MONASH University

MONASH INFORMATION TECHNOLOGY

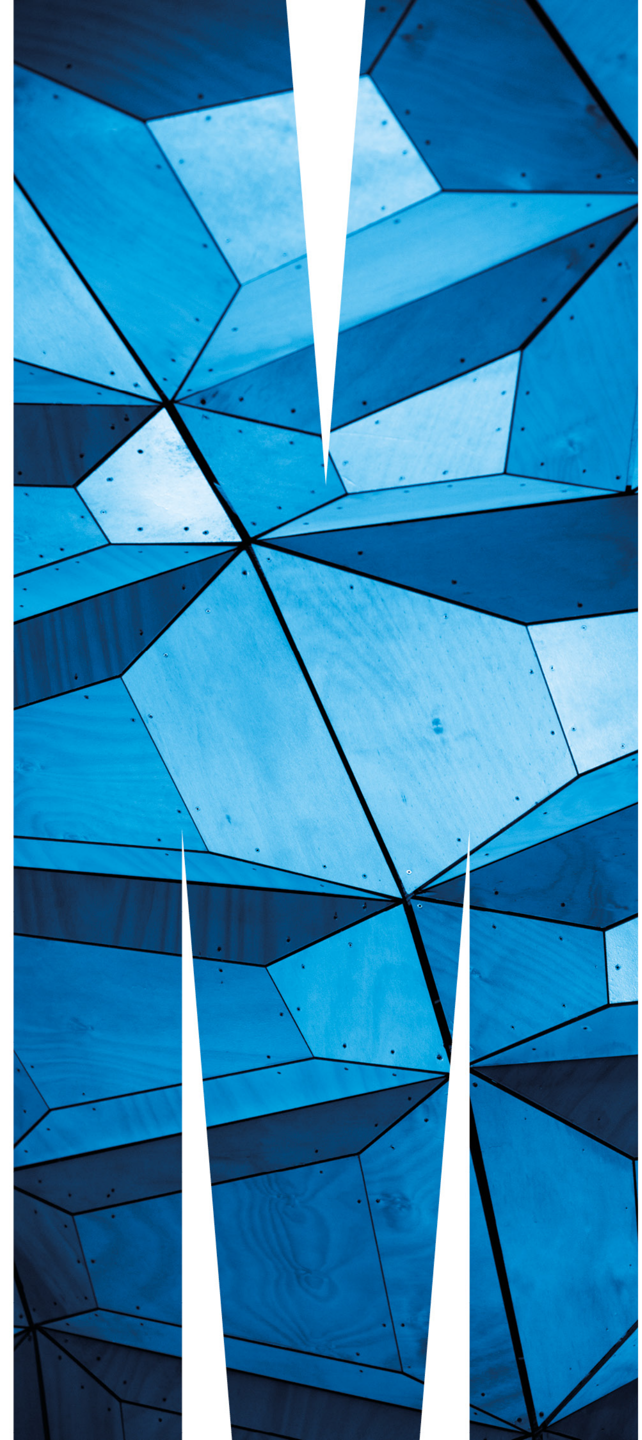
FIT2100 Semester 2 2017

Lecture 2: Operating System Overview

(Reading: Stallings, Chapter 2)

Jojo Wong

GROUP OF EIGHT AUSTRALIA



Lecture 2: Learning Outcomes

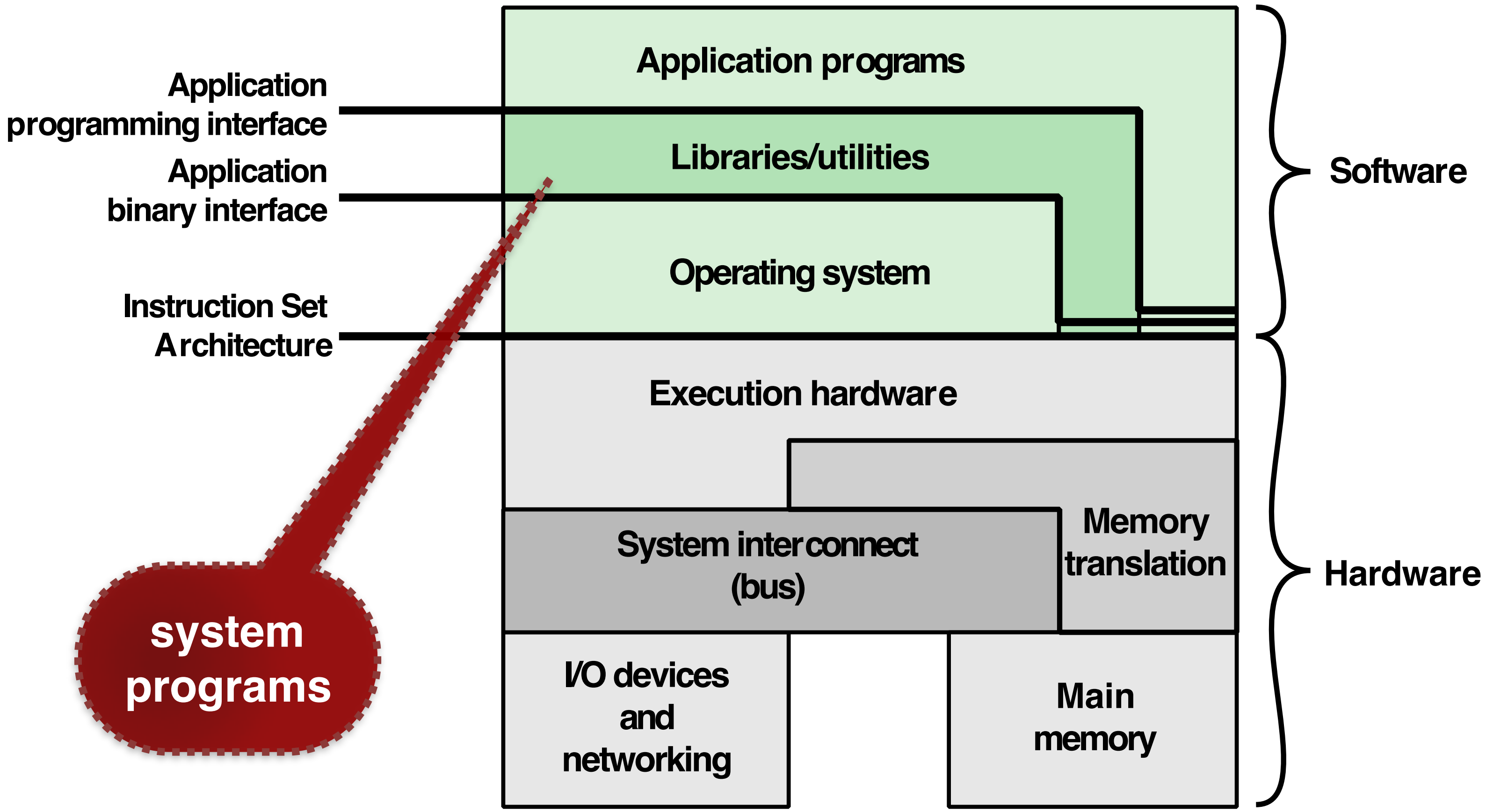
- Upon the completion of this lecture, you should be able to:
  - Understand the objectives and functions of an operating system (OS)
  - Discuss the evolution of operating systems from simple batch systems to modern complex systems
  - Discuss the key design areas of modern operating systems
  - Understand the OS architecture of Unix/Linux, Android, and Windows

What is an operating system?

Operating System

- A program that controls the execution of application programs
- An interface between applications and hardware
- Main objectives of an operating system:
  - Convenience
  - Efficiency
  - Ability to evolve

Computer Hardware and Software Structure



Operating System Services

- Program development
- Program execution
- Access I/O devices
- Controlled access to files
- System access
- Error detection and response
- Accounting

What is the role of an operating system?

The Role of an OS

- A computer is a set of resources for the movement, storage, and processing of data
- The OS is responsible for managing these resources
- The OS is in control of the computer's basic functions

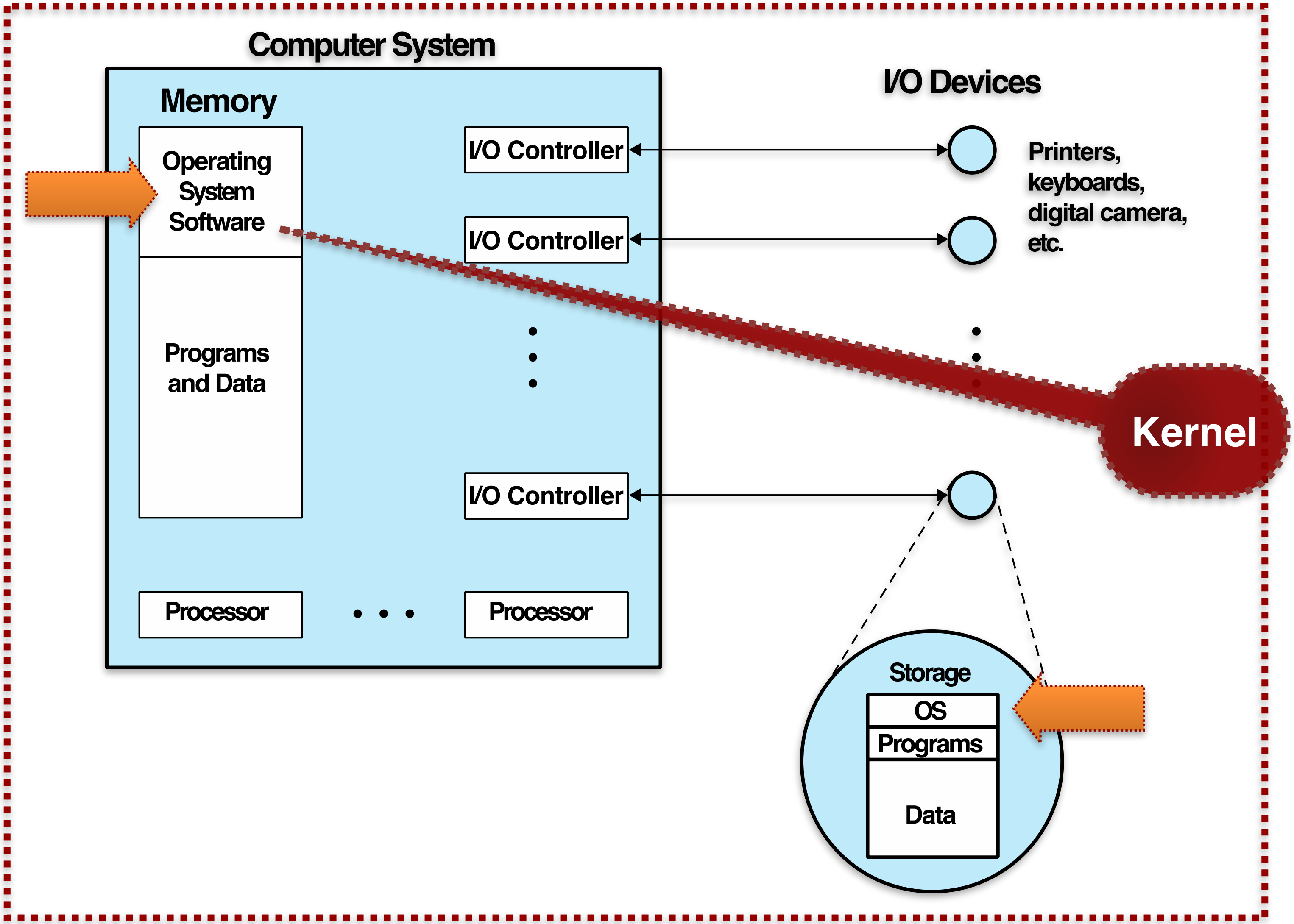
Operating System: As Software

- OS functions in the same way as ordinary computer software

- Program, or suite of programs, executed by the processor

- Frequently relinquishes control and must depend on the processor to allow it to regain control

Operating Systems: As Resource Manager



Operating System: Other Terms

- Resource allocator:
  - Manages and allocates resources to programs
- Control program:
  - Controls the execution of user programs and operation of I/O devices
- Kernel:
  - The one program running at all times (all else being application programs)

Unix/Linux OS

The evolution of operating systems

The Evolution of Operating Systems

- A major OS will evolve over time for a number of reasons:

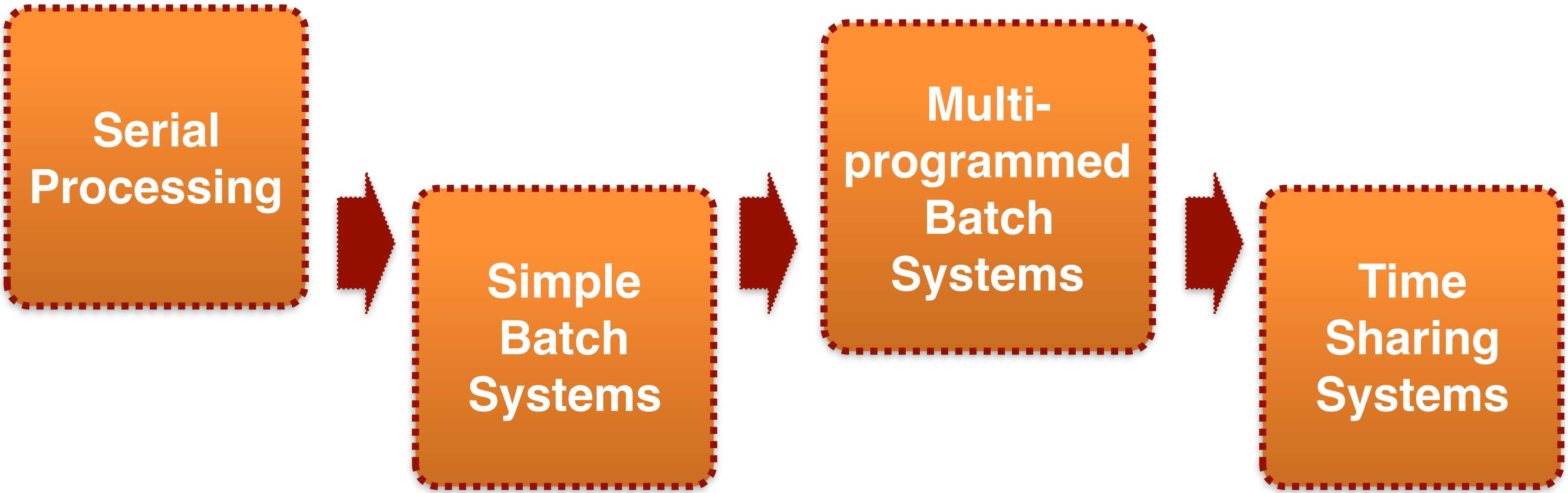
Hardware upgrades

New types of hardware

New services

Fixes

The Evolution of Operating Systems



Serial Processing

Earliest Computers

- No operating system
  - Programmers interacted directly with the computer hardware
- Computers ran from a console with display lights, toggle switches, some form of input device, and a printer
- Users have access to the computer in "series"

Problems

- Scheduling
  - Most installations used a hardcopy sign-up sheet to reserve computer time
  - Time allocations could run short or long, resulting in wasted computer time
- Setup time
  - A considerable amount of time was spent just on setting up the program (job) to run

Simple Batch Systems

- Early computers were very expensive
  - Important to maximise processor utilisation
- The concept of monitor:
  - Users no longer have direct access to the processor
  - Jobs are submitted to a computer operator who batches them together and places them on an input device
  - Program branches back to the monitor when finished

Simple Batch Systems: Job Scheduling

Special type of programming language (JCL) used to provide instructions to the monitor

What *compiler* to use?

What *data* to use?

Simple Batch Systems: Modes of Operation

User Mode

- User program executes in user mode
- Certain areas of memory are protected from user access
- Certain instructions may not be executed

Kernel Mode

- Monitor executes in kernel mode
- Privileged instructions may be executed
- Protected areas of memory may be accessed

Simple Batch Systems: Overhead

- Processor time alternates between execution of user programs and execution of the monitor

- Sacrifices:
  – Some main memory is given over to the monitor
  – Some processor time is consumed by the monitor

- Despite overhead, the simple batch system improves utilisation of the computer

Multiprogrammed Batch Systems

- Processor is often idle:
  - Even with automatic job sequencing
  - I/O devices are slow as compared to processor

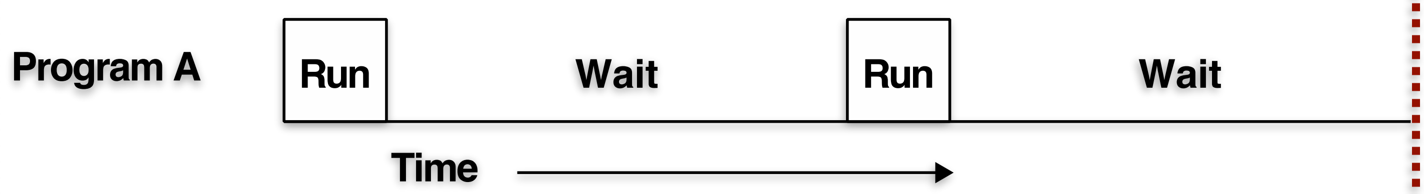
- Example: system utilisation

Read one record from file	15 μs
Execute 100 instructions	1 μs
Write one record to file	15 μs
TOTAL	31 μs

$$\text{Percent CPU Utilization} = \frac{1}{31} = 0.032 = 3.2\%$$

Uniprogramming

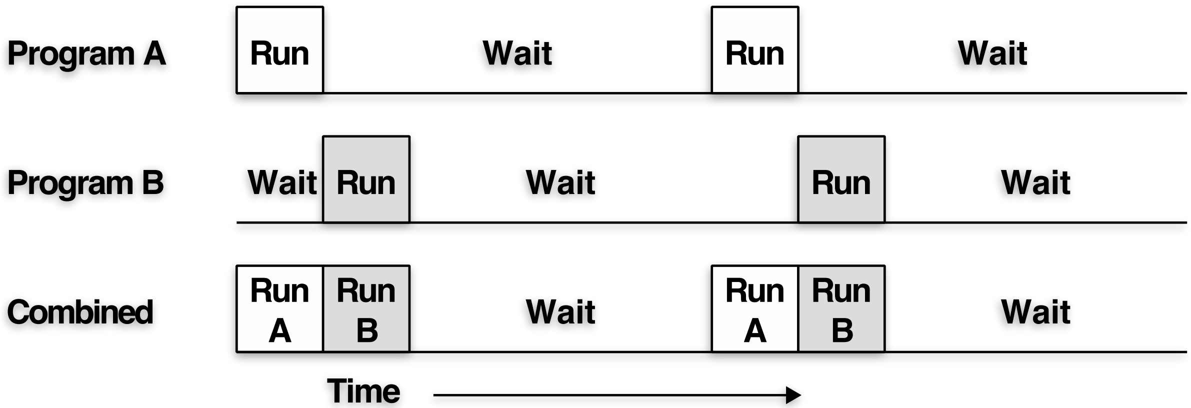
- The processor spends a certain amount of time executing, until it reaches an I/O instruction
- It must then wait until that I/O instruction concludes before proceeding



(a) Uniprogramming

Multiprogramming: Two Programs

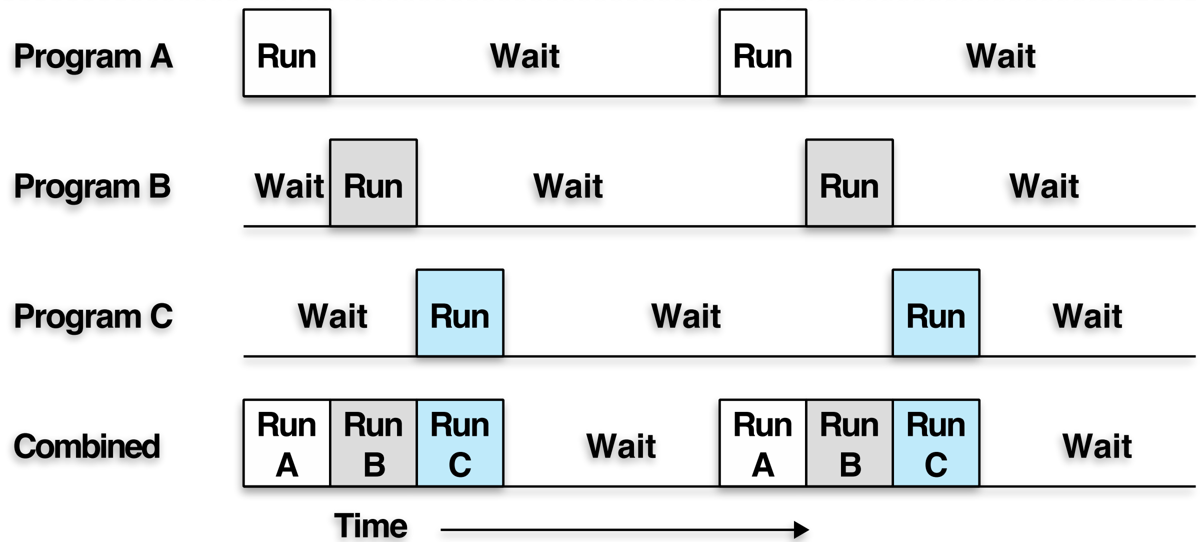
- When one job needs to wait for I/O, the processor can switch to the other job, which is likely not waiting for I/O



(b) Multiprogramming with two programs

Multiprogramming: Three Programs

- Memory is expanded to hold three, four, or more programs and switch among all of them
- A.k.a. multitasking

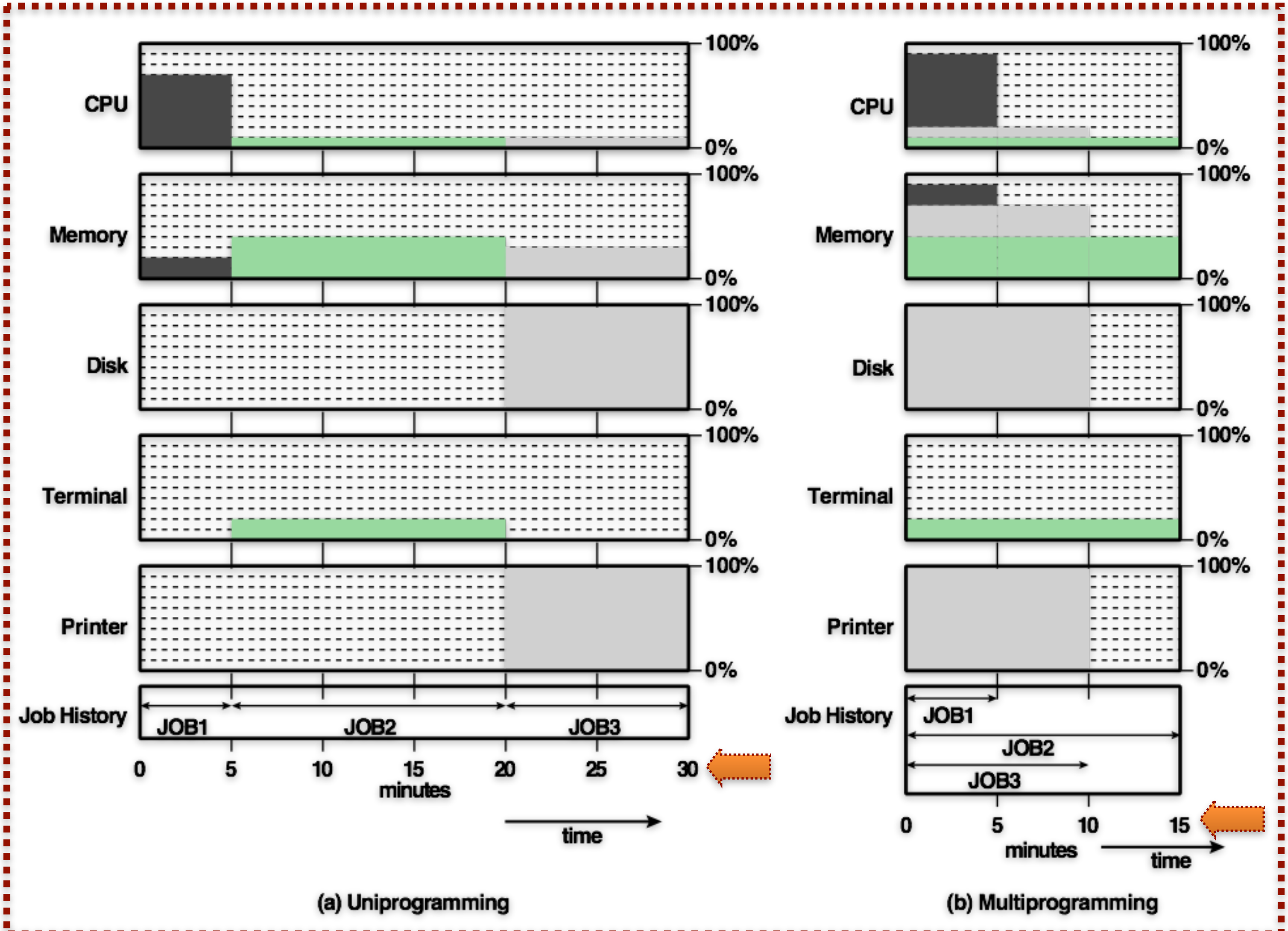


(c) Multiprogramming with three programs

Multiprogramming: Example

	JOB1	JOB2	JOB3
Type of job	Heavy compute	Heavy I/O	Heavy I/O
Duration	5 min	15 min	10 min
Memory required	50 M	100 M	75 M
Need disk?	No	No	Yes
Need terminal?	No	Yes	No
Need printer?	No	No	Yes

Multiprogramming: Effects on Resource Utilisation



Time-Sharing Systems

- Can be used to handle multiple interactive jobs
- Processor time is shared among multiple users
- Principle objective is to minimise response time
- Multiple users simultaneously access the system through terminals, with the OS interleaving the execution of each user program in a short burst or quantum of computation

Time Slicing

Compatible Time-Sharing Systems

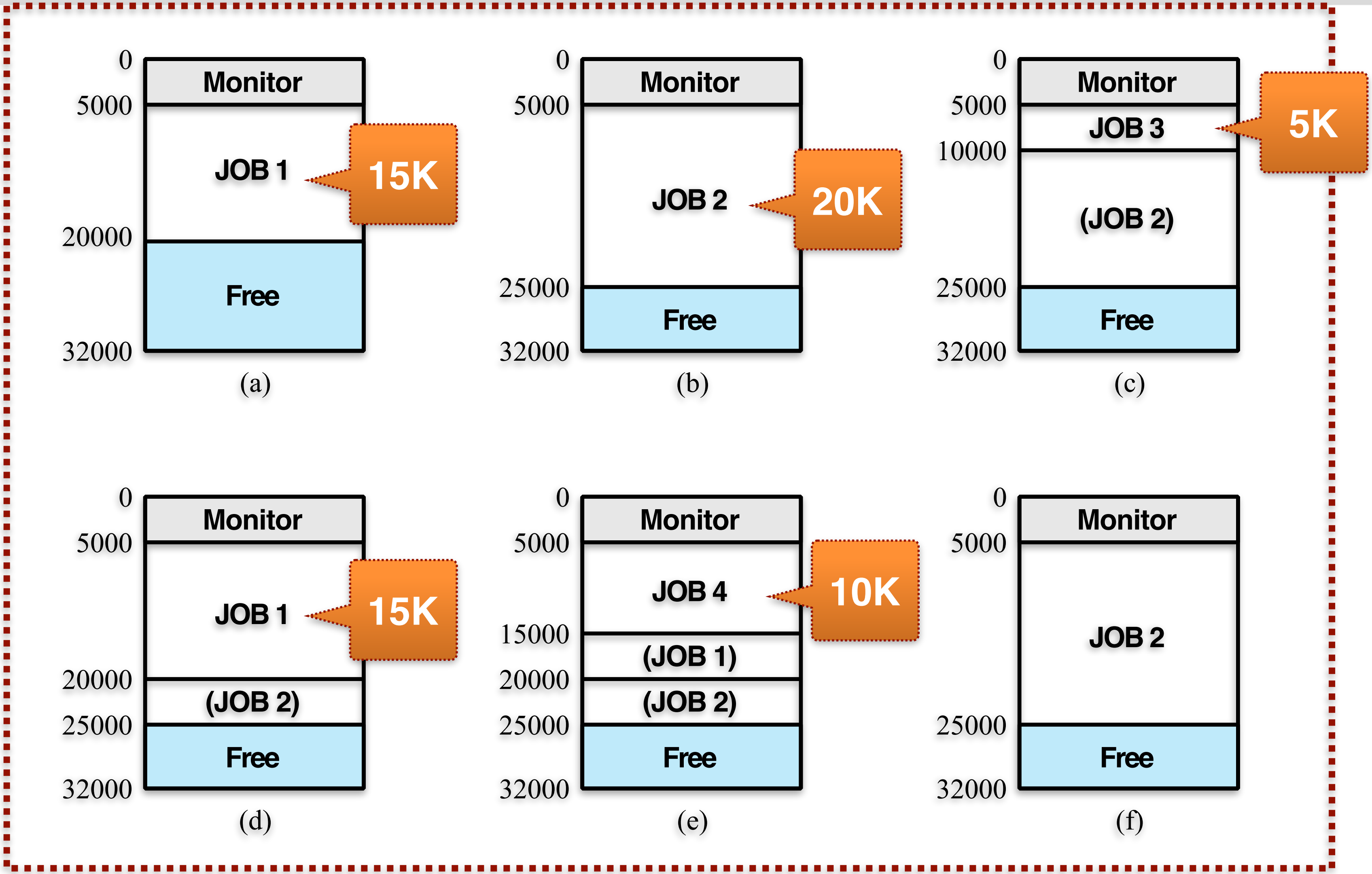
CTSS

- One of the first time-sharing operating systems
- Developed at MIT by a group known as Project MAC
- Ran on a computer with 32,000 36-bit words of main memory, with the resident monitor consuming 5000 of that
- To simplify both the monitor and memory management, a program was always loaded to start at the location of the 5000th word

Time Slicing

- System clock generates interrupts at a rate of approximately one every 0.2 seconds
- At each interrupt, OS regained control and could assign processor to another user
- At regular time intervals, the current user would be preempted and another user loaded in
- Old user programs and data were written out to disk; before new user programs and data were read in

CTSS: Operations



What are the major advances in OS development?

Major Achievements

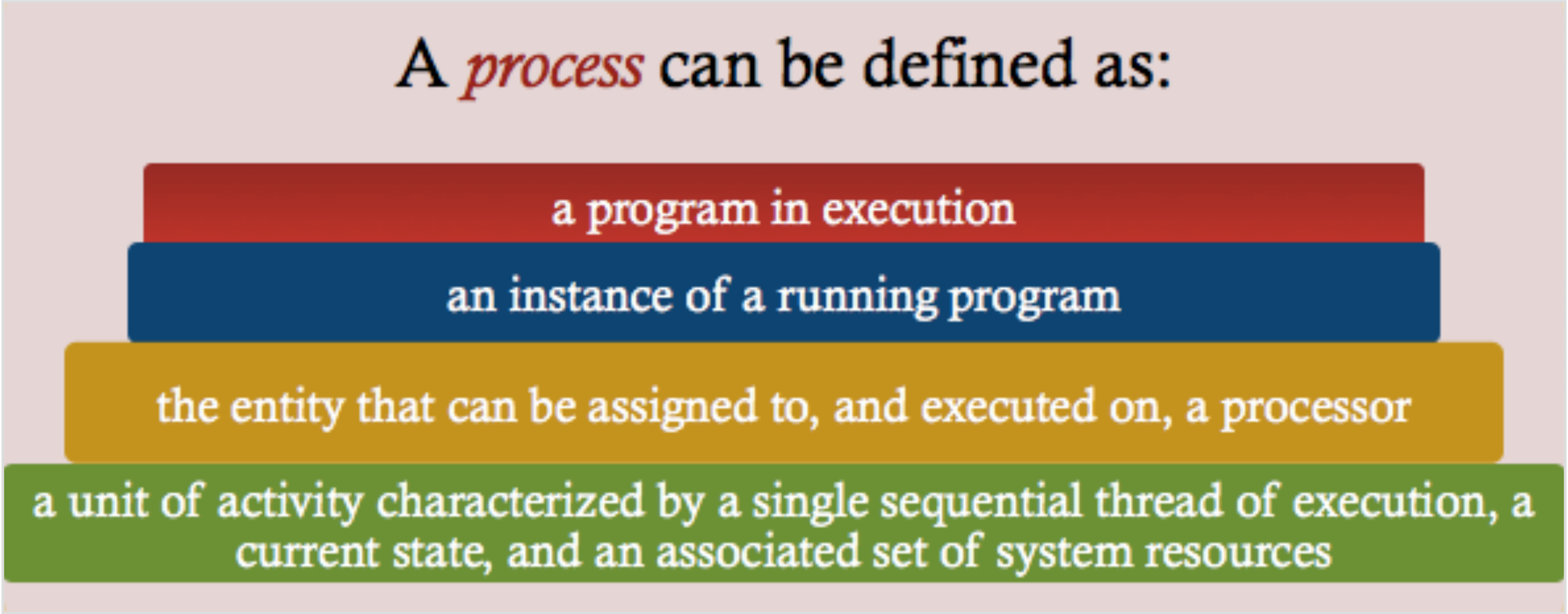
- Operating systems are among the most complex pieces of software ever developed

Major advances in OS development:

- Processes
- Memory management
- Information protection and security
- Scheduling and resource management
- System structure

The Concept of Process

- Fundamental to the structure of operating system



Development of the Concept of Process

- Three major lines of computer system development that created problems in timing and synchronisation

- Multiprogramming batch operation:
  - Processor is switched among the various programs residing in main memory
- Time sharing:
  - Be responsive to the individual users but be able to support many users simultaneously
- Real-time transaction systems:
  - A number of users are entering queries or updates against a database

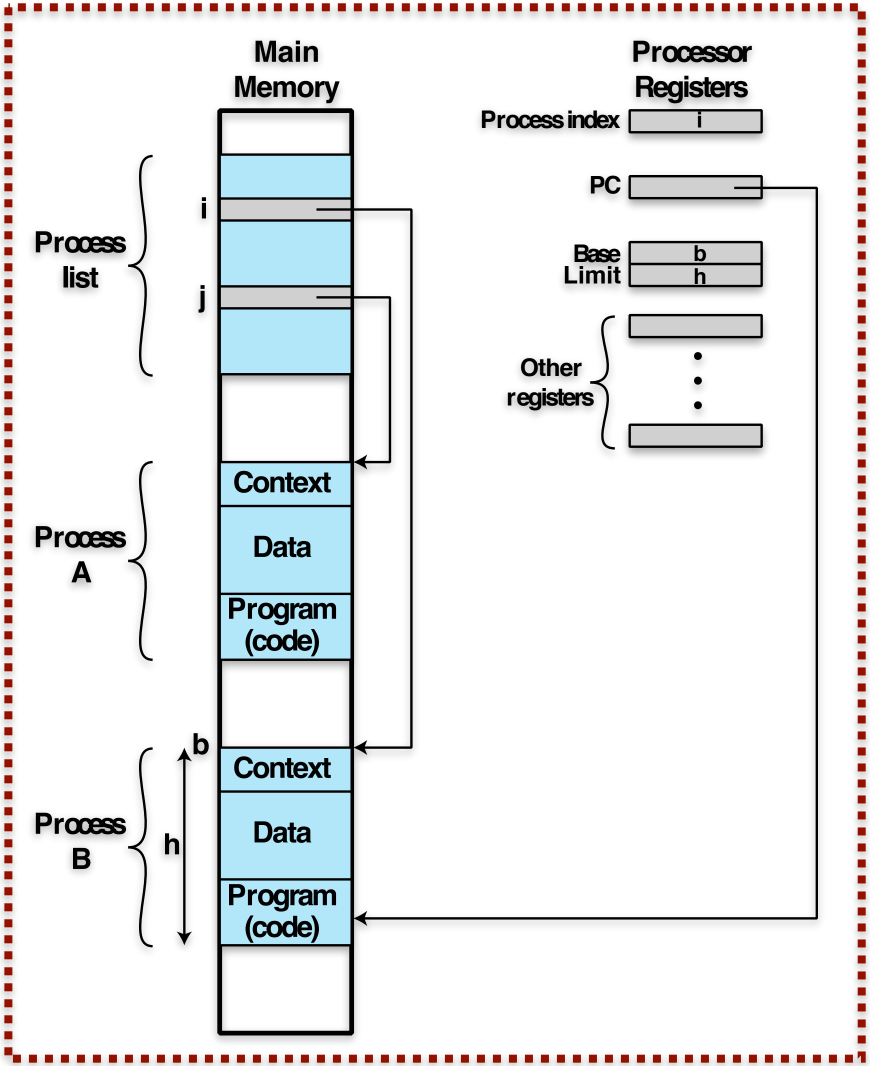
Process: Three Components

- A process contains three components:
  - An executable program
  - Associated data needed by the program (e.g. variables, work space, buffers, etc.)
  - Execution context (or process state) of the program

- The execution context is essential:
  - It is the internal data by which the OS is able to supervise and control the process
  - Includes the contents of the various process registers
  - Includes information such as the priority of the process; and whether the process is waiting for the completion of a particular I/O event

Process Management

- The process is realised as a data structure.
- The entire state of the process at any instant is contained in its context.
- New features can be designed and incorporated into the OS — by expanding the context to include any new information needed to support the features.



Memory Management

- OS has five principal storage management responsibilities:

  - Process isolation
  - Automatic allocation and management
  - Support of modular programming
  - Protection and access control
  - Long-term storage

- Can be achieved using virtual memory and file system facilities

Information Protection and Security

- The nature of the threat that concerns an organisation will vary greatly depending on the circumstances
- The concern involves controlling access to computer systems and the information stored in the systems
- Main issues:
  - Availability
  - Confidentiality
  - Data integrity
  - Authenticity

Scheduling and Resource Management

- The key responsibility of an OS is managing various system resources (main memory, I/O devices, processors)

- Resource allocation policies must consider:
  - Fairness: equal and fair access to resources by competing processes
  - Differential responsiveness: discriminate among different classes of processes
  - Efficiency: maximise throughput and minimise response time; accomodate as many users as possible

OS Architecture

- Demands on operating systems require new approaches to organising the OS architecture

Different approaches and design elements have been tried:

- microkernel architecture
- multithreading
- symmetric multiprocessing
- distributed operating systems
- object-oriented design

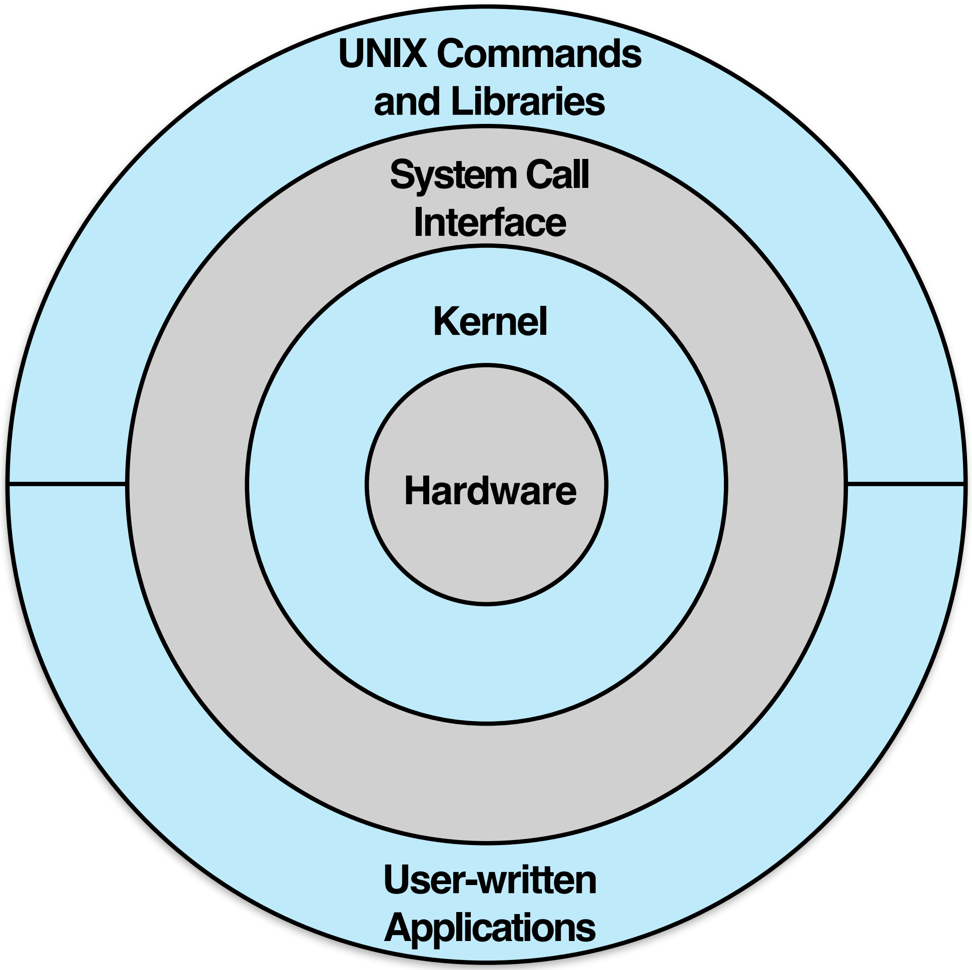
The OS architecture for Windows, Unix/Linux, and Android

Traditional Unix Systems

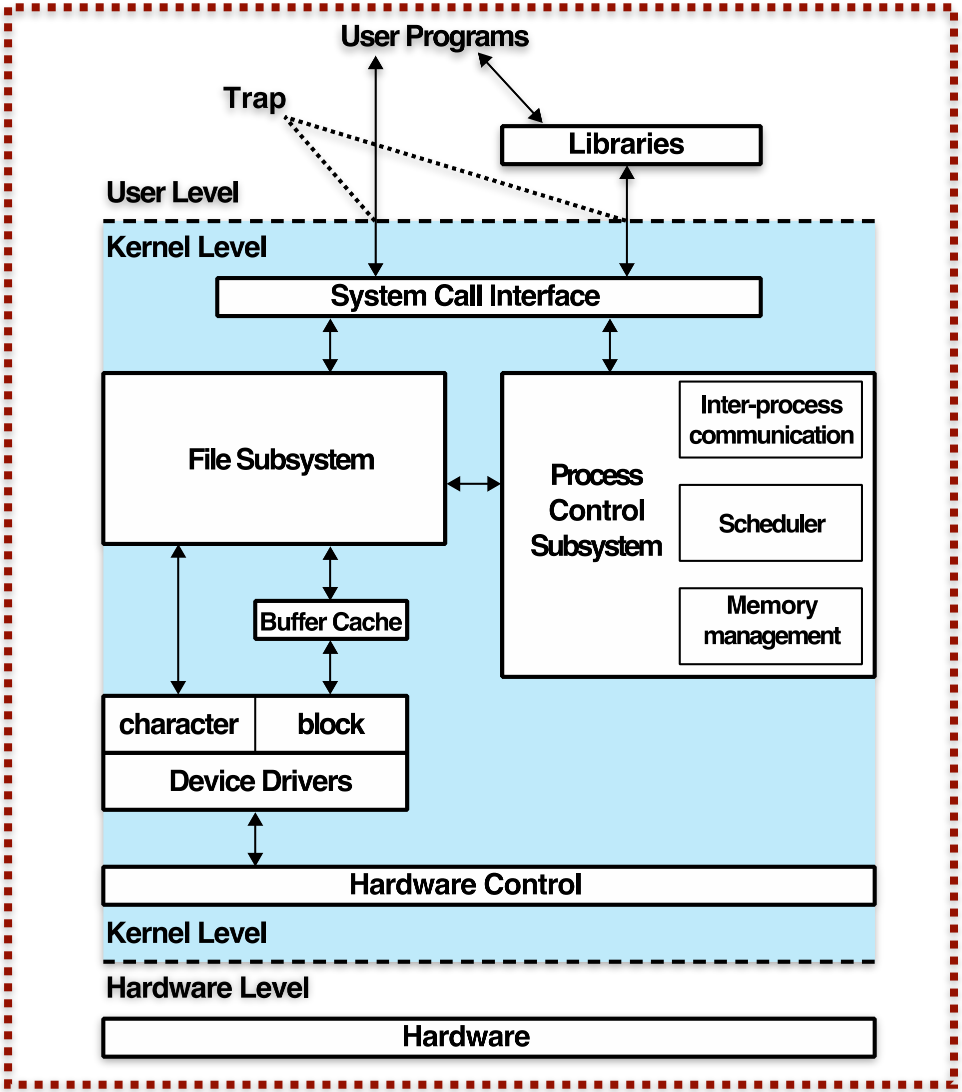
- Developed at Bell Labs and became operational on a PDP-7 in 1970
  - Incorporated many ideas from Multics (and also CTSS)
  - PDP-11 was a milestone because it first showed that UNIX would be an OS for all computers
- Next milestone was rewriting UNIX in the programming language C
  - Demonstrated the advantages of using a high-level language for system code
- Described in a technical journal for the first time in 1974
- First widely available version outside Bell Labs was Version 6 in 1976
  - *Version 7* released in 1978 is the ancestor of most modern UNIX systems
- Commercially marketed as Unix System III and Unix System V
- Most important of the non-AT&T systems was Unix BSD

Berkeley Software Distribution

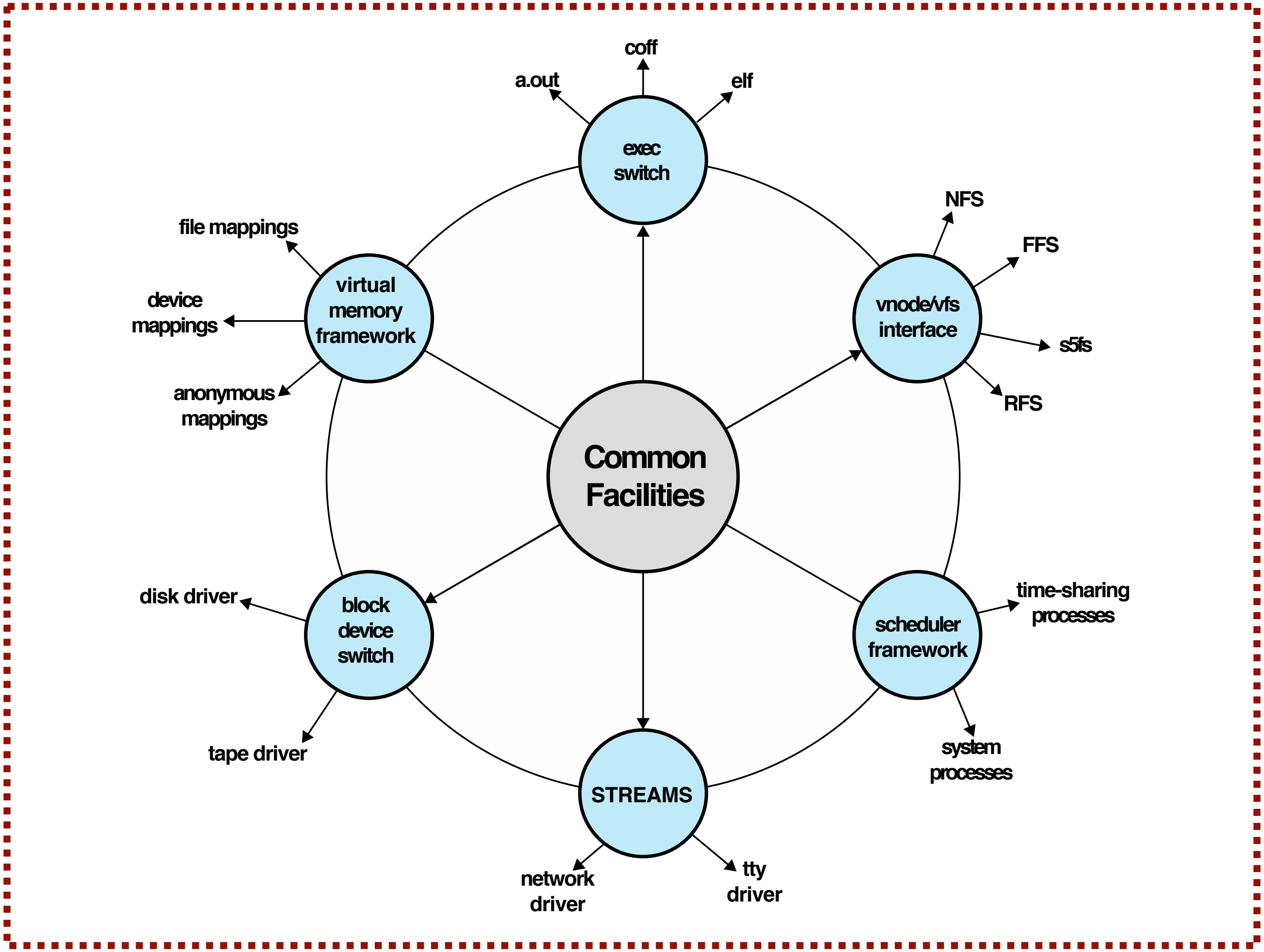
Unix: General Architecture



Unix Kernel: Traditional Architecture



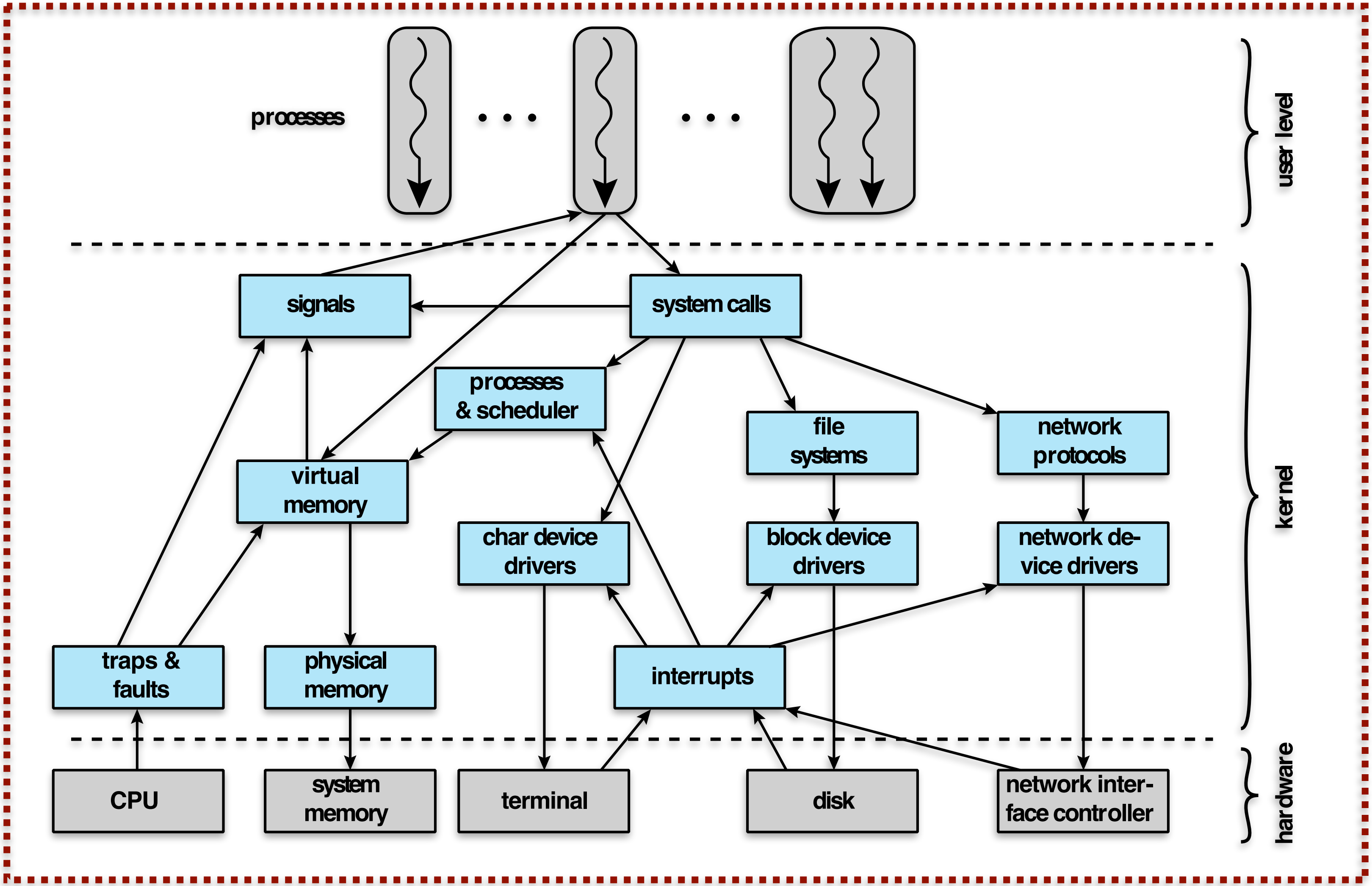
Unix Kernel: Modern Architecture



Linux Systems

- Started out as a UNIX variant for the IBM PC
- Initial version was written by Linus Torvalds, a Finnish student of computer science
- Linux was first posted on the Internet in 1991
- Today it is a full-featured UNIX system that runs on several platforms
- Free and source code is available (GNU Public License)
- Key to success has been the availability of free software packages
- Highly modular and easily configured

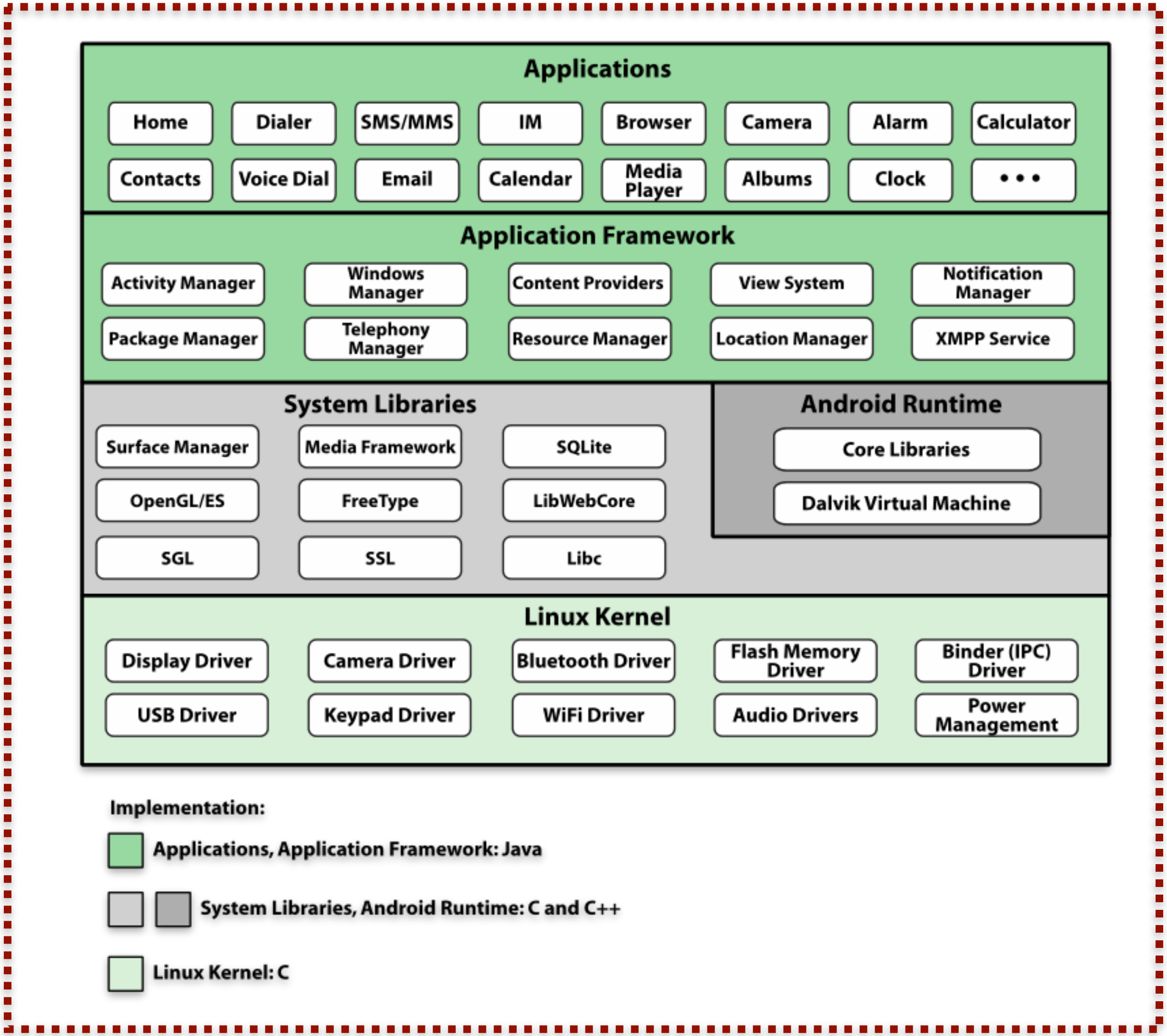
Linux: Kernel Components



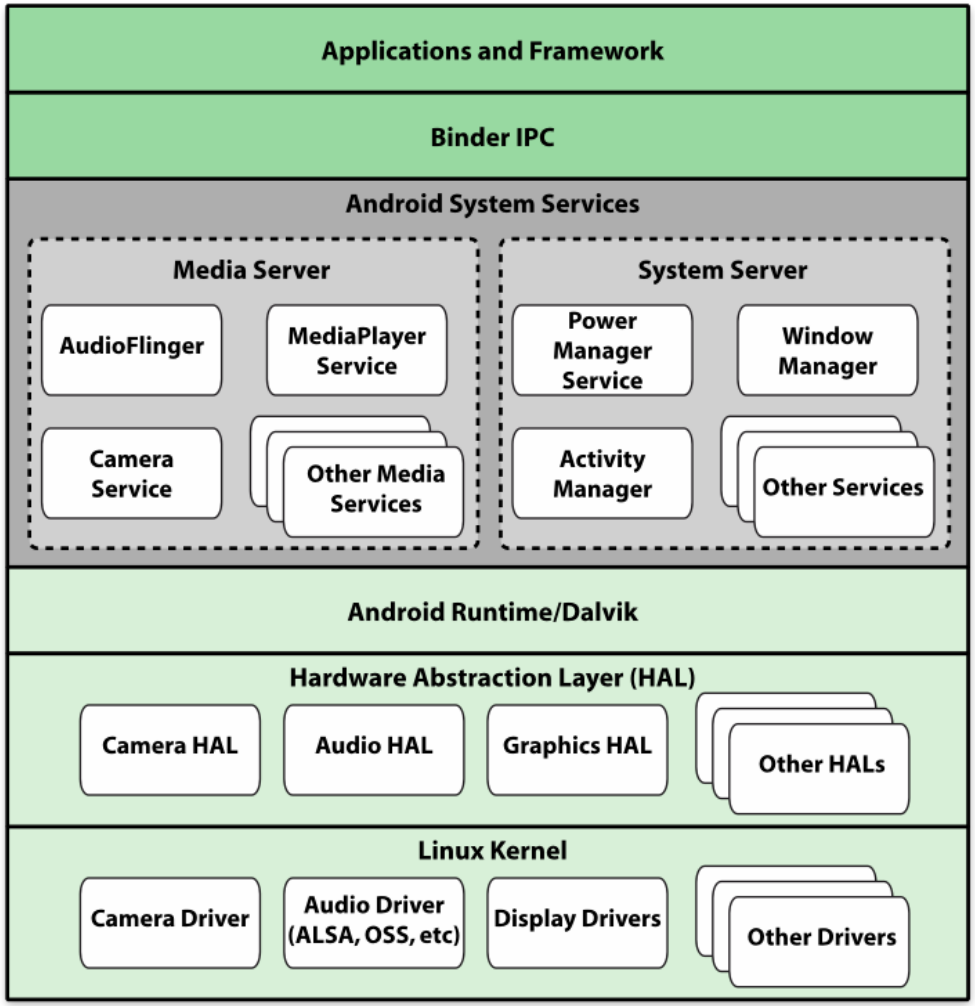
Android OS

- A Linux-based system originally designed for touchscreen mobile devices such as smartphones and tablet computers
- Development was done by Android Inc., which was bought by Google in 2005
- 1st commercial version (Android 1.0) was released in 2008
- Most recent version is Android 7.0 (Nougat)
- The Open Handset Alliance (OHA) was responsible for the Android OS releases as an open platform
- The open-source nature of Android has been the key to its success

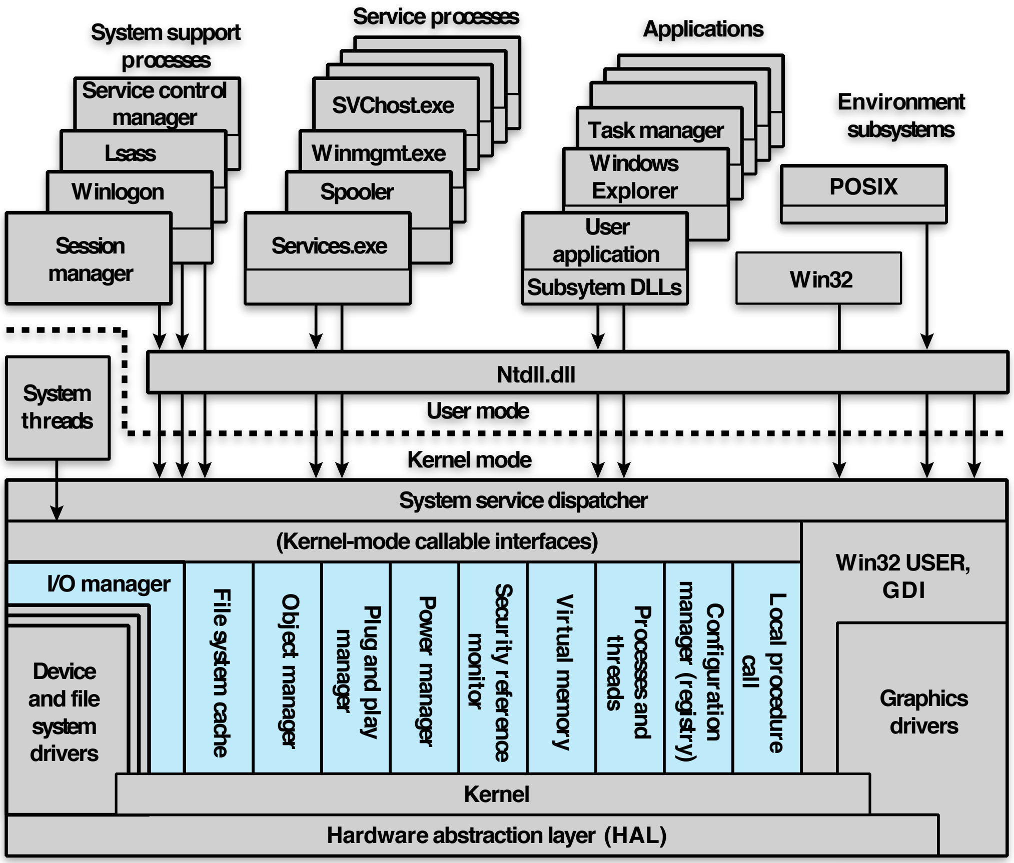
Android: Software Architecture



Android: System Architecture



Microsoft Windows: System Architecture



Lsass = local security authentication server
POSIX = portable operating system interface
GDI = graphics device interface
DLL = dynamic link libraries

Colored area indicates Executive

Windows 8

Summary

- So far, we have discussed:
  - Objectives and functions of OS
  - Evolution of operating systems
  - Key design areas of modern OS
  - OS architecture of Unix/Linux, Android, and Windows

- Next week:
  - Concepts of Processes and Threads

Reminder: The first practical this week.