Modules → Functionality created as module

→ Add functionality as module

→ OOP approach

→ Drivers are examples of modules that are loaded to kernal

Solaris Modular Approach → Everything is a module

Hybrid Approach → Monolithic + Modular for dynamic needs

→ MacOS + iOS → Darwin
↳ Hybrid approach

→ Windows → Mostly Monolithic

System calls → Programming interface to the services provided

→ usually by any API

Read a file → Read system call
↳ In Kernal mode
fscanf
↳ In User mode

POSIX API → Linux API

System Call Implementation
① A number for each system call
② Call is indexed with the table
③ Involves the call and returns status of call &/or return values

Caller POV → Abstract, no idea about the call procedure

Types of System calls → File Management
Device Mgmt
Protection

Info Mgmt
Communications

Wrapping of System calls → System calls are wrapped inside the library functions that are used by Developers

Architecture of UNIX System → One type of layered architecture

Linux Statement — Everything is a file

→ Process control sub system → Software that is a file but run as a process

→ Contains info of the process

echo $SHELL → which shell
↳ Bash / zsh

ps → which processes
↳ Not all

ps aux → All processes from all the users

clear → clear the terminal

pwd → Print working directory

Arrow signs → System folders
↳ bin, lib, lib32

man ls → Manual of any command name
↳ Better than --help flag

ls -l → List in long form

echo → Print something on the screen

cd $DIR → Change directory
↳ cd .. → Back to parent

~	→	Home directory
cd	→	Back to home
cd .. /..	→	Parent of parent
touch	→	Access the file ↳ Create if absent ↳ Else → only access Literally, touch
Cat file	→	concatination
cat > file	→	write in file using cat
cat *.txt	→	Order is last accessed first

Summary: Modules - Monolithic to Modular and hybrid approach - System calls using API - Types of System Calls - Architecture

of UNIX Systems - Important Linux commands - PS - ps aux - cd../..- cat 1.txt> 2.txt - cat *.txt