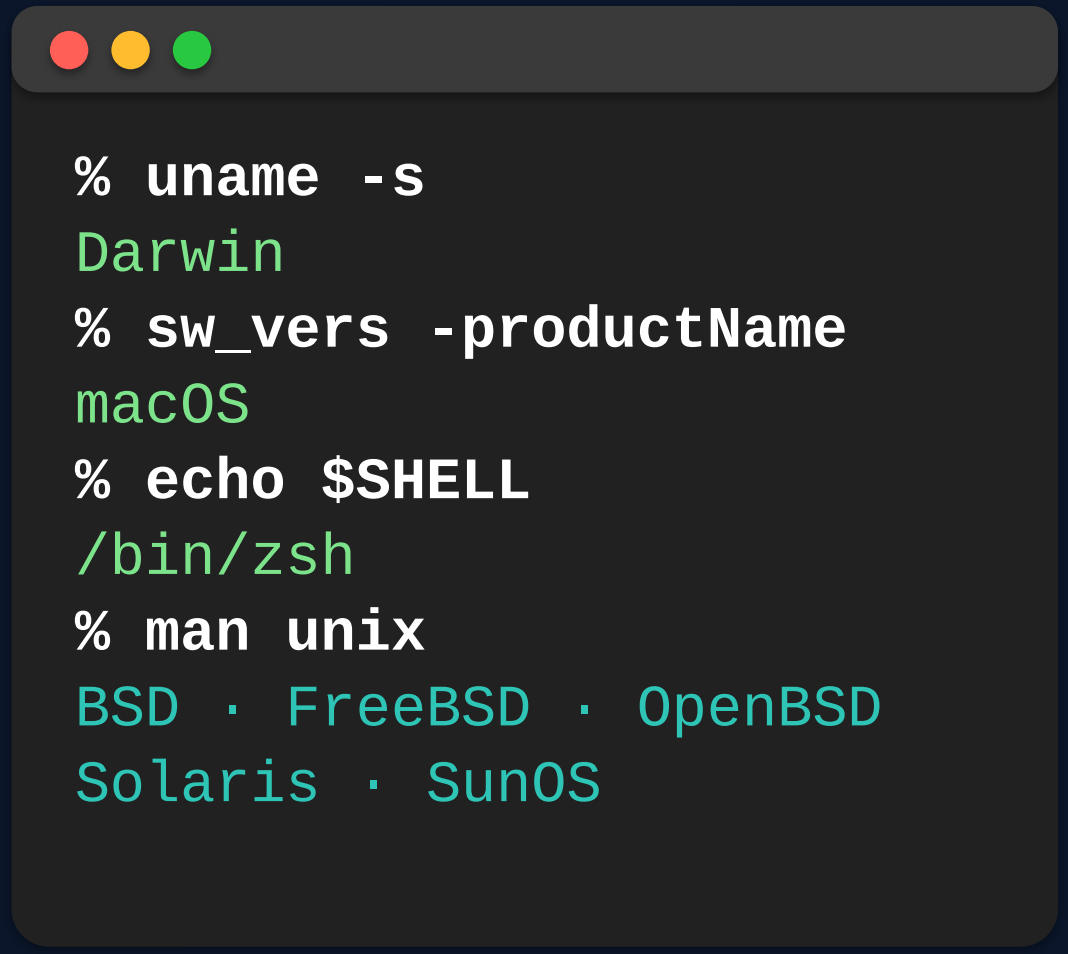


LESSON 6 OF 12

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.