What is an operating system?

1. A virtual machine!
   - offering a more comfortable and safer environment
   (e.g. memory management and protection, hardware abstraction, process management, inter-process communication, ...)

2. A resource manager!
   - coordinating access to hardware resources
   - coordinating access to hardware resources

Operating systems deal with
- processors
- memory
- mass storage
- communication channels
- devices (input, special purpose processors, peripheral hardware, ...)
- tasks (processes/programs which are applying for access to these resources)

The evolution of operating systems
- in the beginning: single user, single program, single task, serial processing — no OS
- 50s: System monitors / batch processing
  - the monitor ordered the sequence of jobs and triggered their sequential execution
- 50s-60s: Advanced system monitors / batch processing
  - the monitor is handling interrupts and timers
  - first support for memory protection
  - first implementations of privileged instructions (accessible by the monitor only)
- early 60s: Multiprogramming systems
  - employ the long device I/O delays for switches to other, runnable programs
- early 60s: Multiprogramming, time-sharing systems
  - assign time-slices to each program and switch regularly
- early 70s: Multitasking systems — multiple developments resulting in UNIX (besides others)
- early 80s: Single-user, single-tasking systems, with emphasis on user interface or APIs
- mid-80s: Distributed/multiprocess operating systems — modern UNIX systems (VMS, BSD)

The types of current operating systems
- Personal computing systems, workstations, and workgroup servers:
  - late 70s: Workstations starting by porting UNIX or VMS to "smaller" computers
  - 80s: PCs starting with almost none of the classical OS-features and services
  - last 20 years: evolving and expanding into current general purpose OSs, like for instance
    - Solaris (based on VMS, BSD, and SunOS)
    - LINUX (open source UNIX re-implementation for x86 processors and others)
    - current Windows (used to be partly based on Windows NT, which is related to VMS)
    - MacOS (Mac kernel with BSD Unix and a proprietary user interface)
  - Multiprocessing is supported by all these OSs to some extent.
  - None of these OSs are suitable for distributed or real-time systems.
  - None of these OSs are suitable for embedded systems, although trials have been performed.

References for this chapter
  - Chapter 4 "The Processor"
  - Chapter 6 "Parallel Processors from Client to Cloud"
  - ACM edition, Morgan Kaufmann 2017

What is an operating system?
Operating Systems

Types of current operating systems

Parallel operating systems
- support for a large number of processors, either:
  - symmetrical: each CPU has a full copy of the operating system
  - asymmetrical: only one CPU carries the full operating system; the others are operated by small operating system stubs to transfer code or tasks.

Distributed operating systems
- all CPUs carry a small kernel operating system for communication services.
- all other OS services are distributed over available CPUs.
- services can be multiplied in order to guarantee availability (fail-safe) or to increase throughput (heavy duty servers).

Operating Systems

Types of current operating systems

Real-time operating systems need to provide...
- the logical correctness of the results as well as
- the correctness of the time, when the results are delivered.

Predictability?
(not performance!)
- All results are to be delivered just-in-time - not too early, not too late.

Real-time operating systems
- Fast context switches?
- Small size?
- Quick response to external interrupts?
- Multi threading?
- Low-level programming interface?
- Inter process communication tools?
- High processor utilization?

Operating Systems

Types of current operating systems

Real-time operating systems
- usually real-time systems, often hard real-time systems.
- very small footprint (0.5 to 25 kByte).
- more limited user interaction.
- 90% to 95% of all processors are working here.

Operating Systems

Types of current operating systems

Embedded operating systems
- usually real-time systems, often hard real-time systems.
- very small footprint (0.5 to 25 kByte).
- more limited user interaction.
- 90% to 95% of all processors are working here.
What is an operating system?

Is there a standard set of features for operating systems?

the term operating system covers 4 kB microkernels, as well as > 1 GB installations of desktop general purpose operating systems.

Is there a minimal set of features?

memory management, process management and inter-process communication/synchronization will be considered essential in most systems.

Is there a minimal set of features?

Is there always an explicit operating system?

some languages and development systems operate with standalone runtime environments.

Typical features of operating systems

- Allocation / Deallocation
- Virtual memory logical vs. physical addresses, segments, paging, swapping etc.
- Memory protection (granule level, separate virtual memory segments, ...)
- Shared memory

Synchronisation / Inter-process communication

- semaphores, mutual exclusion, condition variables, channels, mailboxes, MPI, etc. (chapter 4)
- tightly coupled to scheduling / task switching!

Hardware abstraction

- Device drivers
- API
- Protocols, file systems, networking, everything else...

Monolithic

or ‘the big mess...’

- non-portable
- hard to maintain
- lacks reliability
- all services are in the kernel (on the same privilege level)
- but may reach high efficiency

E.g. most early UNIX systems, MS-DOS (80s), Windows (all non-NT based versions), MacOS (until version 9), and many others...

Monolithic & Modular

- Modules can be platform independent
- Easier to maintain and to develop
- Reliability is increased
- all services are still in the kernel (on the same privilege level)
- may reach high efficiency

E.g. current Linux versions
Typical structures of operating systems

µKernels & client-server models

- µkernel implements essential process, memory, and message handling
- all higher services are dealt with outside the kernel
- significantly easier to maintain
- µkernel is highly hardware dependent
- servers can be redundant and easily replaced
- possibly reduced efficiency through increased communications

µKernels & virtual machines

- µkernel implements essential process, memory, and message handling
- all higher services are user level servers
- significantly easier to maintain
- kernel ensures reliable message passing between clients and servers
- highly modular and flexible
- servers can be redundant and easily replaced
- possibly reduced efficiency through increased communications

Typical structures of operating systems

µkernel, client-server structure

e.g. current research projects, LiSE, etc.

µkernel, distributed real-time operating systems, current distributed O/S research projects

UNIX

- Hierarchical file-system (maintained via 'mount' and 'rmount')
- Universal file-interface applied to files, devices (I/O), as well as IPC
- Dynamic process creation via duplication
- Choice of shells
- Internal structure as well as all APIs are based on 'C'
- Relatively high degree of portability

1 CPU per control-flow

Specific configurations only, e.g.
- Distributed controllers
- Physical process control systems
- 1 CPU per task, connected via a bus-system
- Process management (scheduling not required)
- Shared memory access need to be coordinated

Introduction to processes and threads

Processes

- OS: emulate one CPU for every control-flow
- Address space + Control-flows
- Kernel has full knowledge about all processes, as well as their states, requirements, and currently held resources
- Inside the OS
- Kernel scheduling
- Threads can easily be connected to external events (I/O)
- Outside the OS
- User-level scheduling
- Threads may need to go through their parent process to access I/O

Threads

- Individual control-flows can be handled
- Support for memory protection essential
- Process management (scheduling) required
- Shared memory access need to be coordinated

CPU
### Operating Systems

#### Symmetric Multiprocessing (SMP)

All CPUs share the same physical address space (and access to resources).

- Any process/thread can be executed on any available CPU.

---

#### Process Control Blocks (PCBs)

- Process id: created, ready, executing, blocked, suspended, bored ...
- Scheduling attributes: Priorities, deadlines, consumed CPU-time, ...
- CPU state: Saved extended information while context switches (incl. the program counter, stack pointer, ...)
- Memory attributes/privileges: Memory base, limits, shared areas, ...
- Allocated resources/privileges: Memory base, limited, shared areas, ...
- PCBs (linked thereof) are commonly enqueued at a certain state or condition (awaiting access or change in state).

---

#### Process States

- Created: the task is ready to run, but not yet considered by any dispatcher waiting for admission
- Ready: ready to run or waiting for a free CPU
- Running: holds a CPU and executes
- Blocked: not ready to run or waiting for a resource
- Suspended: swapped out of main memory (non-time critical processes) or waiting for main memory space (and other resources)

---

#### Performance Scheduling

- Requested resource times

---

#### Definition of terms

- Short-term: ready, dispatching
- Medium-term: ready, dispatching
- Long-term: ready, dispatching
**Performance Scheduling**

**First come, first served (FCFS)**

- **Waiting time**: 0.11, average: 5.9 – **Turnaround time**: 3.12, average: 8.4

As tasks apply concurrently for resources, the actual sequence of arrival is non-deterministic; hence even a deterministic scheduling schema like FCFS can lead to different outcomes.

**Performance Scheduling**

**Round Robin (RR)**

- **Waiting time**: 0.5, average: 1.2 – **Turnaround time**: 1.20, average: 5.8

Optimized for swift initial responses.

“Stretches out” long tasks.

Bound maximal waiting time!

In this example:

- The average waiting times vary between 5.4 and 5.9.
- The average turnaround times vary between 8.0 and 8.4.
- Shortest possible maximal turnaround time!

---

**Summary**

- **Operating Systems**
  - Concept
  - Categories
  - Architectures

- **Processes**
  - Definition
  - Relation to architectures
  - Scheduling