Advanced Operating Systems

Through tracing, analysis, and experimentation

ACS/Part III + Part II: Advanced Operating Systems

Lecture 1, Part 1: What is an Operating System?

Prof. Robert N. M. Watson

2023-2024

MPhil/Part III vs. Part II AdvOpSys

- These lectures are shared by two separate courses:
  - ACS / Part III: Advanced Operating Systems
  - Part II: Advanced Operating Systems
- The two courses also share an online lab framework based on the RPi4, JupyterLabs, DTrace, and HWPMC
  - Lab 1 is a tutorial shared by the two courses
- But there are some important differences:
  - Key difference 1: Assessed coursework (after lab 1)
    - ACS/Part III has **2x** independently written **lab reports**
    - Part II has **2x** short-answer **lab assignments**
  - Key difference 2: Assigned readings
    - ACS/Part III assigns additional research readings
- Please be sure to use the right material for your course!

Getting started

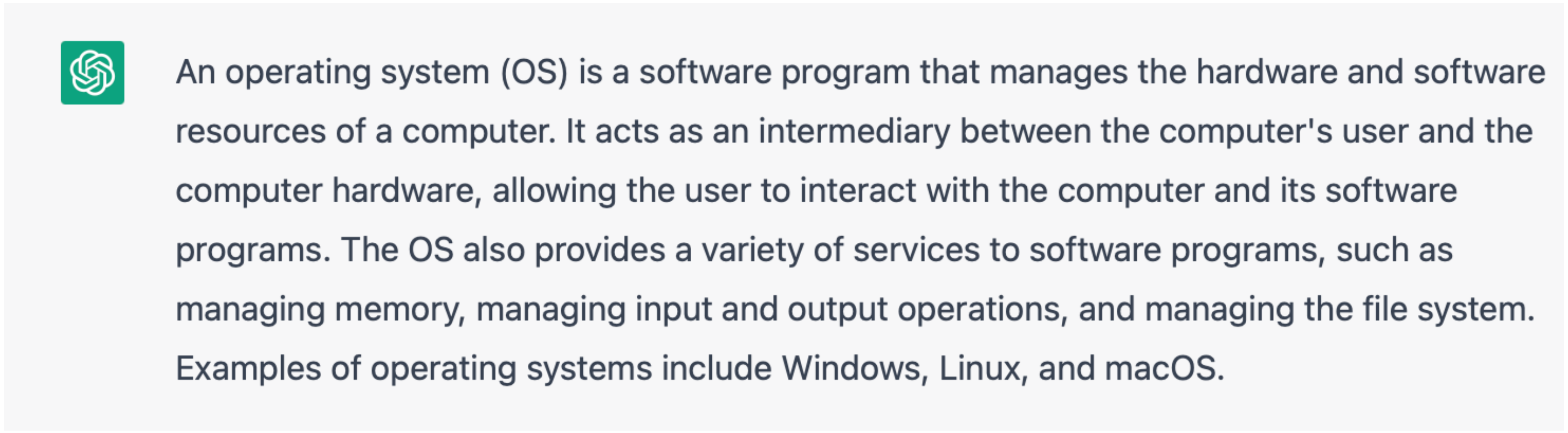
• What is an operating system?	Lecture 1, Part 1
• About the module • Systems research • Lab assignments / reports	Lecture 1, Part 2
• Kernel dynamics • Readings for next lecture	Lecture 1, Part 3

What is an operating system?

[An OS is] low-level software that supports a computer’s basic functions, such as scheduling tasks and controlling peripherals.

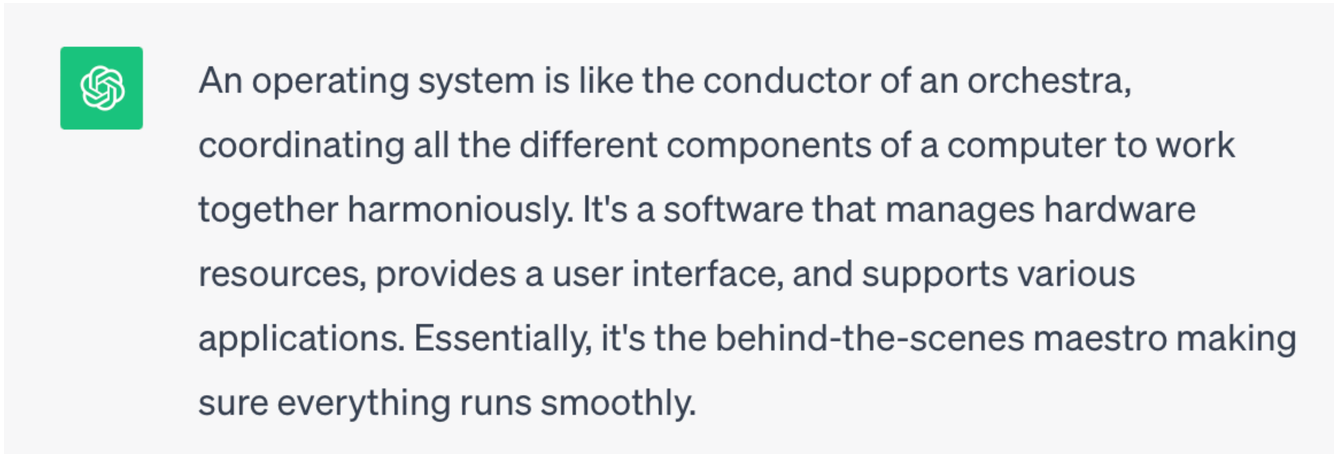
- Google

What is an operating system?


An operating system (OS) is a software program that manages the hardware and software resources of a computer. It acts as an intermediary between the computer's user and the computer hardware, allowing the user to interact with the computer and its software programs. The OS also provides a variety of services to software programs, such as managing memory, managing input and output operations, and managing the file system. Examples of operating systems include Windows, Linux, and macOS.

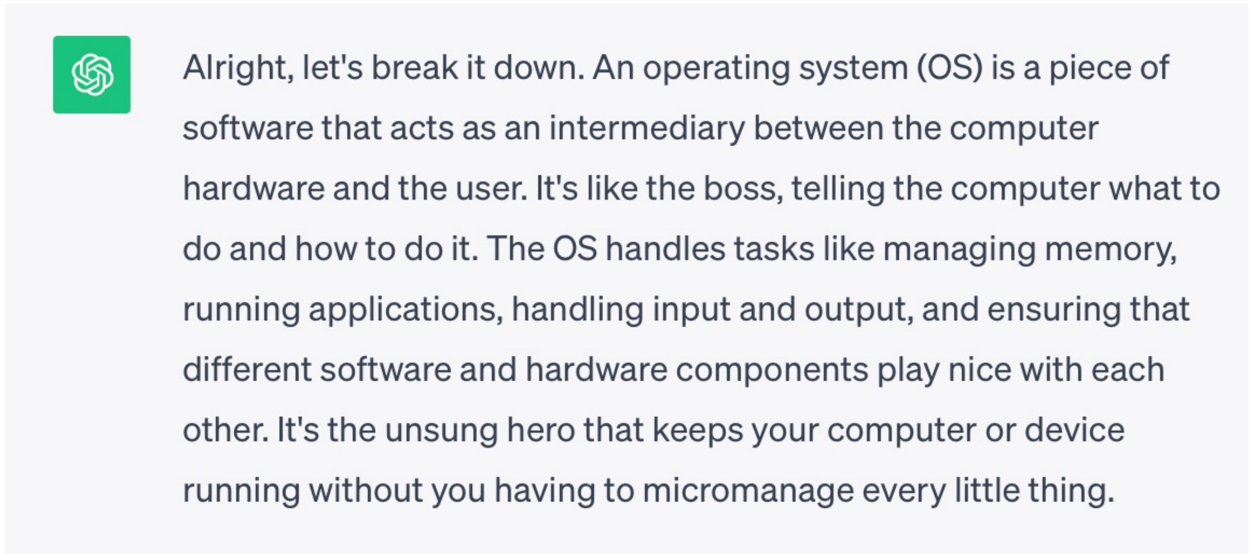
- ChatGPT (2023)

What is an operating system?


An operating system is like the conductor of an orchestra, coordinating all the different components of a computer to work together harmoniously. It's a software that manages hardware resources, provides a user interface, and supports various applications. Essentially, it's the behind-the-scenes maestro making sure everything runs smoothly.

- ChatGPT (2024)

Sure .. But what is an operating system really?


Alright, let's break it down. An operating system (OS) is a piece of software that acts as an intermediary between the computer hardware and the user. It's like the boss, telling the computer what to do and how to do it. The OS handles tasks like managing memory, running applications, handling input and output, and ensuring that different software and hardware components play nice with each other. It's the unsung hero that keeps your computer or device running without you having to micromanage every little thing.

- ChatGPT (2024)

What is an operating system?

But that is basically the 1970s definition,
and not at all a contemporary one.

Today’s general-purpose operating systems consist of GB of binaries and hundreds of millions of LoC.

Further, when you select an operating system,
you select hardware and software ecosystems.

What is an operating system?

Payment services?

Access control?

Threads and processes?

Backup?

Crypto libraries?

Software updates?

Local file systems?

Debug and trace?

Networking and WiFi?

Secure enclaves?

Window system?

User authentication?

Application packaging?

Distributed file-system clients and servers?

Kernel and userspace?

Shell and command-line tools?

Web browser?

Profiling and optimization?

Device drivers?

Remote management?

Class libraries?

Crashdump collection

System libraries?

Virtual machines?

Run-time linker?

Remote access?

.. And surely lots more

Multimedia?

Language runtimes?

General-purpose operating systems

… are for **general-purpose computers:**

- Servers, workstations, mobile devices
- Run **applications** – i.e., software unknown at OS design time
- Abstract the hardware, provide services, 'class libraries'
- E.g., Windows, Apple macOS, Android, iOS, Linux, BSD, …

Userspace	Local and remote shells, GUI, management tools, daemons Run-time linker, system libraries, logging and tracing facilities
– system-call layer –	
Kernel	System calls, hypercalls, remote procedure call (RPC)* Processes, filesystems, IPC, sockets, management Drivers, packets/blocks, protocols, tracing, virtualisation VM, malloc, linker, scheduler, threads, timers, tasks, locks

* Continuing disagreement on whether distributed-filesystem servers and window systems 'belong' in userspace or the kernel

Other kinds of operating systems (1/3)

Specialise the OS for a specific application or environment:

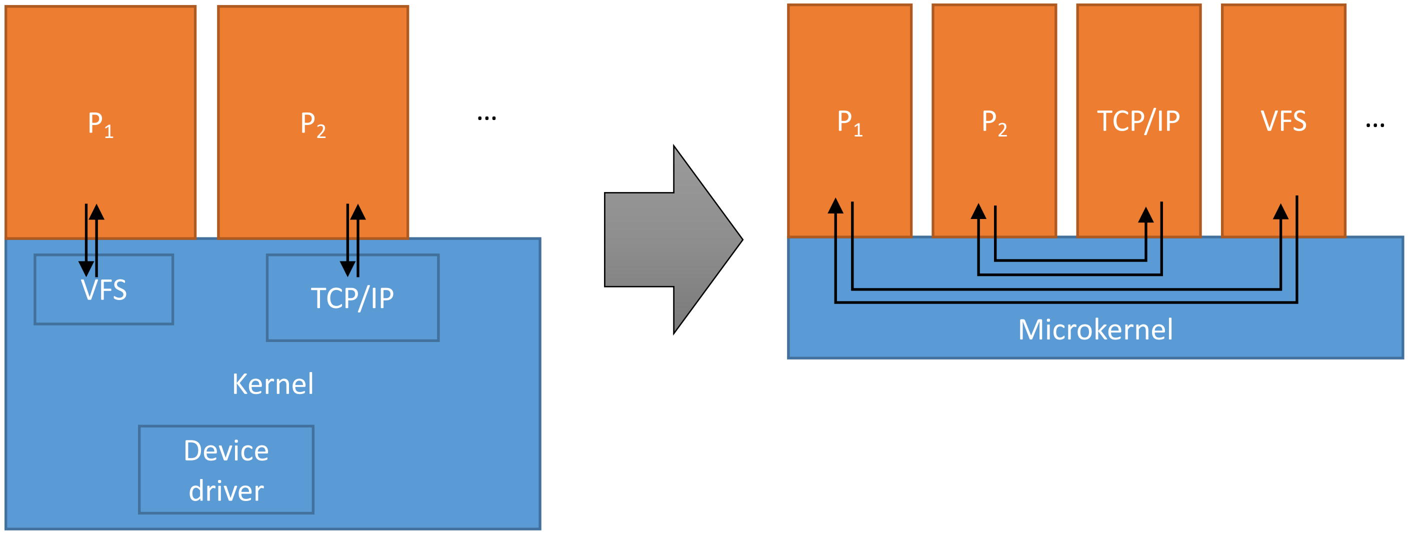
- **Embedded, real-time operating systems**
  - Serve a single application in a specific context
    - E.g., WiFi access points, medical devices, washing machines, cars
  - Small code footprint, real-time scheduling
  - Might have virtual memory / process model
  - Microkernels or single-address space: VxWorks, RTEMS, L4
  - Now also: Linux, BSD (sometimes over a real-time kernel), etc.
- **Appliance operating systems**
  - Apply embedded model to higher-level devices/applications
  - File storage appliances, routers, firewalls, ...
    - E.g., Juniper JunOS, Cisco IOS, NetApp OnTap, EMC/Isilon
  - Under the hood, almost always Linux, BSD, etc.

Key concept: **Operating system as a reusable component**

Other kinds of operating systems? (2/3)

What if we rearrange the boxes?

- **Microkernels, library operating systems, unikernels**
  - Shift code from kernel into userspace to reduce Trusted Computing Base (TCB); improve robustness/flexibility; 'bare-metal' apps
  - Early 1990s: Microkernels are king!
  - Late 1990s: Microkernels are too slow!
    - (But ideas about OS modularity dating from this period are widespread)
  - 2000s/2010s: Microkernels are back! But now 'hypervisors'
  - Sometimes: programming-language runtime as OS



Other kinds of operating systems? (3/3)

- **Hypervisors**
  - Kernels host processes; hypervisors host virtual machines
    - Type-1: Standalone hypervisors (e.g., Xen)
    - Type-2: Integrated with OS kernel (e.g., KVM)
  - Virtualised hardware interface rather than POSIX APIs
  - Paravirtualisation reintroduces OS-like APIs for performance
  - E.g., System/370, VMware, Xen, KVM, VirtualBox, bhyve, Hafnium, ...
  - Many microkernel ideas have found a home here
- **Containers**
  - Hosts multiple userspace instances over a common kernel
  - Controlled namespaces prevent inappropriate accesses
  - Really more about code/ABI (Application Binary Interface) distribution and maintenance

What does an operating system do?

- Key hardware-software surface (w/compiler toolchain)
- Low-level abstractions and services
  - **Operational model**: bootstrap, shutdown, watchdogs
  - **Process model, IPC**: processes, threads, IPC, program model
  - **Resource sharing**: scheduling, multiplexing, virtualisation
  - **I/O**: drivers, local/distributed filesystems, network stack
  - **Security**: authentication, encryption, ACLs, MAC, audit
  - **Local or remote access**: console, window system, SSH
  - **Libraries**: math, protocols, RPC, crypto, UI, multimedia
  - **Monitoring/debugging**: logs, profiling, tracing, debugging

Compiler? Text editor? E-mail package? Web browser?
Can an operating system be “distributed”?