CS 350 Operating Systems

Keven Qiu
Instructor: Bernard Wong
Winter 2024

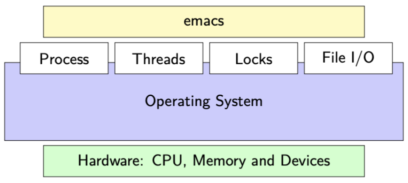
Chapter 1

Introduction

Definition: Operating System (OS)

The layer between applications and hardware.

It usually provides abstractions for applications. This includes managing and hiding details of hardware and can access hardware through low level interfaces unavailable to applications. It often provides protection.



Primitive OS are just a library of standard services (no protection). The system runs one program at a time and there are no bad users or programs.

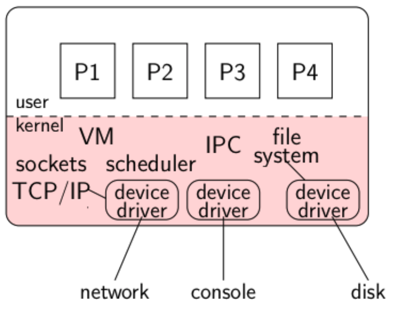
Multitasking is the idea of running more than one process at once. An ill-behaved process can go into an infinite loop and never relinquish the CPU. It can also scribble over other processes' memory. The OS provides mechanisms to address these problems:

- Preemption: take CPU away from looping.
- Memory protection: protect process' memory.

Multi-user OS use protection to serve distrustful users/apps. With n users, the system is not n times slower. Users can use too much CPU (need policies), total memory usage is greater than in the machine (must virtualize), or a super-linear slowdown with increasing demand (thrashing).

The OS structure: Most software runs as user-level processes and the OS kernel runs in privileged mode.

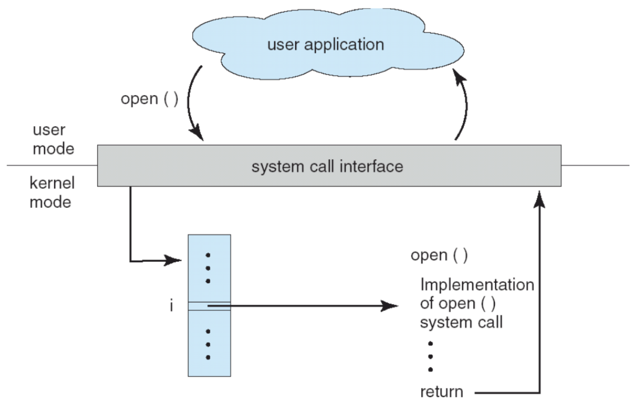
Protection:



- Pre-emption: give application a resource, take it away if needed elsewhere.
- Interposition/mediation: place the OS between application and "stuff". Track all pieces that the application is allowed to use. On every access, look in the table to check that access is legal.
- Privileged & unprivileged modes in CPUs: applications are unprivileged and the OS is privileged. Protection operations can only be done in privileged mode.

Definition: System Calls

Applications can invoke the kernel through system calls. These are special instructions that transfers control to the kernel.



The goal for system calls is to do things applications cannot do in unprivileged mode. The kernel supplies well-defined system call interface.

Applications set up syscall arguments and trap to kernel. The kernel performs operation and returns result.

Higher-level functions are built on the syscall interface. E.g. `printf`, `scanf`, `gets`, etc.

These call the POSIX/UNIX interface. E.g. `open`, `close`, `read`, `write`, etc.

Chapter 2

Processes

Definition: Process

An instance of a program running and the environment for running the program.

Modern OSes run multiple processes simultaneously.

- Multiple processes increase CPU utilization: overlap one process' computation with another.
- Multiple processes can reduce latency: running A then B may require more time than running A and B concurrently.