OS and its Three Pieces

Kartik Gopalan

What is an OS?

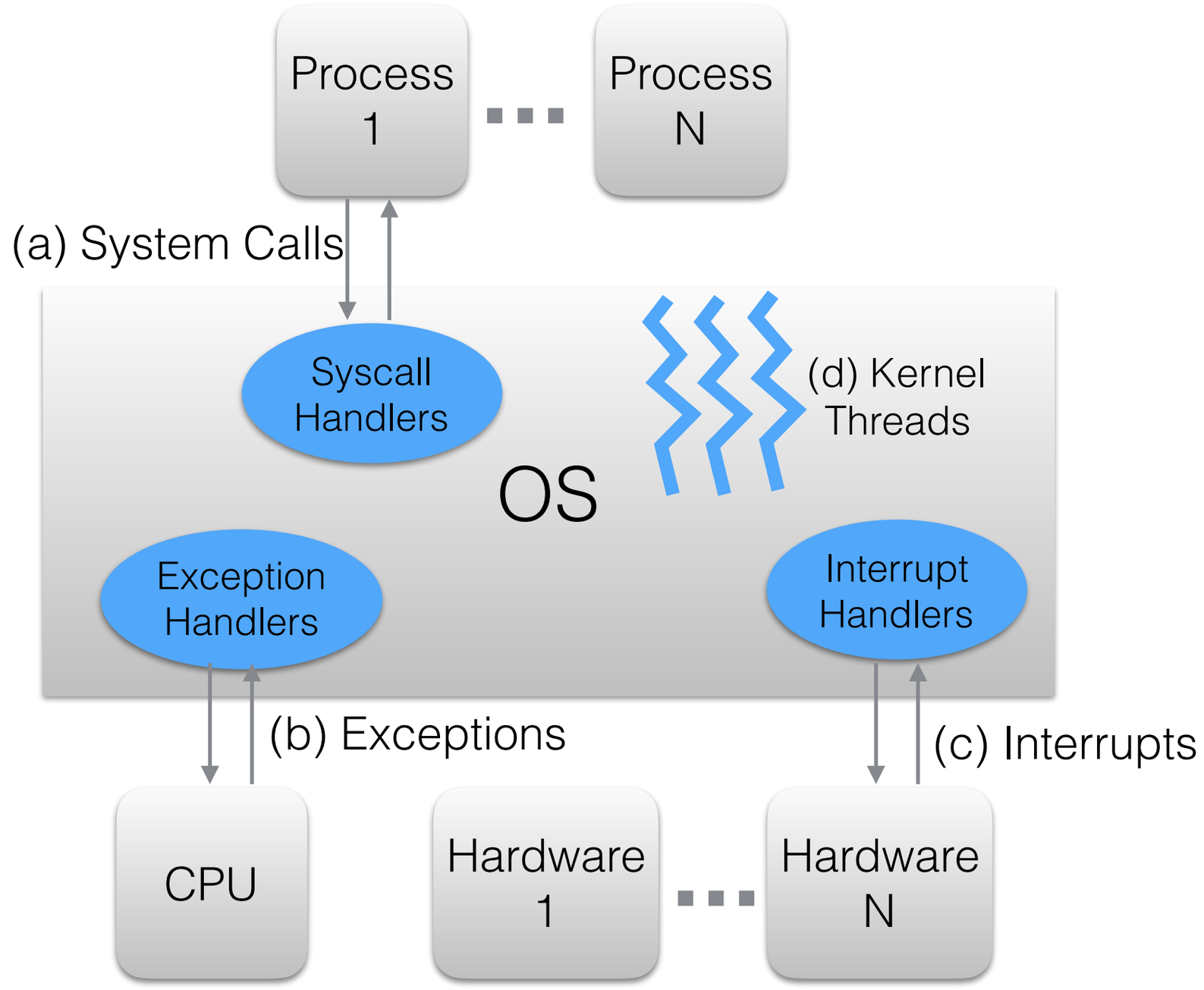
What is an OS?

- A bunch of software and data residing somewhere in memory.
  - But its not just *any* software.
- OS is the *most privileged* software in a computer.
  - *Privileged* means that OS can do special things, like write to disk, talk over the network, control memory and CPU usage, etc.
- OS manages all system resources
  - CPU, Memory, and I/O devices

But when does the OS “run”?

But when does the OS "run"?

Four ways to invoke OS code



Three major tasks of OS

1. Virtualization
2. Concurrency
3. Persistence

Virtualization

- Making a physical resource look like something else (virtual).
- Why virtualize?
  - To make the computer easier to use and program.
- Examples
  - Make one physical CPU look like multiple virtual CPUs
    - One or more virtual CPUs per process
  - Make physical memory (RAM) and look like multiple virtual memory spaces
    - One or more virtual memory spaces per process
  - Make physical disk look like a file system
    - Physical disk = raw bytes.
    - File system = user's view of data on disk.

Concurrency

- Juggling many tasks together
- Examples
  - One physical CPU runs many processes
  - One process runs many threads
  - One OS juggles process execution, system calls, interrupts, exceptions, CPU scheduling, memory management, etc.
- There's a LOT of concurrency in modern computer systems.
- And its the source of most of the system complexity.

Persistence

- Storing data “forever”
  - On hard disks, SSDs, CDs, floppy disks, tapes, phono discs, paper!
- But its not enough to just store raw bytes
- Users want to
  - Organize data (via file systems)
  - Share data (via network or cloud)
  - Access data easily
    - …and recover data when lost.
  - Protect data from being stolen.

History of OS

- 1950s and 1960s: Early operating systems were simple batch processing systems
  - Users provided their own “OS” as libraries.
- 1960s and1970s: Multi-programming on mainframes
  - Concurrency, memory protection, Kernel mode, system calls, hardware privilege levels, trap handling
  - Earliest Multics hardware and OS on IBM mainframes
  - Which led to the first UNIX OS which pioneered file systems, shell, pipes, and the C language.
- 1980s: Personal computing era
  - MacOS, IBM PC and its DOS, Windows, and so forth.
  - Unfortunately, many lessons from earlier multi-programming era were forgotten and had to be re-learned (painfully).
- 1980s also saw the fragmentation of UNIX
  - Each big company had its own version (IBM, Apple, HP, SUN, SGI, NCR, AT&T….)
  - LOT of legal wrangling over IP and copyrights
- 1990s: Then came BSD and Linux
  - Open source.
  - Led the way to modern OSes and cloud platforms
- 1990s also saw wider adoption of threads and parallelism
- 2000 and beyond: Mobile device OS and hypervisors
  - Android, iOS
  - VMWare ESX, Xen, Linux/KVM etc.