OS Structure

1DV512 – Operating Systems

Dr. Kostiantyn Kucher

kostiantyn.kucher@lnu.se

November 4, 2020

Based on the Operating System Concepts slides by Silberschatz, Galvin, and Gagne (2018)

Suggested OSC book complement: Chapters 1 & 2

Agenda

- Motivation and Introduction to Operating Systems
- Operating System Structure
- Operating System Architectures and Examples
- Summary

Typical Computer System Organization (for Personal Computers)

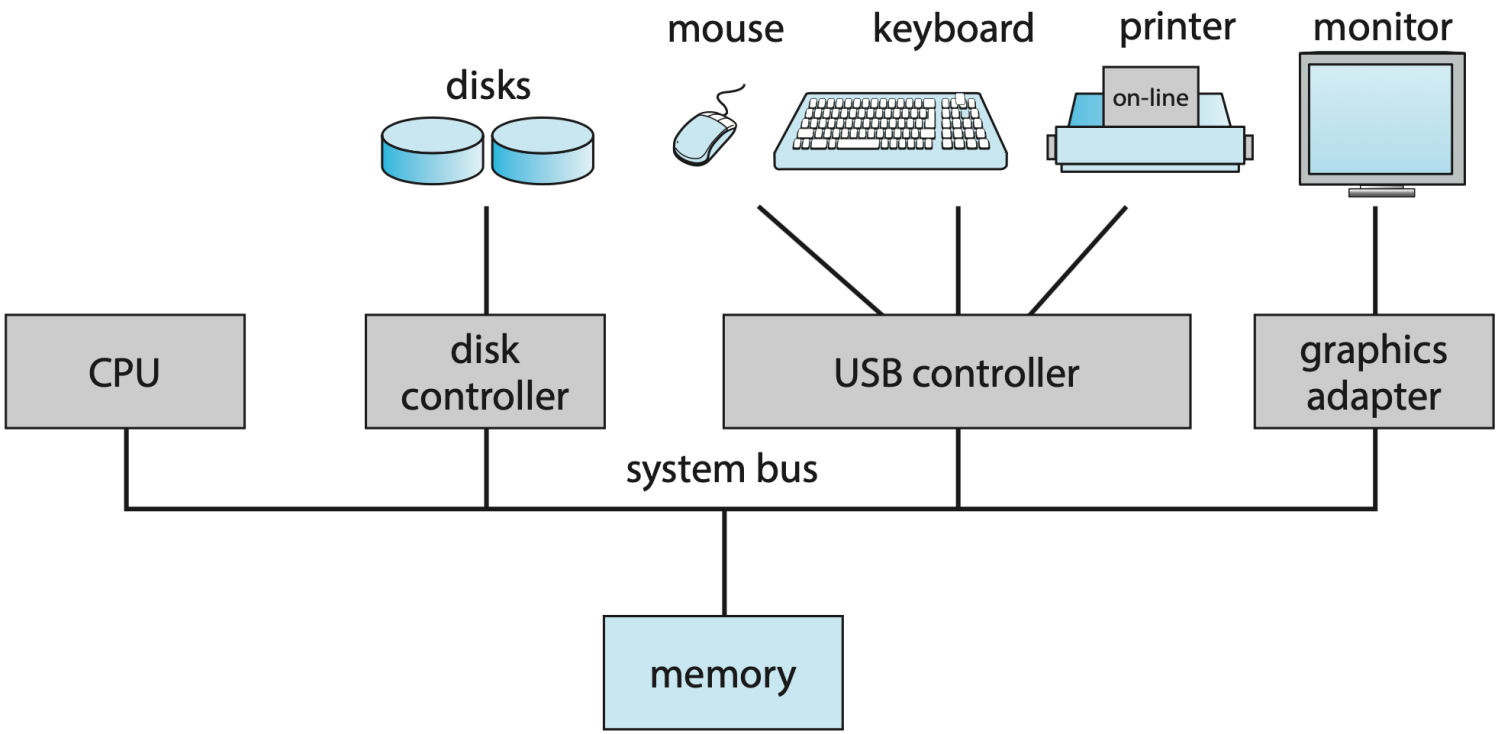


Fig. 1.2 in OSC book

Hardware

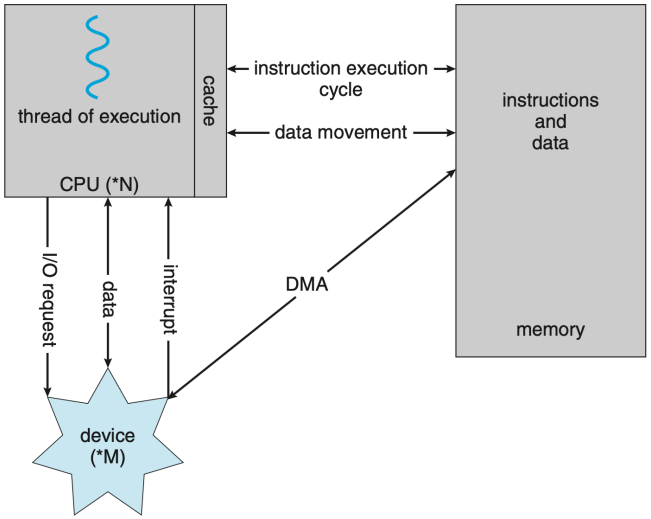


Fig. 1.7 in OSC book

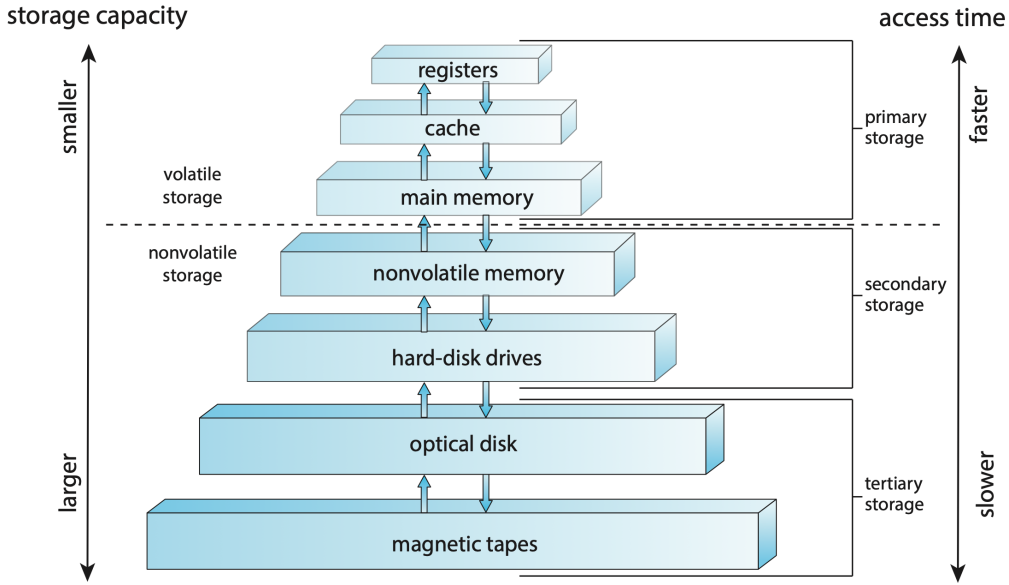


Fig. 1.6 in OSC book

Abstract View of Computer Systems

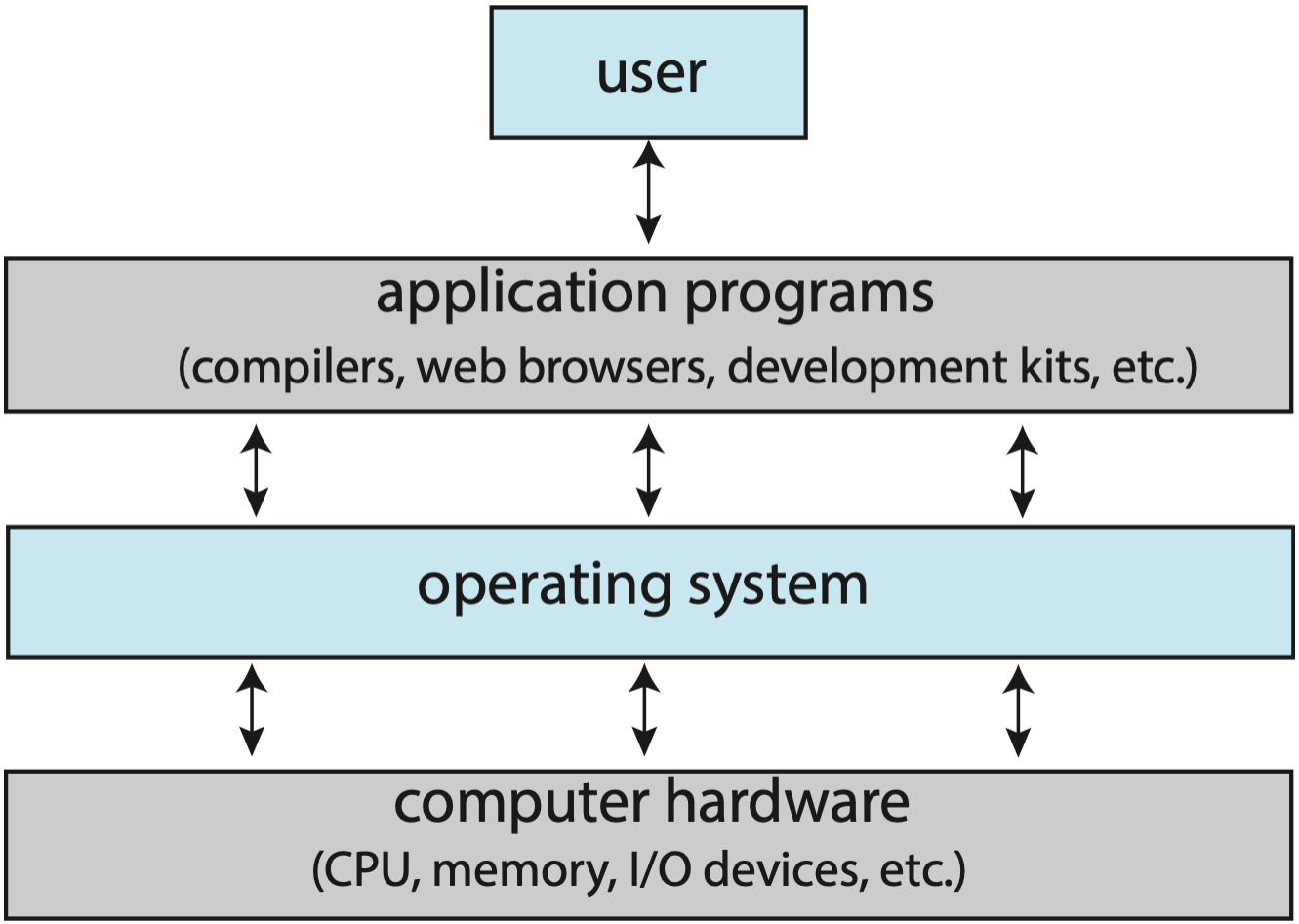
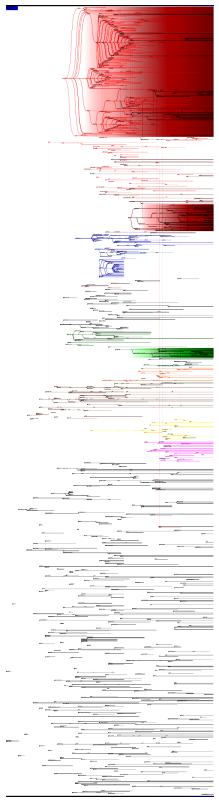


Fig. 1.1 in OSC book

What is an Operating System?

- OS ⇒ A program that acts as an intermediary between a user of a computer and the computer hardware
- OS goals:
  - Execute user programs and make solving user problems easier
  - Make the computer system convenient to use
  - Use the computer hardware in an efficient manner
- The main components of OS:
  - The one program running at all times on the computer ⇒ the *kernel*
  - System programs
  - All the rest ⇒ application programs
- Historically, at least between 800 and 900 different operating systems were created for various purposes since ∼1960s



http://maps-and-tables.blogspot.com/2019/01/almost-complete-timeline-and-family.html

Agenda

- Motivation and Introduction to Operating Systems
- Operating System Structure
- Operating System Architectures and Examples
- Summary

Operating System Services

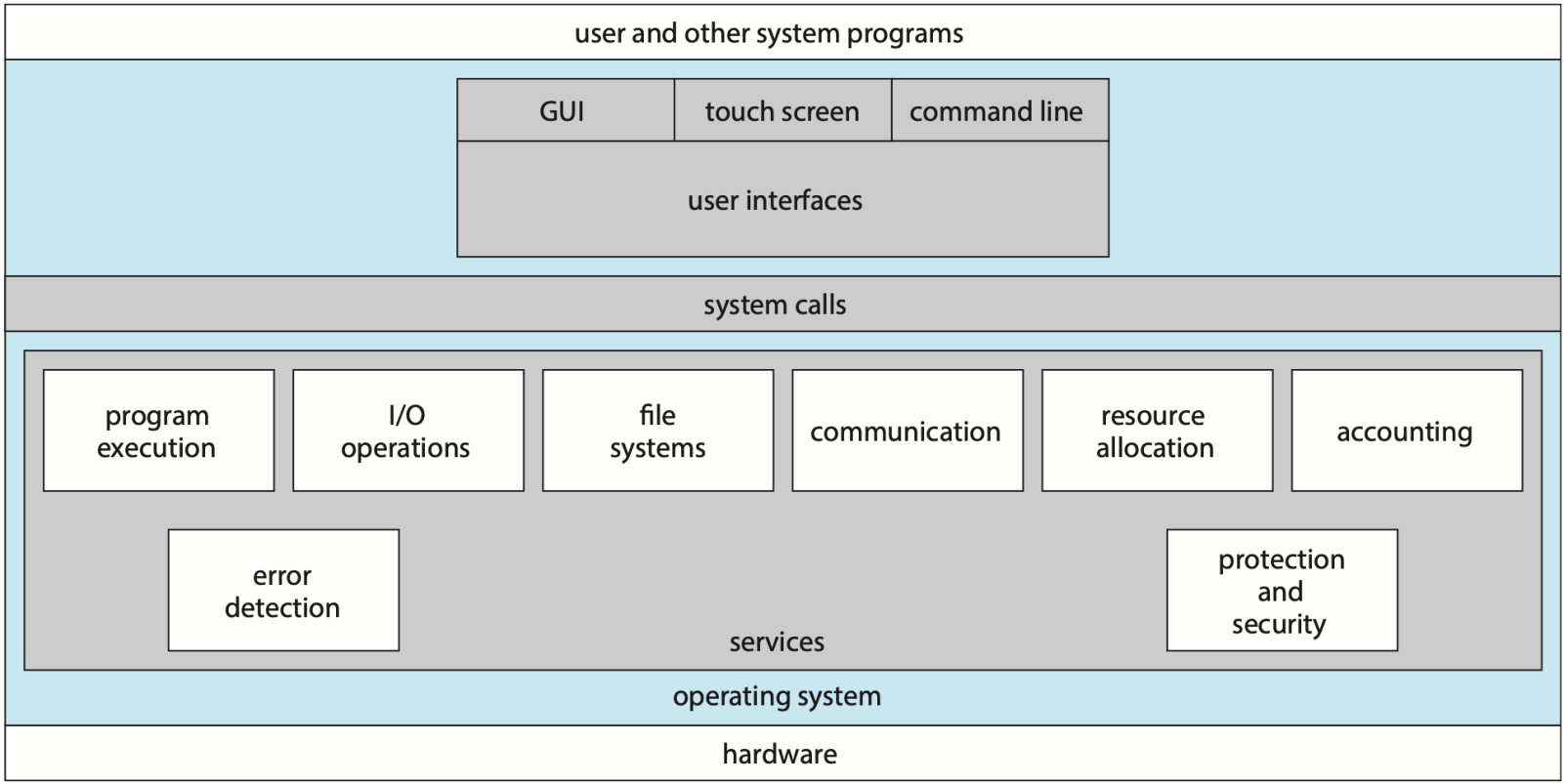


Fig. 2.1 in OSC book

User and Kernel Mode

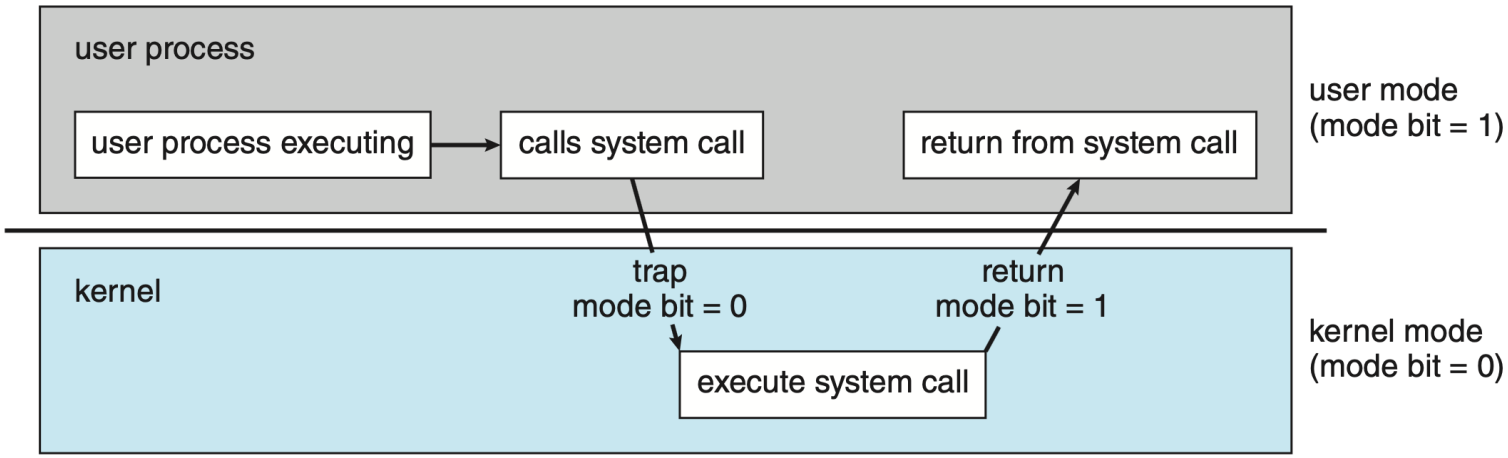


Fig. 1.13 in OSC book

- Dual-mode operation allows OS to protect itself and other system components ⇒ involves support from hardware ("mode bit")
- At system boot time, the hardware starts in *kernel mode* ⇒ the operating system is then loaded and starts user applications in user mode
- When a user is running ⇒ mode bit is "user", and similarly for kernel mode
- *System call* changes mode to kernel, return from call resets it to user
- Some instructions designated as *privileged*, only executable in kernel mode ⇒ I/O control, timer management, and interrupt management

System Calls

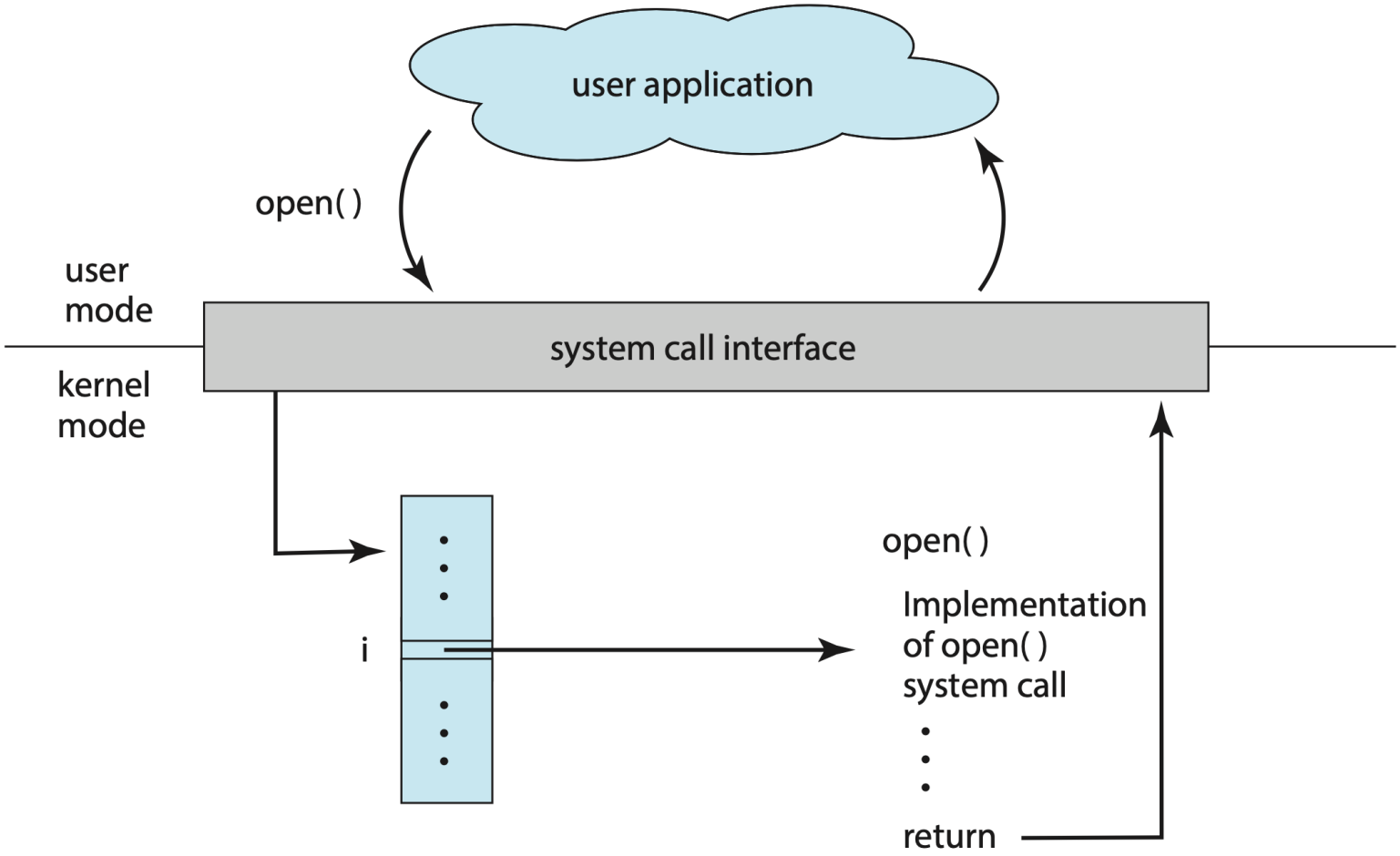


Fig. 2.6 in OSC book

System Calls (cont.)

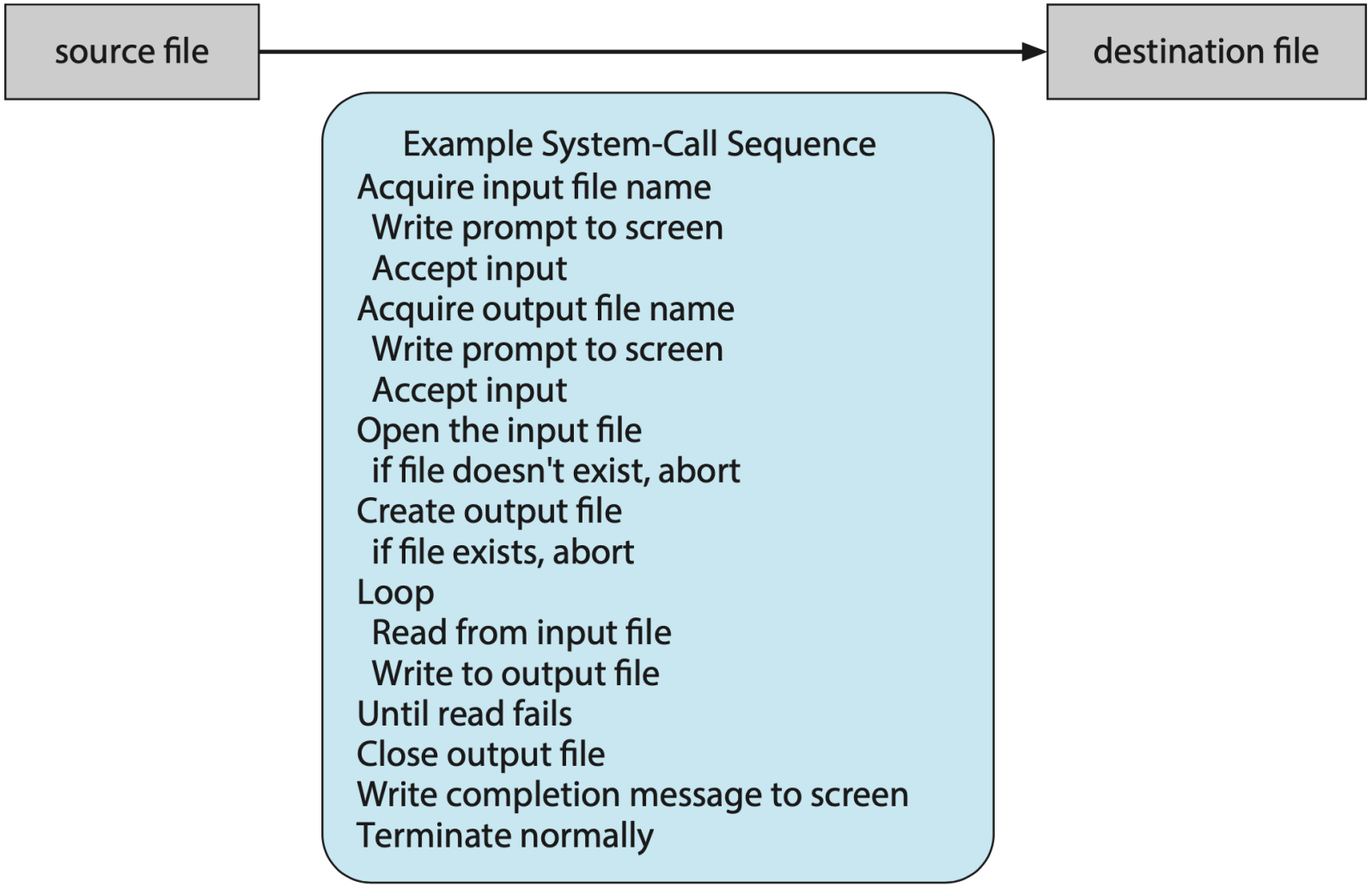


Fig. 2.5 in OSC book

System Calls (concl.)

	Windows	Unix
Process control	CreateProcess() ExitProcess() WaitForSingleObject()	fork() exit() wait()
File management	CreateFile() ReadFile() WriteFile() CloseHandle()	open() read() write() close()
Device management	SetConsoleMode() ReadConsole() WriteConsole()	ioctl() read() write()
Information maintenance	GetCurrentProcessID() SetTimer() Sleep()	getpid() alarm() sleep()
Communications	CreatePipe() CreateFileMapping() MapViewOfFile()	pipe() shm_open() mmap()
Protection	SetFileSecurity() InitlializeSecurityDescriptor() SetSecurityDescriptorGroup()	chmod() umask() chown()

Ch. 2 in OSC book

Agenda

- Motivation and Introduction to Operating Systems
- Operating System Structure
- Operating System Architectures and Examples
- Summary

OS Architecture Approaches

- *Traditional* approach
  - A single, static binary file that runs in a single address space ⇒ *monolithic* kernel
  - Simple and efficient
  - Difficult to extend and scale up to more complex tasks
- *Layered* approach
  - Bottom layer ⇒ hardware interface
  - Highest layer ⇒ user interface
  - Often used as inspiration rather than implemented directly
- *Modular* approach
  - OS services provided through modules that can be loaded and removed during run time
  - Minimal kernel ⇒ *microkernel* approach
  - Most OS services run as user-level applications
- In practice: combination of this approaches
  - E.g., Linux kernel is monolithic (rather than microkernel), but it can be extended through modules

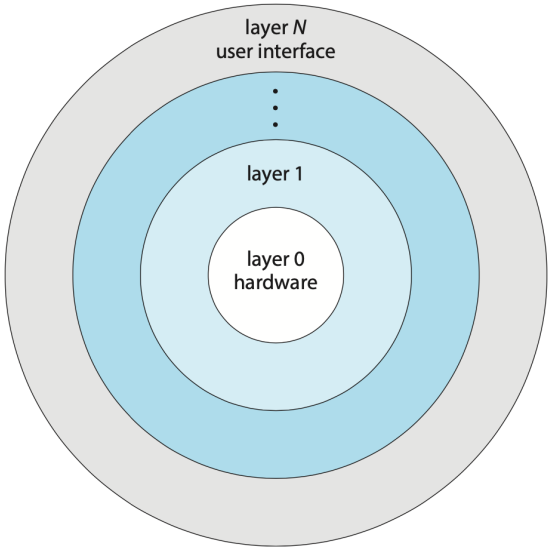


Fig. 2.14 in OSC book

Unix Architecture

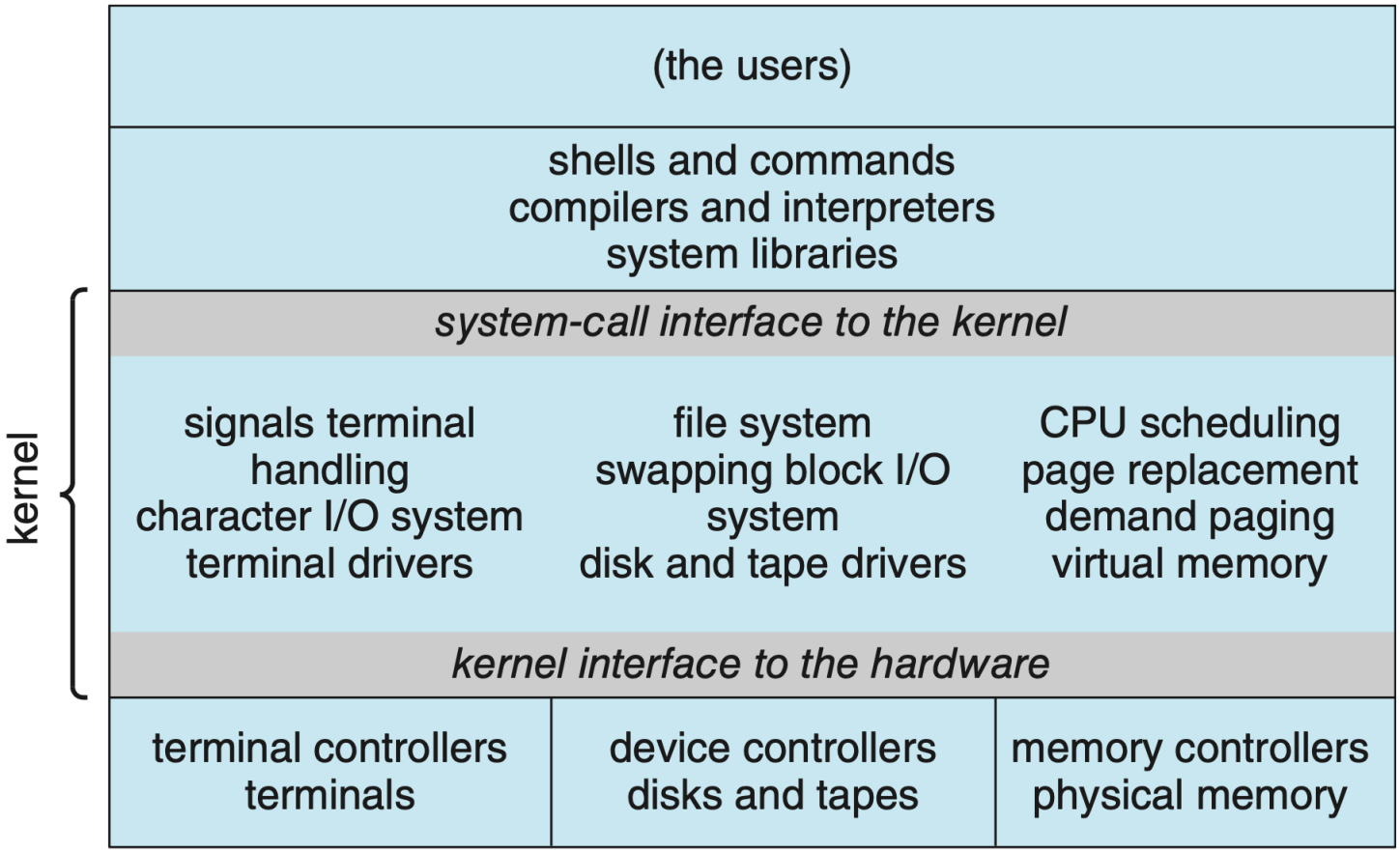
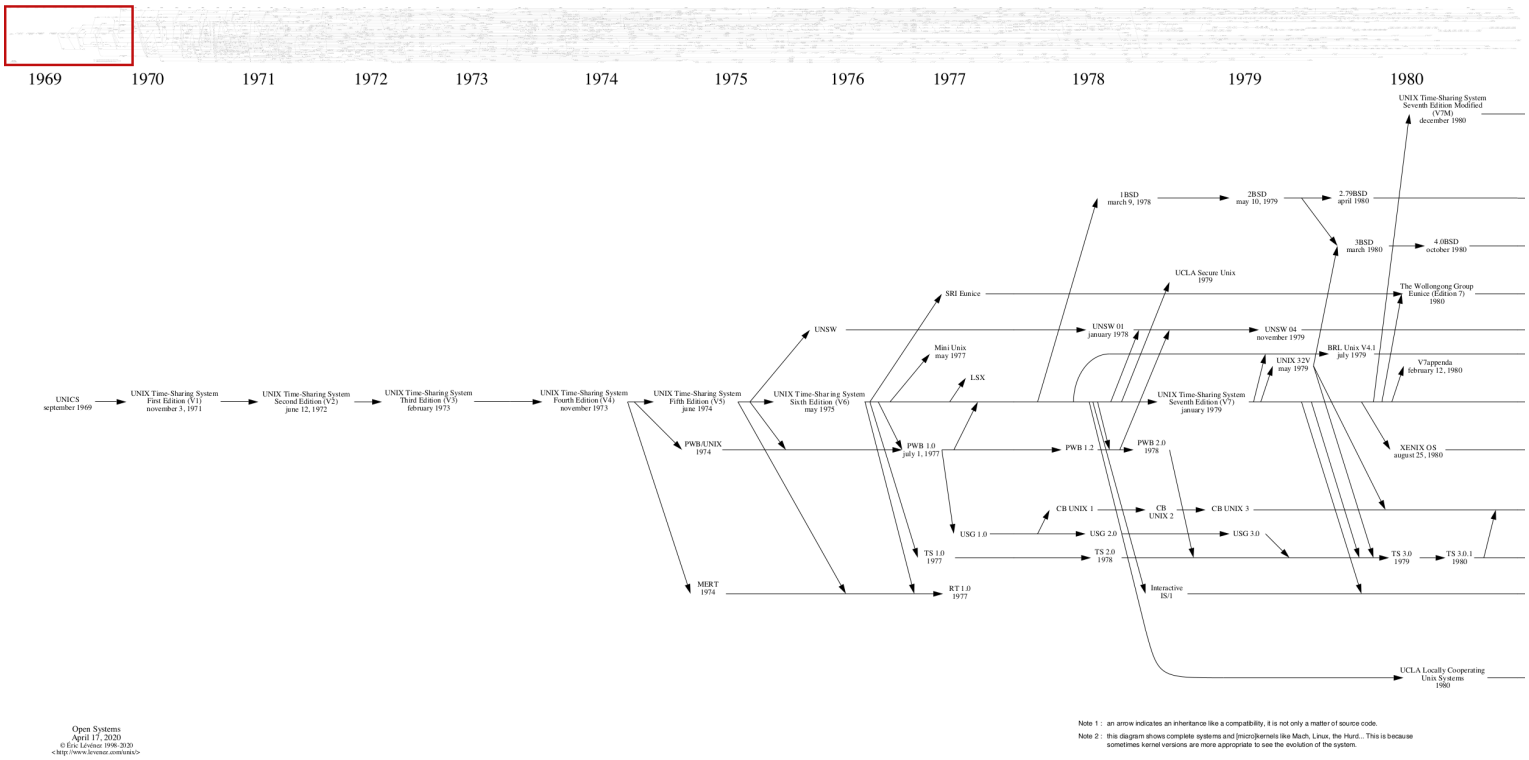


Fig. 2.12 in OSC book

Unix-like Systems History Timeline



https://www.levenez.com/unix/

Linux Architecture

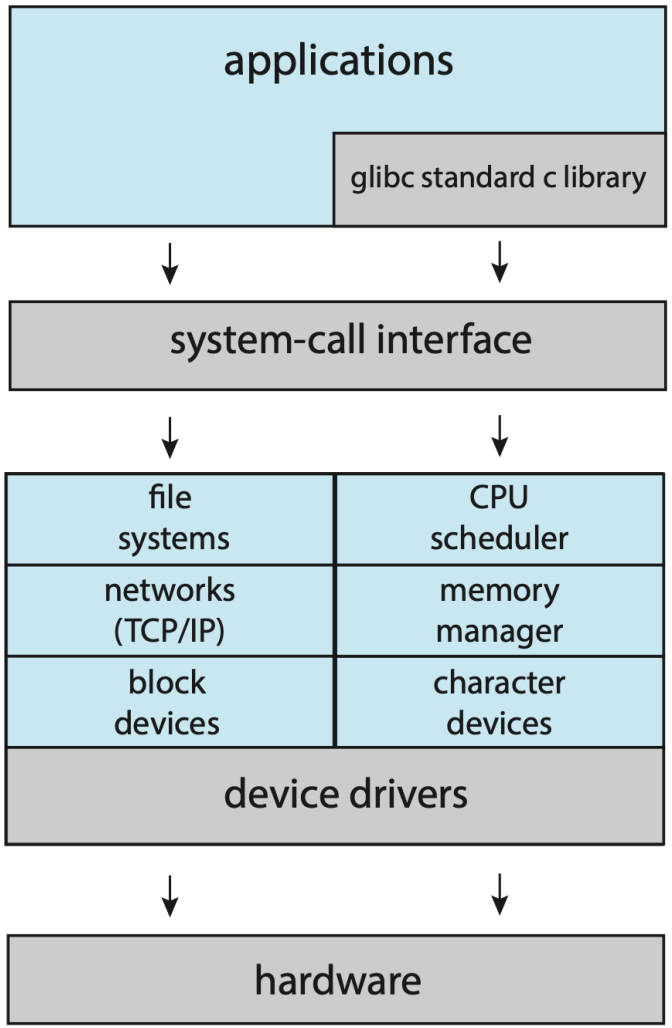


Fig. 2.13 in OSC book

Android Architecture

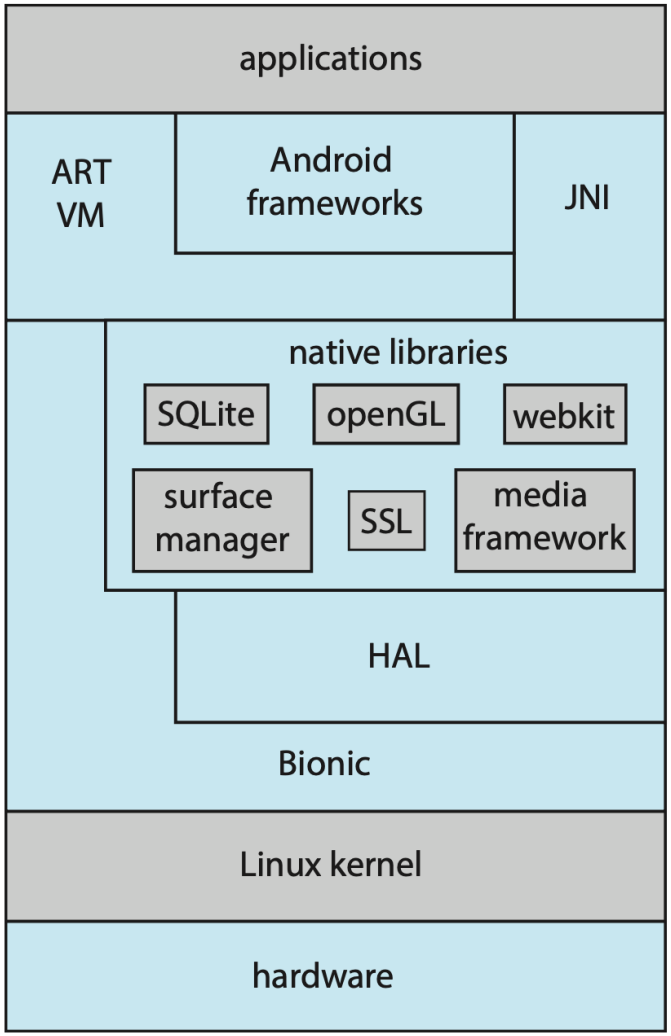


Fig. 2.18 in OSC book

macOS Architecture

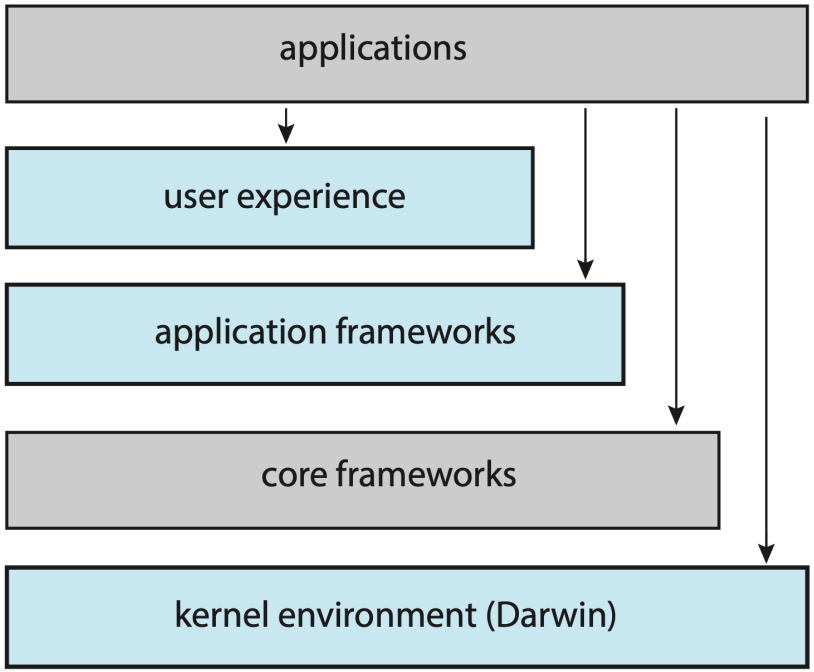


Fig. 2.16 in OSC book

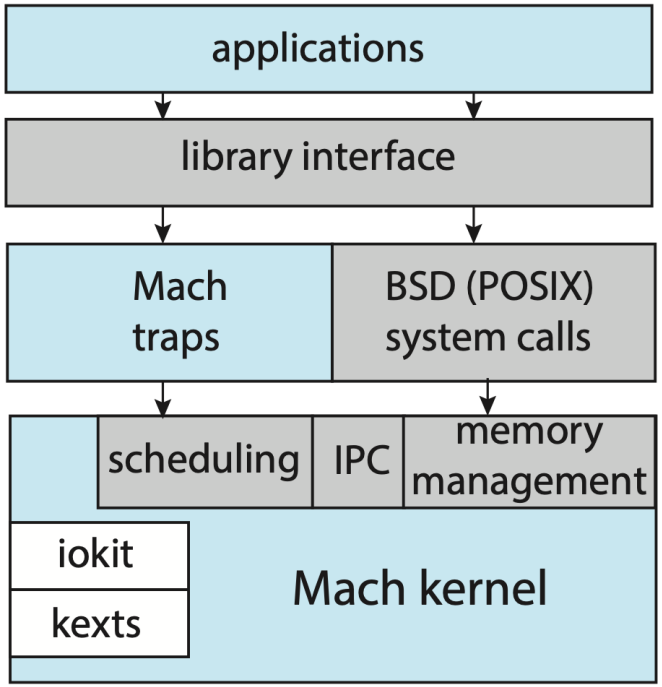


Fig. 2.17 in OSC book

Windows 10

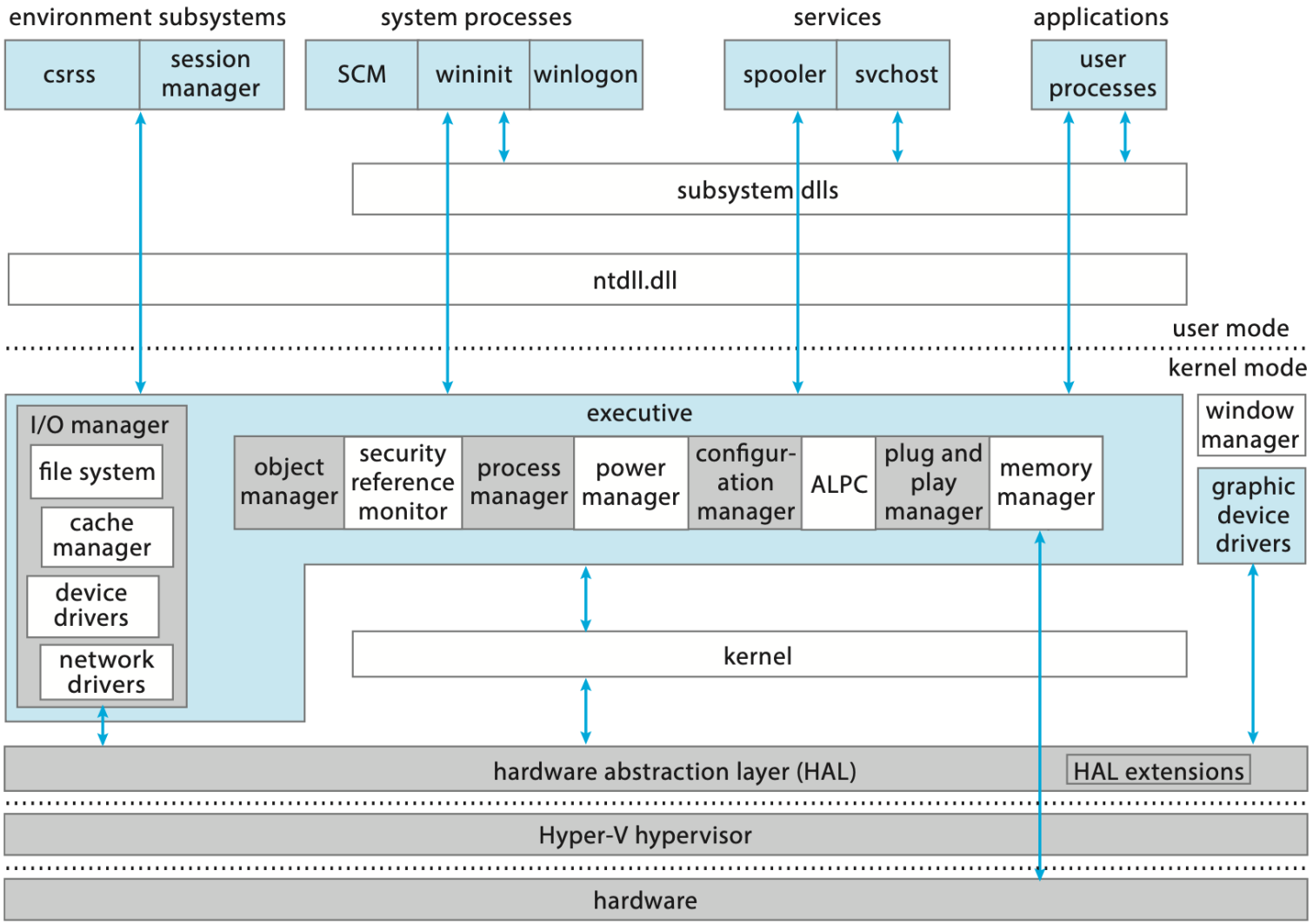


Fig. 21.1 in OSC book

Windows 10 (cont.)

- Layered architecture with loadable drivers in the I/O system ⇒ new file systems, new kinds of I/O devices, and new kinds of networking can be added while the system is running
- The lowest-level kernel "executive" runs in kernel mode and provides the basic system services and abstractions
- Windows 10 can also use its Hyper-V hypervisor to provide an orthogonal (logically independent) security model through Virtual Trust Levels (VTLs)
  - Normal World (VTL 0) ⇒ kernel mode (kernel executive, HAL, drivers) + user mode (system processes, Win32 subsystem, applications. . .)
  - Secure World (VTL 1) ⇒ kernel mode (secure kernel and the Hyper-V hypervisor component) + user mode (secure user mode)

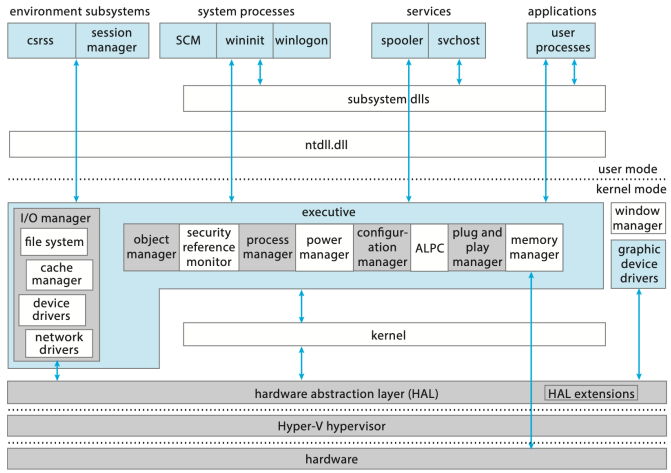


Fig. 21.1 in OSC book

Agenda

- Motivation and Introduction to Operating Systems
- Operating System Structure
- Operating System Architectures and Examples
- Summary

Summary

- An operating system is software that manages the computer hardware, as well as providing an environment for application programs to run
- An operating system provides an environment for the execution of programs by providing services to users and programs
- There is a variety of OS architectures that are driven by specific design goals
- Many contemporary operating systems are constructed as hybrid systems using a combination of a monolithic kernel and modules

Further resources:

- A Brief History of Operating Systems
  `http://homepage.divms.uiowa.edu/~jones/opsys/notes/03.shtml`
- Unix History `https://www.levenez.com/unix/`
- Operating Systems: An (Almost) Complete Timeline and Family Tree
  `http://maps-and-tables.blogspot.com/2019/01/almost-complete-timeline-and-family.html`