Textbook:
Operating Systems – Internals and Design Principles,
by ***William Stallings.***
Publisher: Pearson.

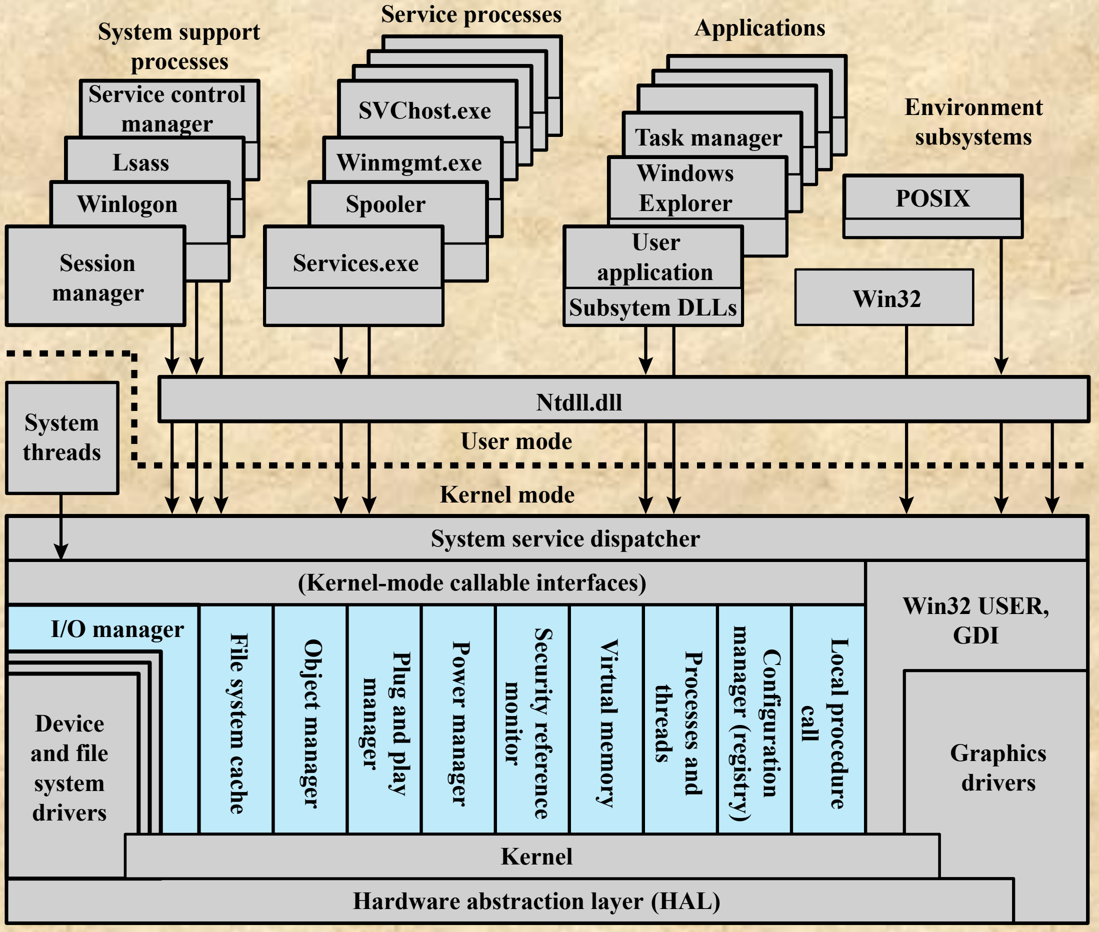
Chapter 2
Operating Systems Overview

Part C

“A short Walk through the Zoo”

- The various **fundamental principles and theoretical techniques**, which we have already looked at **in parts A and B of this triple-lecture on Chapter 2**, are combined and implemented differently in the various Operating Systems which are (or have been) available in academia and industry.

- In **this final part of our discussion of Chapter 2**, which covers the **remainder of this chapter from Section 2.7 onwards**, we will take a glimpse at several different Operating Systems in order to see how the general theoretical concepts, which are generally valid, have been put to practice in different concrete forms and OS implementations.



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

Colored area indicates Executive

Figure 2.14 Windows Architecture

Kernel-Mode Components of Windows

- Executive
  - Contains the core OS services, such as memory management, process and thread management, security, I/O, and interprocess communication
- Kernel
  - Controls utilization of the CPUs. The Kernel manages thread scheduling, process switching, exception and interrupt handling, and multiprocessor synchronization
- Hardware Abstraction Layer (HAL)
  - Maps between generic hardware commands and responses and those unique to a specific platform and isolates the OS from platform-specific hardware differences
- Device Drivers
  - Dynamic libraries that extend the functionality of the Executive. These include hardware device drivers that translate user I/O function calls into specific hardware device I/O requests and software components for implementing file systems, network protocols, and any other system extensions that need to run in kernel mode
- Windowing and Graphics System
  - Implements the GUI functions, such as dealing with windows, user interface controls, and drawing

SMP

- "Windows" is also known for its support of symmetric multiprocessing (SMP)
  - **OS routines can run on any available CPUs, whereby different routines can run simultaneously on different CPUs**
  - **Windows provides mechanisms for sharing data and resources between processes and flexible interprocess communication capabilities**

UNIX

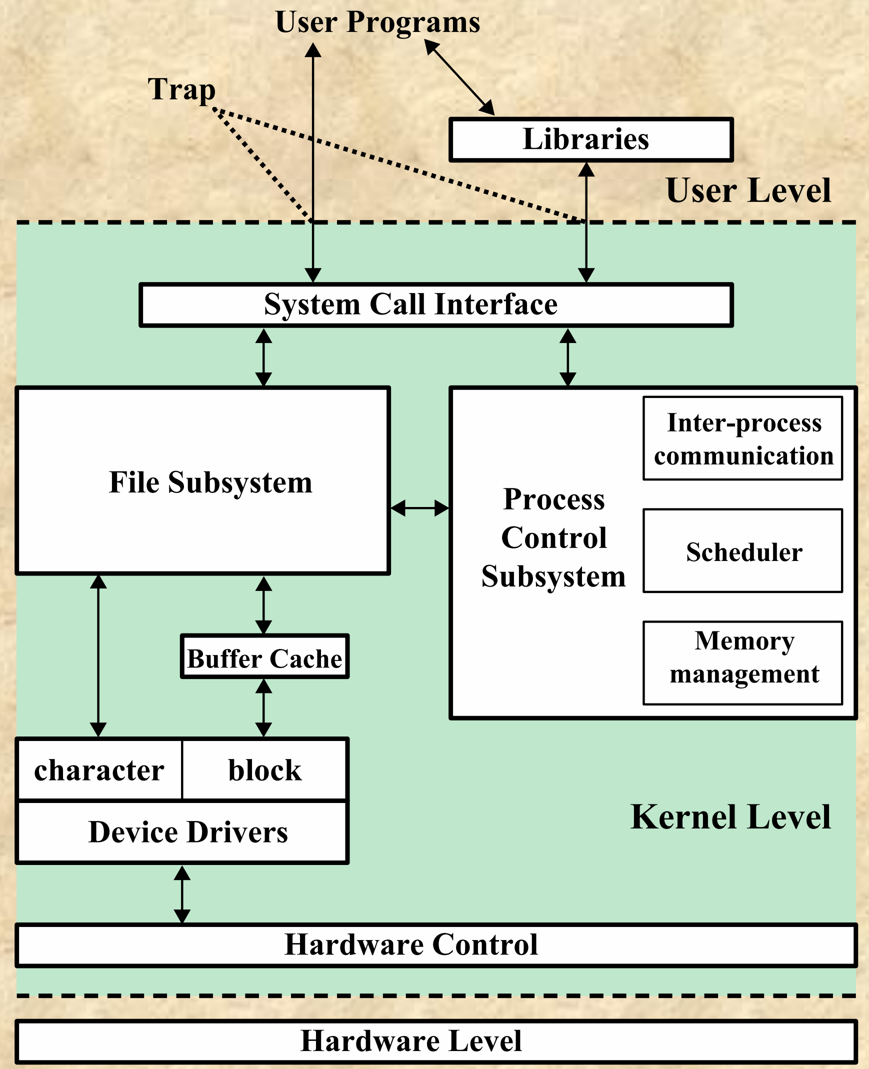


Figure 2.15 Traditional UNIX Kernel

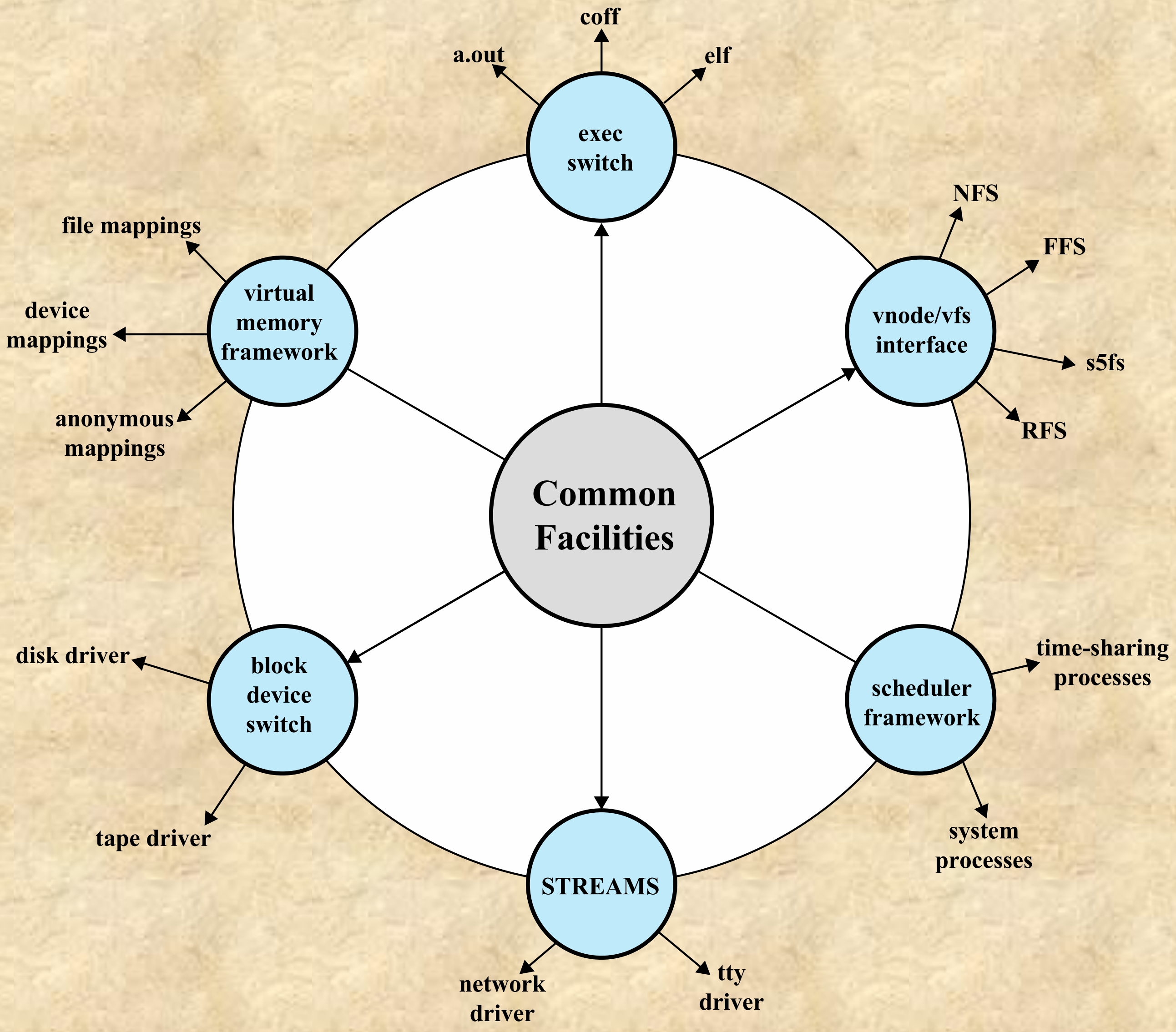


Figure 2.16 Modern UNIX Kernel

LINUX Overview

- Started as a UNIX variant for the IBM PC
- Linus Torvalds, a Finnish **student of computer science, programmed the initial version**
- Linux was first posted on the Internet in 1991
- Today it is a full-featured UNIX system that runs on virtually all platforms
- Is free and the source code is available
- Key to the success of Linux has been the availability of free software packages under the auspices of the Free Software Foundation (FSF)
- Highly modular and easily configured

Modular Structure

- Linux development is global and done by a loosely associated group of independent developers
- Although Linux does not use a microkernel approach, it achieves many of the potential advantages of the approach by means of its particular modular architecture
- Linux is structured as a collection of modules, a number of which can be automatically loaded and unloaded on demand

Loadable Modules

- Relatively independent blocks
- A module is an object file whose code can be linked to and unlinked from the kernel at runtime
- A module is executed in kernel mode on behalf of the current process
- Have two important characteristics:
  - Dynamic linking
  - Stackable modules

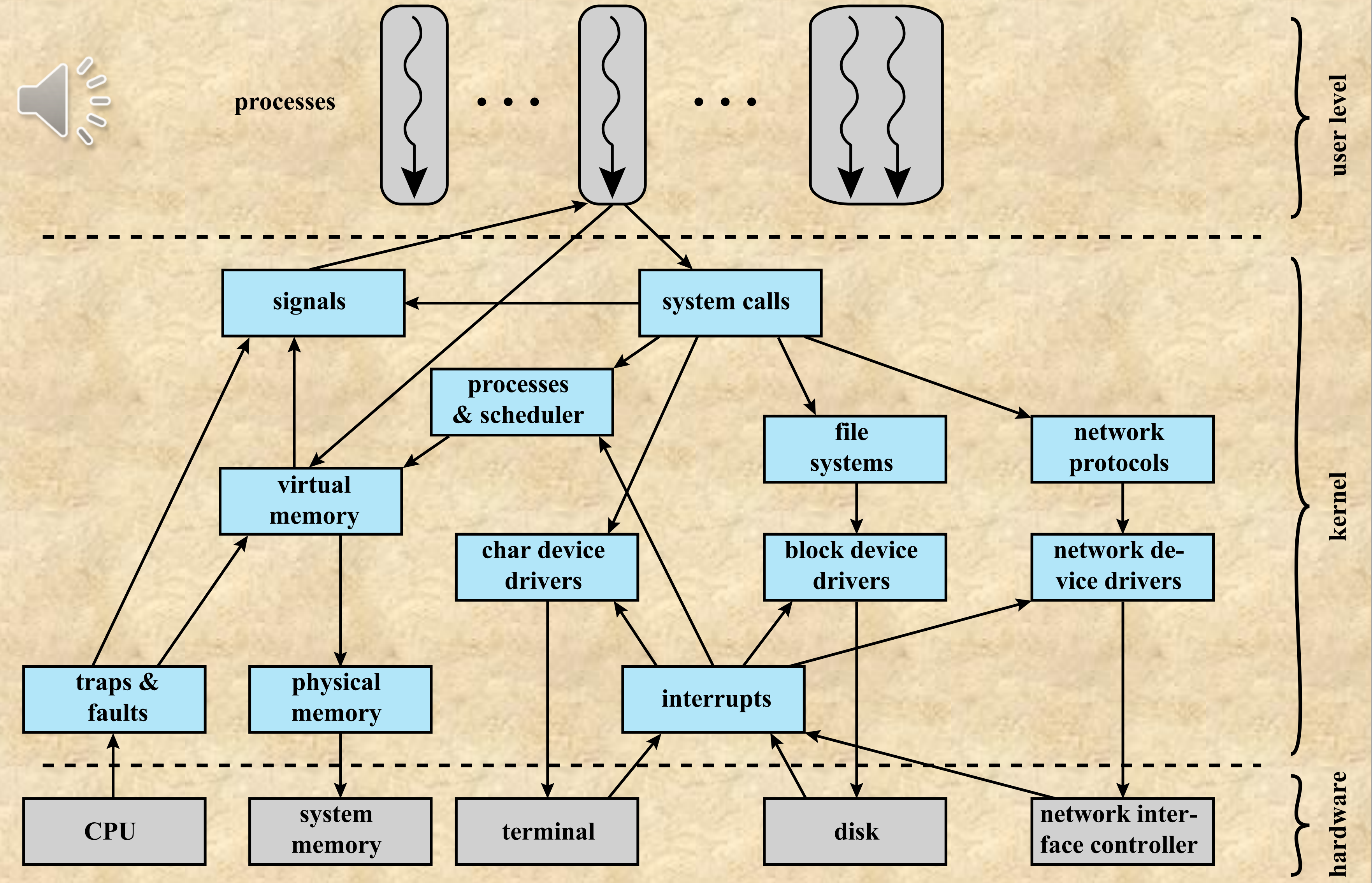
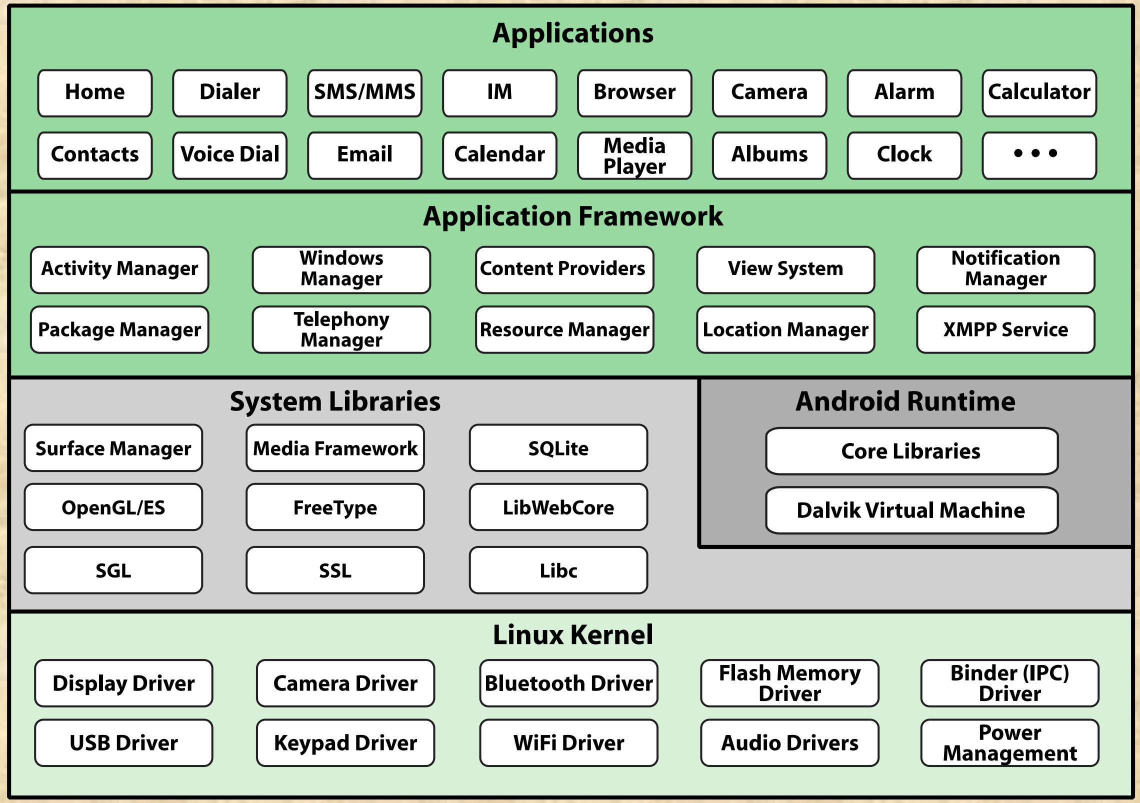


Figure 2.19 Linux Kernel Components

Android Operating System

- A Linux-based system originally designed for mobile phones
- The most popular mobile OS
- 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)
- Android has an active community of developers and enthusiasts who use the Android Open Source Project (AOSP) source code to develop and distribute their own modified versions of the operating system
- The open-source nature of Android has been the key to its success



Implementation:

🟩 Applications, Application Framework: Java

⬜ ⬛ System Libraries, Android Runtime: C and C++

⬜ Linux Kernel: C

Figure 2.20 Android Software Architecture

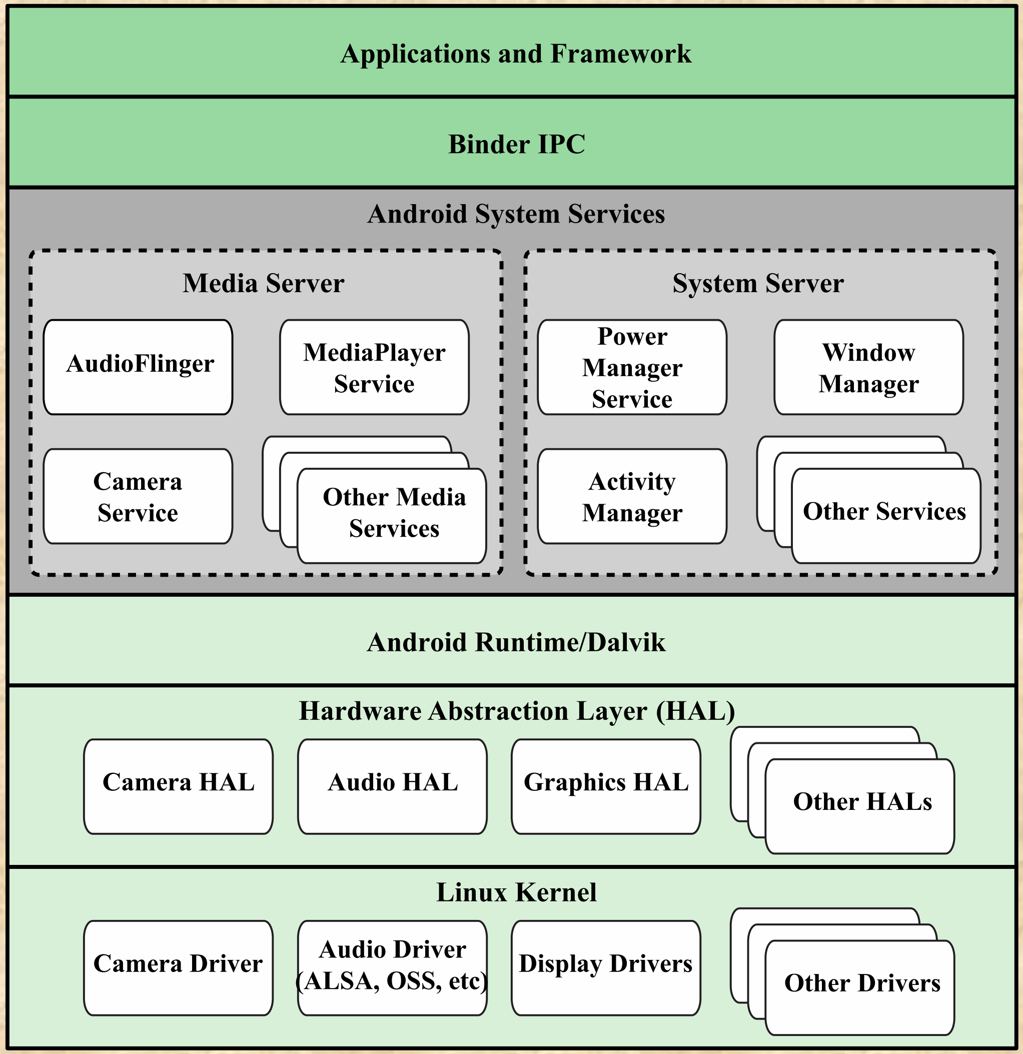


Figure 2.22 Android System Architecture

Power Management: new features

Alarms

- Implemented in the Linux kernel and is visible to the app developer through the AlarmManager in the RunTime core libraries
- Is implemented in the kernel so that an alarm can trigger even if the system is in sleep mode
    - This allows the system to go into sleep mode, saving power, even though there is a process that requires a wake up

Wakelocks

- Prevents an Android system from entering into sleep mode
- These locks are requested through the API whenever an application requires one of the managed peripherals to remain powered on
- An application can hold one of the following wakelocks:
    - Full_Wake_Lock
    - Partial_Wake_Lock
    - Screen_Dim_Wake_Lock
    - Screen_Bright_Wake_Lock

Conclusion of Chapter 2

- **Chapter 2** mainly served the purpose of an "**Extended Table of Contents**" in our textbook.
- It provided you with an overview of what an operating system is, of what most important components it is made, and what are the most crucial issues which the designer of an operating system must take into consideration.
- Many further details of all this are explained and discussed in the subsequent chapters of the book.