lab; change the job to run every 30 minutes
- Next: Lesson 10 — Network Configuration, Firewall & SSH

Thank You! 🙏

Free videos & PDFs: www.MyWebUniversity.com

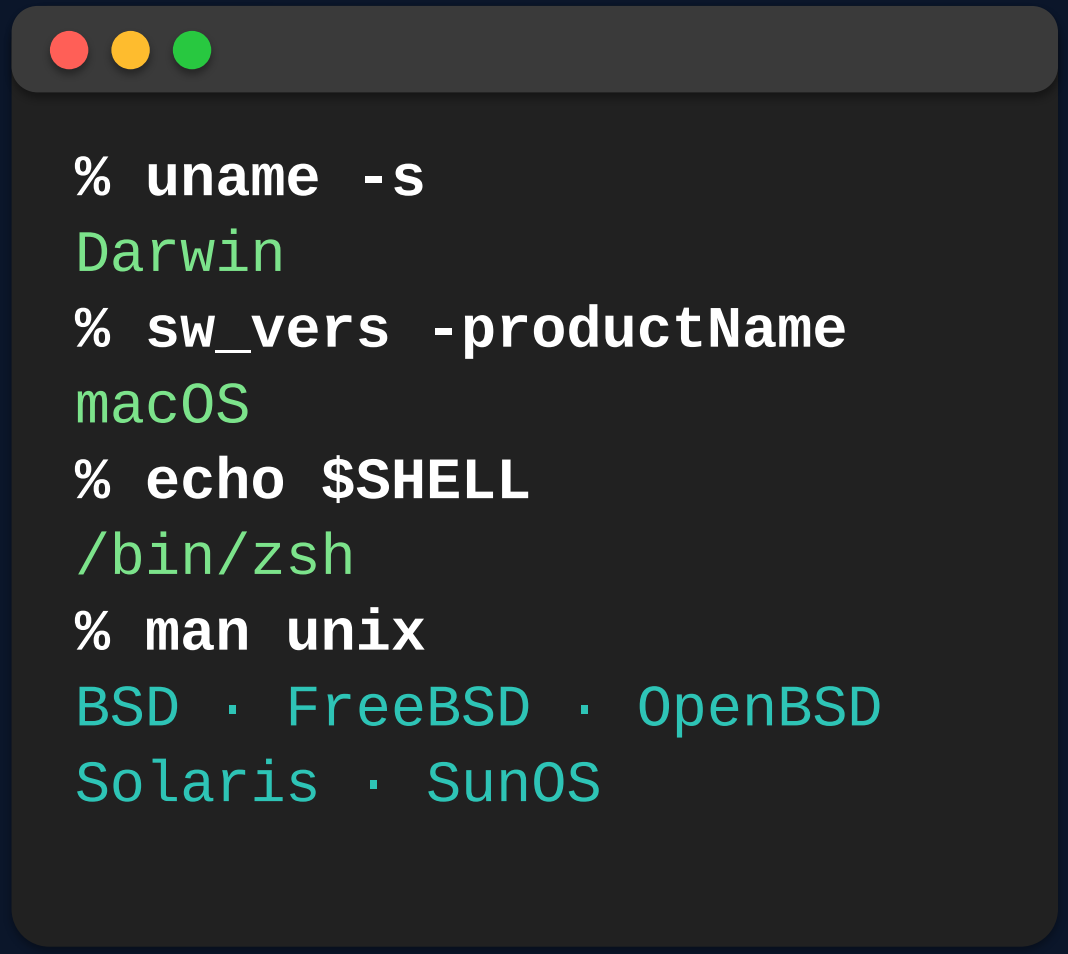
Slides for this lesson: MyWebUniversity.com/presentations

YouTube: @MyWebUniversity · @MyWebUniversity-Dari

Wahidullah Lutfy — Founder and Director of IT of MyWebUniversity, and CS teacher

Book: The Direct Path to Linux Ubuntu (Amazon · Barnes & Noble)

Learn it, then pass it on.



LESSON 10 OF 12

macOS UNIX

Darwin & BSD

Network Configuration, Firewall & SSH

Full Course · macOS 27 Golden Gate

What You Will Learn in Lesson 10

- Network services, interfaces and the Network settings panel
- networksetup: DHCP, static IP and DNS from Terminal
- ifconfig, ipconfig, netstat, route, scutil and hostnames
- Troubleshooting: ping, traceroute, dig, nc, curl, networkQuality
- The Application Firewall and the pf packet filter
- SSH: Remote Login, keys, config, hardening, scp and rsync
- Networking on FreeBSD, OpenBSD and Solaris

Interfaces and Network Services

```
% networksetup -listallhardwareports
Hardware Port: Ethernet      Device: en0
Hardware Port: Wi-Fi         Device: en1
% networksetup -listallnetworkservices
% networksetup -getinfo "Ethernet"       # IP, mask, router, DHCP or manual

% ifconfig en0 | grep -E 'ether|inet '     # MAC and IP addresses
% ipconfig getifaddr en0                   # just the IPv4 address
% route -n get default | grep gateway      # default router
```

Example: device names vary (a UTM VM usually shows Ethernet en0).

Network Settings in the GUI

Where	What you set
System Settings → Network → (service) → Details	TCP/IP: DHCP or manual, router, IPv6
… → Details → DNS	DNS servers and search domains
… → Details → Hardware	MAC address, speed, MTU
Network → ⋯ → Set Service Order	Which connection is used first
Network → Firewall	Application Firewall on/off, options
Wi-Fi → Details	Private Wi-Fi address, Low Data Mode, forget network

networksetup: Configure From Terminal

```
% sudo networksetup -setmanual "Ethernet" 192.168.64.50 255.255.255.0 192.168.64.1
% sudo networksetup -setdnsservers "Ethernet" 1.1.1.1 9.9.9.9
% networksetup -getdnsservers "Ethernet"

% sudo networksetup -setdhcp "Ethernet"           # back to automatic
% sudo networksetup -setdnsservers "Ethernet" Empty

% networksetup -setairportpower en1 off           # Wi-Fi off / on
% sudo networksetup -setv6automatic "Wi-Fi"
```

Use addresses from your own network. In UTM's default shared network, DHCP is easiest.

Names: Hostnames and DNS

```
% scutil --get ComputerName;  scutil --get LocalHostName;  hostname
% sudo scutil --set ComputerName "MWU Lab Mac"     # shown in Finder, sharing
% sudo scutil --set LocalHostName mwu-lab          # Bonjour: mwu-lab.local
% sudo scutil --set HostName mwu-lab.example.com   # UNIX hostname

% scutil --dns | grep nameserver | sort -u         # resolvers in use
% dig +short www.MyWebUniversity.com
% dscacheutil -q host -a name www.apple.com        # lookup the macOS way
% sudo dscacheutil -flushcache; sudo killall -HUP mDNSResponder   # flush DNS
```

Troubleshooting From the Bottom Up

```
% ifconfig en0 | grep 'status\|inet '   # 1. link up? have an IP?
% ping -c 3 $(route -n get default | awk '/gateway/{print $2}')  # 2. router
% ping -c 3 1.1.1.1                     # 3. the internet by number
% dig +short apple.com                  # 4. DNS works?
% nc -vz github.com 443                 # 5. is the port open?
% curl -I https://www.MyWebUniversity.com # 6. does the service answer?
% traceroute -n 1.1.1.1                 # where does it stop?
% networkQuality                        # speed and responsiveness
```

Ports and Connections: netstat and lsof

```
% netstat -rn -f inet | head              # routing table
% netstat -an -p tcp | grep LISTEN       # listening TCP ports
% lsof -iTCP -sTCP:LISTEN -n -P           # ...with program names
% lsof -i -n -P | grep ESTABLISHED | head # current connections
% arp -a                                  # neighbors on the LAN
% sudo tcpdump -i en0 -n port 53 -c 10    # watch DNS packets
```

The Application Firewall

```
% FW=/usr/libexec/ApplicationFirewall/socketfilterfw
% sudo $FW --getglobalstate
Firewall is disabled. (State = 0)
% sudo $FW --setglobalstate on
% sudo $FW --setstealthmode on            # don't answer pings/probes
% sudo $FW --listapps
% sudo $FW --add /opt/homebrew/bin/nginx  # allow an app
% sudo $FW --blockapp /Applications/Unwanted.app
```

pf: The Packet Filter From OpenBSD

```
% sudo pfctl -s info | head -3            # is pf enabled?
% grep anchor /etc/pf.conf                # macOS loads com.apple/* anchors

% echo 'block drop in quick proto tcp to any port 8080' | sudo tee /etc/pf.anchors/mwu
% sudo pfctl -a com.apple/mwu -f /etc/pf.anchors/mwu   # load our rules
% sudo pfctl -E                           # enable pf (with a reference)
% sudo pfctl -a com.apple/mwu -s rules     # show them

% sudo pfctl -a com.apple/mwu -F rules     # flush our rules when done
```

Test pf on your VM first: a wrong rule can cut off your SSH session.

SSH: Remote Login and Keys

```
# On the Mac you want to reach (or: General → Sharing → Remote Login):
% sudo systemsetup -setremotelogin on      # Terminal may need Full Disk Access
% sudo systemsetup -getremotelogin

# On your own computer:
% ssh-keygen -t ed25519 -C "wahid@mwu-lab"
% ssh-copy-id wahid@mwu-lab.local
% ssh wahid@mwu-lab.local
% ssh-add --apple-use-keychain ~/.ssh/id_ed25519   # passphrase in Keychain
```

SSH Config, Hardening and File Transfer

```
# ~/.ssh/config  (client)
Host lab
    HostName mwu-lab.local
    User wahid
    UseKeychain yes
    AddKeysToAgent yes
% ssh lab;   scp report.pdf lab:~/;   rsync -avz ~/site/ lab:~/site/

# Server: /etc/ssh/sshd_config.d/100-mwu.conf
PasswordAuthentication no
KbdInteractiveAuthentication no
PermitRootLogin no
% sudo launchctl kickstart -k system/com.openssh.sshd
```

Keep one SSH session open while you test a new server setting.

Networking on Other UNIX Systems

Task	FreeBSD	OpenBSD	Solaris 11.4
Set DHCP	sysrc ifconfig_em0=DHCP	echo 'inet autoconf' > /etc/hostname.em0	ipadm create-addr -T dhcp net0/v4
Apply	service netif restart	sh /etc/netstart em0	(immediate)
Show links	ifconfig	ifconfig	dladm show-phys; ipadm
Firewall	pf or ipfw (pf_enable=YES)	pf (on by default)	PF (svcadm enable firewall)
SSH server	sysrc sshd_enable=YES	rcctl enable sshd	svcadm enable ssh

Lab: Connect, Protect, Reach

```
% networksetup -listallhardwareports; ipconfig getifaddr en0
% scutil --dns | grep nameserver | sort -u;  dig +short apple.com
% sudo scutil --set LocalHostName mwu-lab
% sudo /usr/libexec/ApplicationFirewall/socketfilterfw --setglobalstate on
% sudo systemsetup -setremotelogin on
# From your host Mac (or another VM):
% ssh-keygen -t ed25519 && ssh-copy-id you@mwu-lab.local && ssh you@mwu-lab.local
% lsof -iTCP -sTCP:LISTEN -n -P | grep sshd
```

Summary & Homework

- Hardware ports (en0) vs network services; networksetup and scutil
- Troubleshoot bottom-up: link → router → IP → DNS → port → service
- Application Firewall (socketfilterfw) + pf anchors (pfctl)
- SSH: ed25519 keys, ssh-copy-id, ~/.ssh/config, no passwords, no root
- Homework: do the lab; block and unblock a port with pf
- Next: Lesson 11 — Storage, APFS, Security & Backup

Thank You! 🙏

Free videos & PDFs: www.MyWebUniversity.com

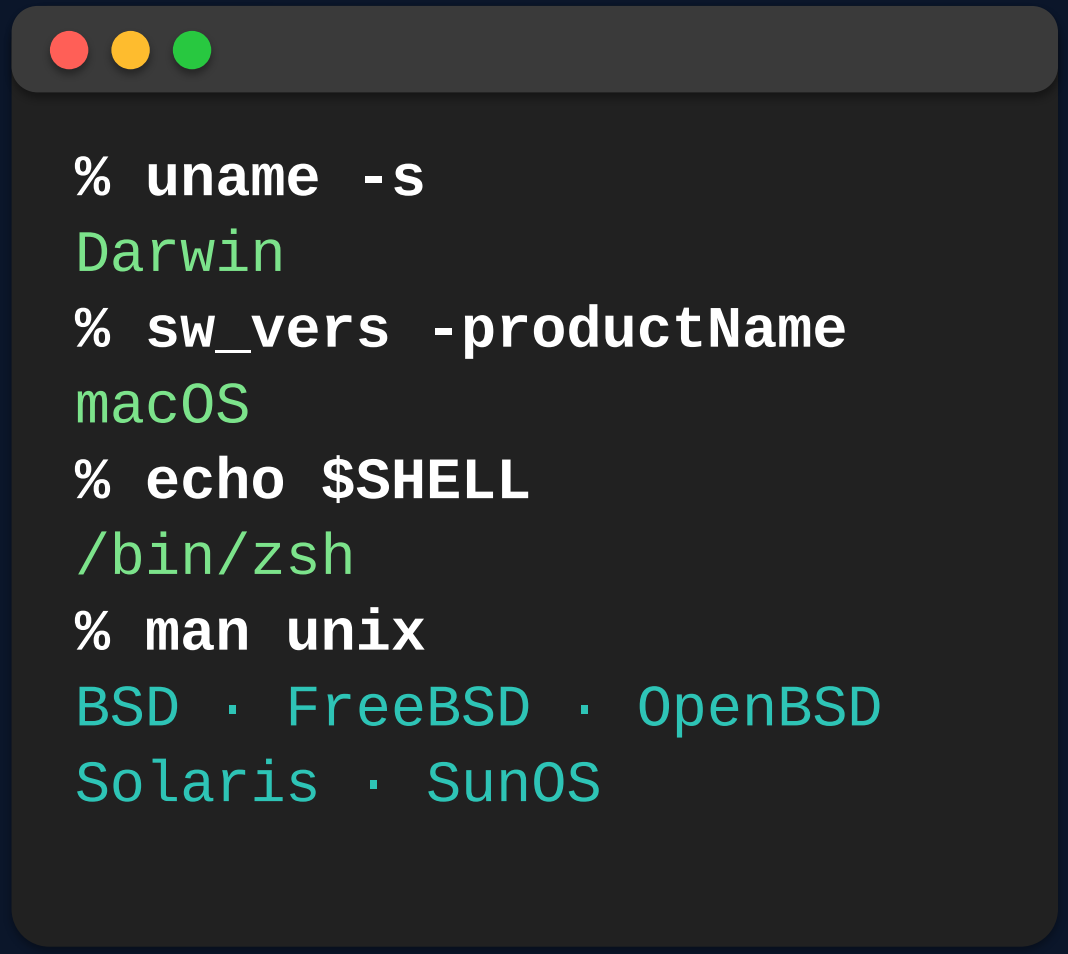
Slides for this lesson: MyWebUniversity.com/presentations

YouTube: @MyWebUniversity · @MyWebUniversity-Dari

Wahidullah Lutfy — Founder and Director of IT of MyWebUniversity, and CS teacher

Book: The Direct Path to Linux Ubuntu (Amazon · Barnes & Noble)

Learn it, then pass it on.



LESSON 11 OF 12

macOS UNIX

Darwin & BSD

Storage, APFS, Security & Backup

Full Course · macOS 27 Golden Gate

What You Will Learn in Lesson 11

- APFS: containers, volumes, snapshots and clones
- diskutil: list, inspect, add volumes, erase, verify
- Disk images with hdiutil, including encrypted ones
- Security layers: FileVault, SIP, Gatekeeper, TCC, XProtect
- ACLs, extended attributes, quarantine and file flags
- Time Machine and tmutil; the 3-2-1 backup rule
- ZFS and encryption on FreeBSD, OpenBSD and Solaris

APFS in One Slide

Concept	What it means
Container	A pool of space on one disk partition; volumes inside share it
Volume	A file system: Macintosh HD (System), Data, Preboot, Recovery, VM
Volume group	System + Data volumes that appear as one tree (firmlinks)
Snapshot	A read-only, point-in-time copy; Time Machine and updates use them
Clone	cp -c copies a file instantly; space is used only for changes
Encryption	Built in; FileVault encrypts the Data volume

diskutil: See and Manage Disks

```
% diskutil list                          # every disk, container, volume
% diskutil info / | grep -E 'Volume Name|File System|FileVault|Free'
% diskutil apfs list | less              # containers in detail

% sudo diskutil apfs addVolume disk3 APFS Projects   # shares the space
% diskutil apfs deleteVolume /Volumes/Projects
% diskutil verifyVolume /                # check (First Aid, read-only)
% diskutil apfs listSnapshots /
% df -h /;  du -sh ~/* | sort -h | tail # space used, biggest folders
```

Container names (disk3) vary: read them from diskutil list first.

External Disks: Format, Mount, Eject

```
% diskutil list external                          # find the USB disk, e.g. disk4
% diskutil eraseDisk APFS MyUSB GPT disk4         # Mac only (ERASES disk4!)
% diskutil eraseDisk ExFAT SHARE GPT disk4        # Mac + Windows + Linux

% diskutil unmount /Volumes/MyUSB;  diskutil mount MyUSB
% diskutil eject disk4

# Formats macOS reads/writes: APFS, Mac OS Extended (HFS+), ExFAT, MS-DOS (FAT32)
# NTFS: read-only by default
```

Triple-check the disk identifier before any erase command.

Disk Images: hdiutil

```
% hdiutil create -size 2g -fs APFS -volname Vault \
      -encryption AES-256 -stdinpass ~/vault.dmg      # type a password
% hdiutil attach ~/vault.dmg                          # asks for it
% cp ~/lab/*.sql /Volumes/Vault/
% hdiutil detach /Volumes/Vault

% hdiutil create -srcfolder ~/lab -format UDZO ~/lab.dmg   # compressed copy
% hdiutil verify ~/lab.dmg
% cp -c bigfile.iso clone.iso      # APFS clone: instant, no extra space
```

The Security Layers of macOS

Layer	Protects	Check it
FileVault	Data at rest if the Mac is stolen	fdesetup status
System Integrity Protection	System files, even from root	csrutil status
Signed System Volume	The OS itself, by a cryptographic seal	diskutil info / | grep -i sealed
Gatekeeper + notarization	Against untrusted apps	spctl --status
TCC privacy controls	Camera, mic, files, Full Disk Access	System Settings → Privacy & Security
XProtect	Known malware, updated by Apple	automatic

FileVault, SIP and Gatekeeper From Terminal

```
% fdesetup status
FileVault is On.
% sudo fdesetup enable                    # prints a recovery key: store it safely!
% sudo fdesetup list                      # users who can unlock the disk

% csrutil status
System Integrity Protection status: enabled.

% spctl --status                          # assessments enabled
% spctl -a -vv /Applications/Safari.app   # accepted? who signed it?
% tccutil reset Camera                    # reset camera permissions
```

ACLs, Extended Attributes and Flags

```
% chmod +a "labstudents allow read,write" ~/Shared-Lab      # add an ACL entry
% ls -le ~/Shared-Lab                      # show ACLs (the + sign)
% chmod -a# 0 ~/Shared-Lab                 # remove entry 0;  chmod -N removes all

% xattr -l ~/Downloads/tool.zip            # extended attributes
com.apple.quarantine: 0083;...;Safari;...
% xattr -d com.apple.quarantine ~/Downloads/tool.zip   # only if you trust it!

% chflags uchg important.txt;  ls -lO important.txt     # immutable flag
% chflags nouchg important.txt
```

Time Machine and tmutil

```
# GUI: System Settings → General → Time Machine → Add Backup Disk
% sudo tmutil setdestination -a /Volumes/TM-Backup
% tmutil destinationinfo
% tmutil startbackup --block            # back up now and wait
% tmutil latestbackup;  tmutil listbackups | tail -3
% sudo tmutil addexclusion -p ~/Downloads   # skip a folder
% tmutil localsnapshot;  tmutil listlocalsnapshots /

# Restore one file: open its folder → Enter Time Machine, or:
% tmutil restore <backup-path>/Users/wahid/lab/school.db ~/lab/
```

The 3-2-1 Backup Rule

- 3 copies of your data: the original + 2 backups
- 2 different kinds of storage (e.g. external SSD + network/cloud)
- 1 copy off-site (another building or the cloud)
- Automate it: Time Machine, launchd + rsync, database dumps
- Encrypt backups that leave your control
- Test a restore every month — an untested backup is only a hope

Storage and Security on Other UNIX Systems

	FreeBSD	OpenBSD	Solaris 11.4
Main file systems	ZFS, UFS	FFS	ZFS
Snapshots	zfs snapshot pool/home@now	(none in FFS)	zfs snapshot rpool/export/home@now
Disk encryption	GELI, ZFS native encryption	softraid CRYPTO (bioctl)	ZFS encryption
Send backups	zfs send | ssh host zfs recv	dump / restore, rsync	zfs send | zfs recv
Extra hardening	securelevel, jails, Capsicum	pledge, unveil, W^X	RBAC, zones, immutable zones

Lab: Protect the Data

```
% diskutil list;  diskutil info / | grep -iE 'sealed|FileVault'
% sudo diskutil apfs addVolume disk3 APFS Projects;  df -h /Volumes/Projects
% hdiutil create -size 200m -fs APFS -volname Vault -encryption AES-256 -stdinpass ~/vault.dmg
% hdiutil attach ~/vault.dmg && cp ~/lab/school.db /Volumes/Vault/ && hdiutil detach /Volumes/Vault
% fdesetup status;  csrutil status;  spctl --status
% mkdir ~/Shared-Lab && chmod +a "staff allow read" ~/Shared-Lab && ls -le ~ | grep Shared
% tmutil localsnapshot && tmutil listlocalsnapshots /
% diskutil apfs deleteVolume /Volumes/Projects
```

Summary & Homework

- APFS: containers share space; snapshots, clones, built-in encryption
- diskutil list / info / addVolume / eraseDisk; hdiutil for disk images
- FileVault, SIP, sealed system, Gatekeeper, TCC, XProtect
- ACLs (chmod +a), xattr quarantine, chflags uchg
- Time Machine + tmutil; follow 3-2-1 and test restores
- Next: Lesson 12 — Administering macOS: Desktop vs Server

Thank You! 🙏

Free videos & PDFs: www.MyWebUniversity.com

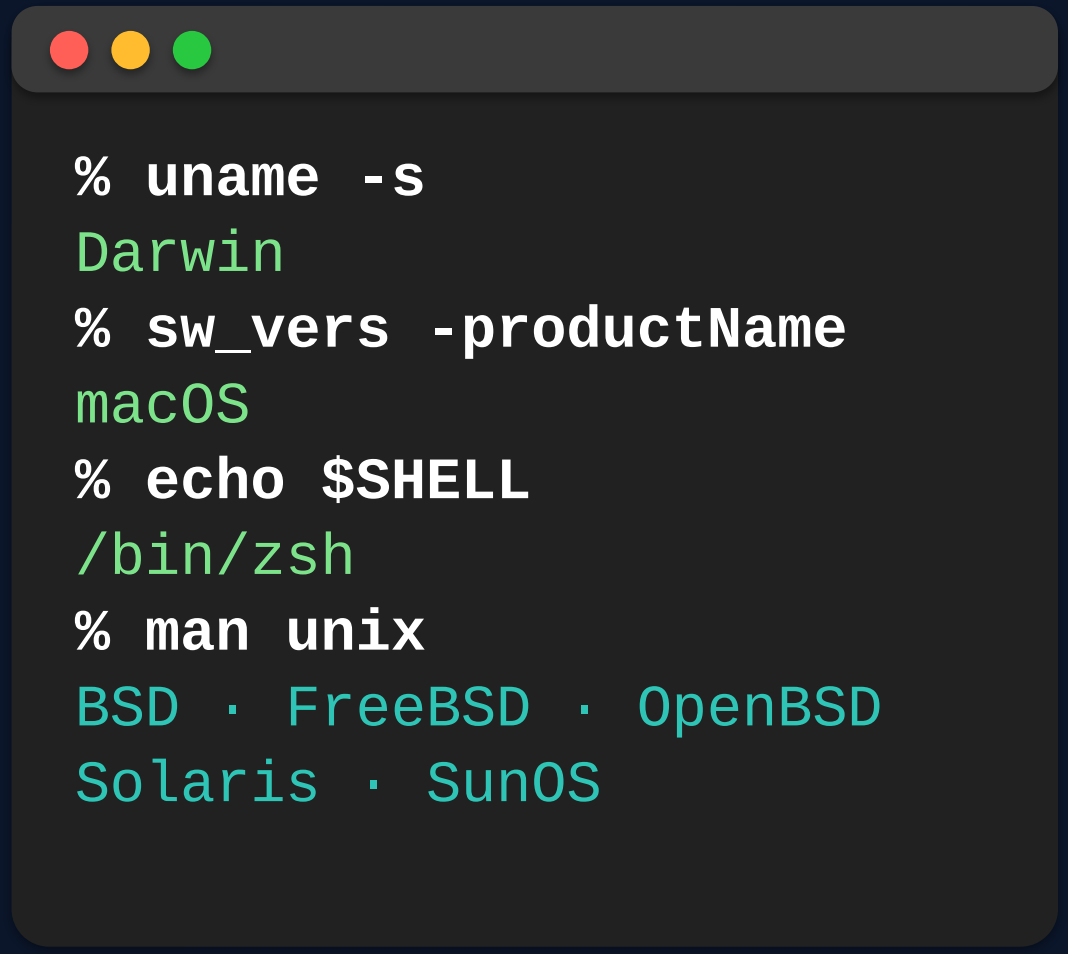
Slides for this lesson: MyWebUniversity.com/presentations

YouTube: @MyWebUniversity · @MyWebUniversity-Dari

Wahidullah Lutfy — Founder and Director of IT of MyWebUniversity, and CS teacher

Book: The Direct Path to Linux Ubuntu (Amazon · Barnes & Noble)

Learn it, then pass it on.



LESSON 12 OF 12

macOS UNIX

Darwin & BSD

Administering macOS: Desktop vs Server Roles

Full Course · macOS 27 Golden Gate

What You Will Learn in Lesson 12

- Desktop vs server: what changes for an administrator
- From macOS Server to services built into macOS
- Always-on settings: power, restart, FileVault authrestart
- File Sharing, Screen Sharing, Content Caching, Time Machine server
- Hosting a web service with launchd
- Managing many Macs: MDM and configuration profiles
- Monitoring, troubleshooting, and choosing BSD or Solaris for a server

Desktop vs Server Administration

	Desktop Mac	Server Mac
Users	One or a few people logged in	Often nobody logged in; services run as daemons
Uptime	Sleeps, restarts for updates any time	Never sleeps; planned maintenance windows
Services	LaunchAgents, apps	LaunchDaemons: file, web, database, caching
Access	Keyboard and screen	SSH, Screen Sharing, MDM
Network	DHCP, Wi-Fi is fine	Wired Ethernet, static IP or DHCP reservation
Priorities	Privacy, usability	Availability, monitoring, backups, change control

macOS Server Is Retired: What Replaced It

Old macOS Server role	Today on macOS
File server	General → Sharing → File Sharing (SMB)
Time Machine server	File Sharing → share options → 'Share as a Time Machine backup destination'
Caching server	General → Sharing → Content Caching
Remote management	Screen Sharing / Remote Management (Apple Remote Desktop)
Profile Manager (MDM)	Third-party or open source MDM + Apple Business / School Manager
Web, mail, DNS, wiki	Homebrew services (nginx, Apache, Postfix…) or Linux/BSD servers

Always-On Settings

```
% pmset -g                                  # current power settings
% sudo pmset -a sleep 0 disksleep 0 displaysleep 10   # never sleep
% sudo pmset -a womp 1                      # wake for network access
% sudo pmset -a autorestart 1               # power back on after an outage
% sudo systemsetup -setrestartfreeze on     # restart if the system freezes

% sudo fdesetup authrestart                 # reboot once without the FileVault prompt
% sudo shutdown -r +5 "Maintenance reboot"  # warn users, restart in 5 min
```

File Sharing (SMB)

```
# GUI: General → Sharing → File Sharing → ⓘ → add folders, set users & rights
% sudo mkdir -p /Users/Shared/LabShare
% sudo chown :labstudents /Users/Shared/LabShare && sudo chmod 2770 /Users/Shared/LabShare
% sudo sharing -a /Users/Shared/LabShare -S LabShare   # add an SMB share point
% sharing -l                                          # list share points

# From another Mac:   Finder → Go → Connect to Server →  smb://mwu-lab.local
% open smb://mwu-lab.local/LabShare
% smbutil view //student1@mwu-lab.local               # shares on a server
```

Screen Sharing, Content Caching, Time Machine Server

```
# Screen Sharing (VNC-compatible):  General → Sharing → Screen Sharing
% open vnc://mwu-lab.local                          # from another Mac

# Content Caching: saves internet bandwidth for every Apple device on the LAN
% AssetCacheManagerUtil status
% sudo AssetCacheManagerUtil activate

# Time Machine destination for other Macs:
#   File Sharing → share folder → ⓘ → 'Share as a Time Machine backup destination'
% tmutil destinationinfo                            # on the client Mac
```

Hosting a Web Service With launchd

```
% brew install nginx
% sudo brew services start nginx      # system daemon: starts at boot
% curl -I http://localhost:8080        # Homebrew nginx listens on 8080
% echo '<h1>MWU Lab Server</h1>' > /opt/homebrew/var/www/index.html

% sudo launchctl list | grep nginx
% ls /Library/LaunchDaemons | grep nginx   # the daemon plist brew created
% sudo /usr/libexec/ApplicationFirewall/socketfilterfw --add /opt/homebrew/bin/nginx
```

Managing Many Macs: MDM

- Apple Business Manager / Apple School Manager: own and assign devices
- Automated Device Enrollment: new Macs enroll at first boot
- MDM server pushes configuration profiles, apps and updates
- Profiles: Wi-Fi, VPN, FileVault, passcode, restrictions (.mobileconfig)
- Products: Jamf, Kandji, Mosyle, Intune; open source: NanoMDM, MicroMDM
- Check a Mac: profiles status -type enrollment ; sudo profiles show

Monitoring and Maintenance Routine

- Daily: backups succeeded? disk space? (df -h, tmutil latestbackup)
- Daily: failed services and errors (launchctl list, log show --last 1d)
- Weekly: softwareupdate --list; brew outdated; review admin group
- Weekly: listening ports (lsof -iTCP -sTCP:LISTEN) — anything new?
- Monthly: test a restore; review users, sudoers.d, firewall rules
- Automate checks with launchd scripts that email or notify you

Troubleshooting Checklist

Symptom	First commands	Lesson
Won't boot / update failed	Recovery → Disk Utility First Aid, reinstall; tmutil	2, 11
App crashes or hangs	Console crash reports, sample, spindump, Safe Mode	3, 9
Mac is slow	top -o cpu, Memory Pressure, fs_usage, df -h	3, 9, 11
Can't log in / no admin	dscl . -read /Groups/admin, Secure Token, Recovery	8
Service not running	launchctl print, log show --predicate, plutil -lint	9
Network problem	ifconfig → ping router → dig → nc -vz → curl	10
Disk full / can't eject	du -sh | sort -h, tmutil listlocalsnapshots, lsof	11

Choosing a Server OS

Choose	When you need	Strengths
macOS	Xcode build machines, small offices, caching, Apple-centric labs	Easy to manage, MDM, great hardware
FreeBSD	Storage, network appliances, high-traffic web/CDN	ZFS, jails, performance, permissive license
OpenBSD	Firewalls, routers, VPN gateways, security-critical hosts	pf, secure defaults, simplicity, code audits
Solaris 11.4	Existing Oracle/SPARC enterprise systems	ZFS, zones, SMF, DTrace, long support
Linux	Most cloud and container workloads	Huge ecosystem (see our RHEL, Rocky, Ubuntu courses)

Capstone Lab: Build a Lab Server

```
% sudo scutil --set LocalHostName mwu-server              # Lesson 10
% sudo pmset -a sleep 0 autorestart 1 womp 1              # Lesson 12
% sudo dseditgroup -o create labstudents; sudo sysadminctl -addUser student1 -password -   # 8
% sudo mkdir -p /Users/Shared/LabShare && sudo sharing -a /Users/Shared/LabShare -S LabShare
% sudo systemsetup -setremotelogin on                     # SSH, keys only (10)
% brew install nginx postgresql@17 && sudo brew services start nginx   # 5, 7, 9
% sudo /usr/libexec/ApplicationFirewall/socketfilterfw --setglobalstate on   # 10
% tmutil localsnapshot; pg_dump ... ; launchd backup job   # 7, 9, 11
% lsof -iTCP -sTCP:LISTEN -n -P        # document every open port
```

Write a one-page 'server document': name, IP, services, ports, users, backups.

Course Review: 12 Lessons

#	You can now	#	You can now
1	Explain UNIX, BSD, Solaris, Darwin, XNU	7	Run SQLite, PostgreSQL, MySQL; back up
2	Install, upgrade, recover macOS	8	Manage users, groups, sudo, Secure Tokens
3	Master the GUI, apps and utilities	9	Control processes; write launchd services
4	Work fluently in zsh and the file system	10	Configure networks, pf and SSH
5	Install software with brew, ports, pkg	11	Manage APFS, security layers, backups
6	Compile, script and debug	12	Run a Mac as a server; troubleshoot

Your Next Steps

- Rebuild the capstone server from memory, then script it
- Install FreeBSD or OpenBSD in a VM and repeat Lessons 4, 9 and 10
- Continue with our free RHEL, Rocky and Ubuntu Linux courses
- Read: The Direct Path to Linux Ubuntu (free chapters online)
- Certifications to consider: Apple Certified Support Professional, Linux+
- Teach someone else what you learned — that is how knowledge grows

Thank You! 🙏

Free videos & PDFs: www.MyWebUniversity.com

Slides for this lesson: MyWebUniversity.com/presentations

YouTube: @MyWebUniversity · @MyWebUniversity-Dari

Wahidullah Lutfy — Founder and Director of IT of MyWebUniversity, and CS teacher

Book: The Direct Path to Linux Ubuntu (Amazon · Barnes & Noble)

Learn it, then pass it on.