CS2106 Intro Op. Systems Notes

AY23/24 Sem 2, github.com/gerteck

1. Introduction

Course objectives: Introduces basic concepts in operating systems. Focusing on OS structure and architecture, process management, memory management, file management and OS protection mechanism. At the end of the course, identify and understand major functionalities of modern operating systems and extend and apply the knowledge in future related courses.

Supplementary Text: Modern Operating System (5th Edition), by Andrew S. Tanenbaum, Pearson, 2023.

Learning Outcomes

- Understand how an **OS manages computational resources for multiple users and applications, and the impact on application performance**
- Appreciate the **abstractions and interfaces provided by OS**
- Write **multi-process / thread programs** and avoid common pitfalls such as **deadlocks, starvation and race conditions.**
- Write system programs that utilizes **POSIX** syscall for process, memory and I/O management.
- Self-learn and explore advanced OS topics.
- Understand important design principles in complex systems.

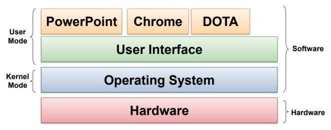
Areas to focus on: Try to understand how things are running in parallel, since we naturally think sequentially. Secondly, how we can manage memory and how they combine and interact (in strange ways), synchronization.

Operating System OS

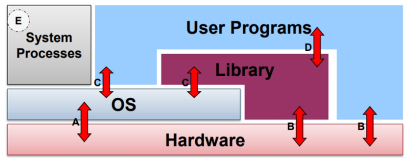
An OS is a program that acts as an intermediary between a computer user and the computer hardware. Motivation for OS:

- Manage resources and coordination. (Resource Allocator: Process synchronization, resource sharing)
- Simplify programming (Abstraction of hardware / hardware virtualization, convenient services)
- Enforce usage policies
- Security and protection
- User Program Portability (across different hardware)
- Efficiency (Optimize for particular usage and hardware).

Kernel Mode: Complete access to all hardware resources.
User Mode: Limited / Controlled access to hardware resources.



Generic OS Components



- **A**: OS executing machine instructions
- **B**: normal machine instructions executed (program/library code)
- **C**: calling OS using ***system call interface***
- **D**: user program calls library code
- **E**: system processes
  - Provide high level services, usually part of OS

- OS is known as the **kernel**.
  Program that deals with hardware issues, provide system call interface and special code for interrupt handlers, device drivers.
- Kernel code is different from normal programs:
  No use of system call in kernal code, can't use normal libraries, no normal I/O (must do I/O itself).
- **Implementing OS**: Historically in assembly/machine, now in HLLs (C, C++). Heavily hardware achitecture dependent. Challenges include complexity, debugging, codebase size.

OS Structures

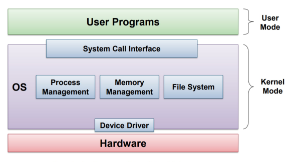
Monolithic OS: One Big program.

- Well understood, good performance, but highly coupled components (everything running in kernel mode) and usually devolved into very complicated internal structure.

Microkernel OS:

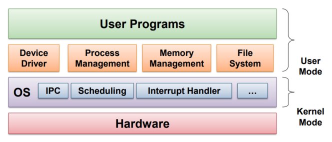
- Kernel is very small and clean, only providing basic and essential facilities.
- Inter-Process Communication **(IPC)**, Address space management, Thread management etc.
- Higher level services are built on top of basic facilities, run as server process *outside* of OS, use IPC to communicate.
- Kernel is more robust and extendible, better isolation and protection between kernel and high level services. But, lower performance. (Latency)

Monolithic Kernel Illustration



Generic Architecture of Monolithic OS Components

Microkernel Components



Generic Architecture of Microkernel OS Components

Other OS Structure

- **Layered Systems**: Generalization of monolithic system, organize components into hierarchy of layers. Lowest is hardware, highest is user interface.
- **Client-Server Model**: Variation of microkernel. Two classes of processes: Client p. request service from server process, server process built on top of microkernel. Client & Server process can be on separate machine.

Upcoming Topics

- 2. Process Abstraction
- 3. Process Scheduling
- 4. Inter-Process Communication
- 5. Threads + Synchronization
- 6. Memory Management
- 7. Disjoint Memory Management
- 8. File System Management
- 9. File System Implementation

End.