50004 - Operating Systems - Lecture 1

Oliver Killane

12/10/21

Lecture Recording

Lecture recording is available here

Introduction

Covering the theoretical side of OS, with many concepts to be implemented in the Pintos lab.

Course Delivery

Week	Form
2-3	Remote live lectures on teams (Tues and Thurs)
4	No lectures
5-11	Hybrid lectures (Teams & Huxley)

Part 1 - Dr Pietzuch

- Overview and Introduction
- Processes and Threads
- Overview and Introduction

Part 2 - Luis

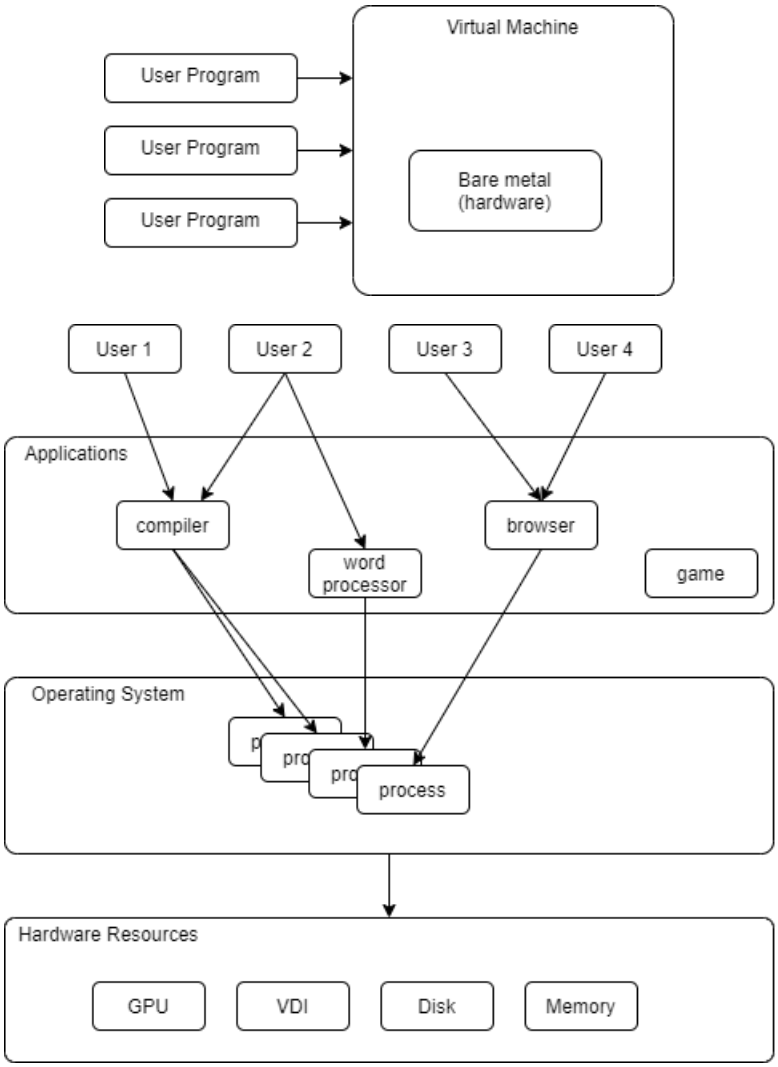
- Memory Management
- Device Management
- Disk Management
- File Systems
- Security
- Virtualisation

Recommended Reading

1. Modern Operating Systems
2. Operating Systems: 3 easy pieces (free online)
3. Understanding the Linux Kernel
4. Inside Windows NT & 2000

Computer Architecture Overview

OS provides a clean interface for different types of applications to run of different hardware.
OS provides resource management and useful abstractions.



OS Characteristics

The OS must be able to:

- Must share Data, programs and hardware (time and space multiplexing)
- Must offer resource allocation (efficient and fair use of memory, CPU etc)
- Must offer simultaneous access to resources, or if this is not possible, enforce mutual exclusion.

- Must protect against accidental or malicious corruption of Data.

Can also support concurrency (logical or actual)

- Support simultaneous activities occurring, e.g multiple users & programs
- Can switch activities at any arbitrary time, and must ensure this does not result in program failures
- Ensure safe concurrency through primitives (e.g semaphores, locks)

The OS must also take into account non-determinism, as it cannot predict what interrupts/events will occur and when. (events occur in an unpredictable order)

Also needs to consider persistent storage (e.g HDD, SSD etc).

- easy access to files through user-defined names.
- Enforces access controls (permissions to read, write, copy, etc).
- Can Protect against failure (e.g backups).
- Manage storage devices, partitions etc.

OS Structure

Monolithic

Whole OS is on executable. File system, drivers etc are built into the kernel and run with privilege.

Microkernel

- Kernel provides some abstraction, manages resources, but many services run as servers (e.g file system, drivers, etc).
- IPC slow, so lower performance than monolithic.
- More reliable.

Hybrid Kernel

Mix of the two above, used by OS's such as windows.

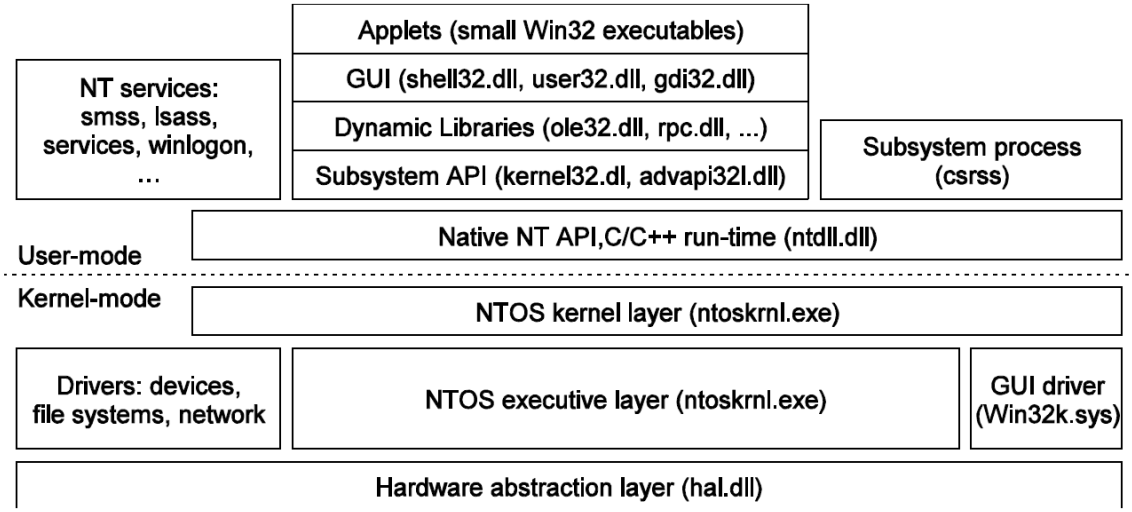
Linux

Linux is a variant of Unix, developed as a monolithic version of the Minix kernel.

- System calls implemented by putting args in registers & on the stack and issuing a trap.
- Supports many programs through GNU (GNU's not unix) project.
- Interrupt handlers are the primary method of interacting with devices.
- Supports dynamically loadable modules.
- exposes many devices and services as files.

Windows

A hybrid kernel, NT replaced MS-DOS and was inspired by VMS.



NTOS is loaded from ntoskrnl.exe (windows NT operating system kernel executable) at boot. It consists of two layers:

1. Executive - Most of the services.
2. Kernel Thread scheduling & synchronisation, traps, interrupt handlers, and CPU management.

There is also a **HAL** (Hardware Abstraction Layer) that abstracts away DMA (direct memory access) operations, BIOS config & CPU types. This in theory makes windows easy to poert to new devices and architectures. However in practice for mainly commercial reasons this has not occurred.