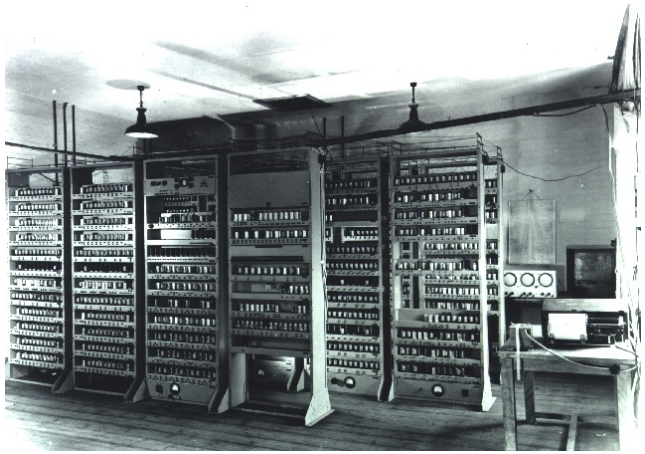
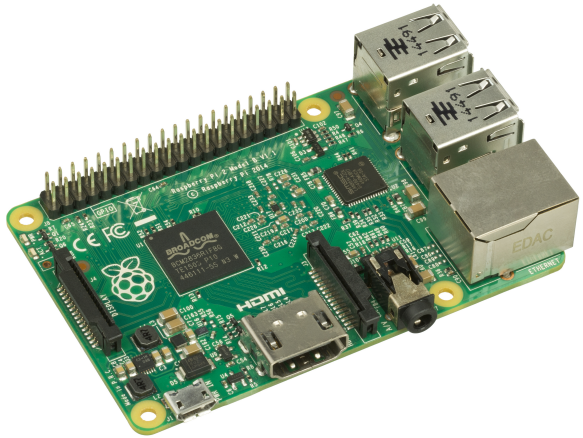
EDSAC (circa 1949)



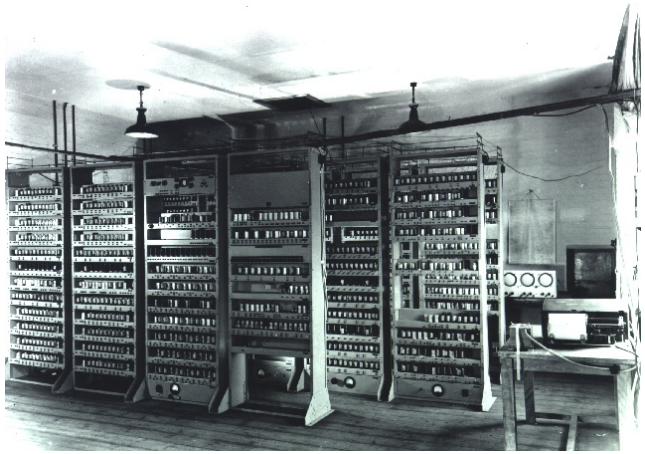
RaspberryPi2 (circa 2015)



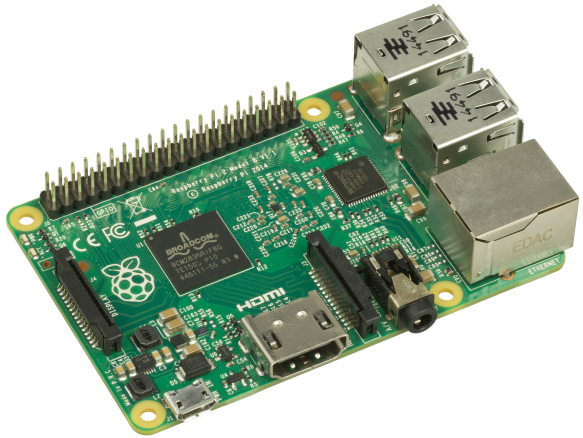
http://www.cl.cam.ac.uk/Relics/jpegs/edsac99-9.jpg

http://en.wikipedia.org/wiki/File:Raspberry-Pi-2-Bare-BR.jpg

EDSAC (circa 1949)



RaspberryPi2 (circa 2015)



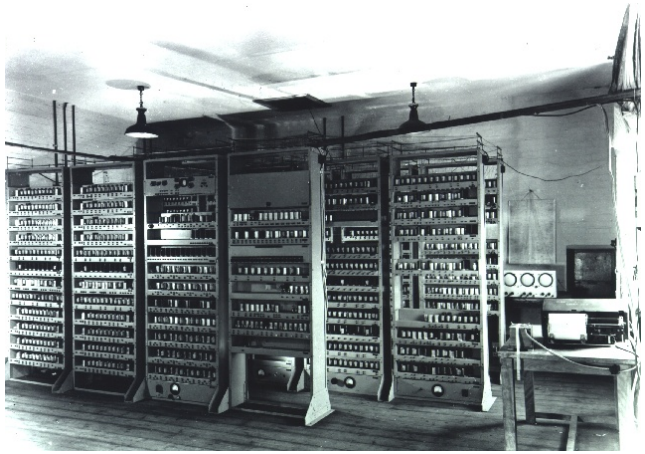
- Question: what's changed?

---

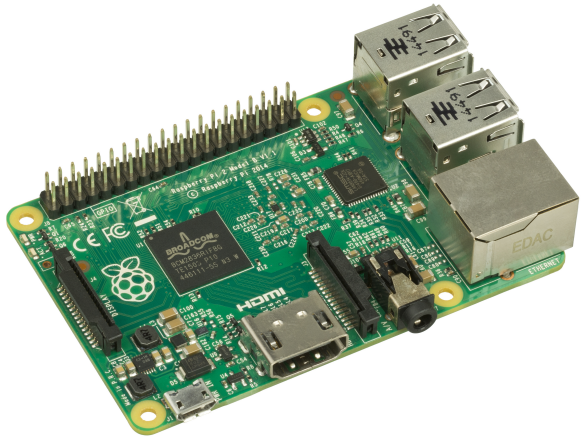
http://www.cl.cam.ac.uk/Relics/jpegs/edsac99-9.jpg

http://en.wikipedia.org/wiki/File:Raspberry-Pi-2-Bare-BR.jpg

EDSAC (circa 1949)



RaspberryPi2 (circa 2015)



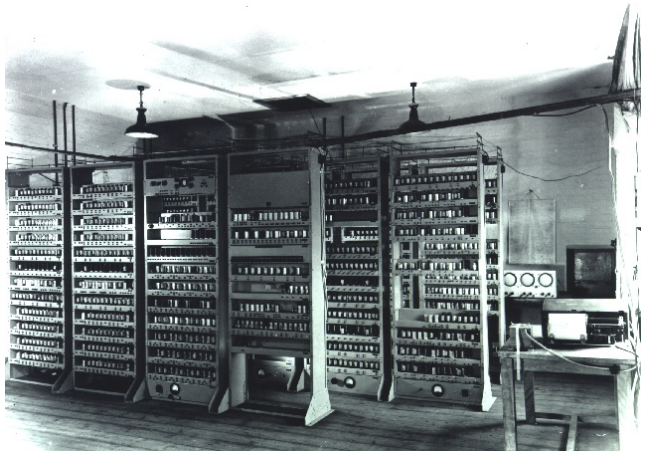
- Question: what's changed?
- Answer:
  1. *technology* trends:
     - Moore's Law, Joy's Law, Wirth's Law, Koomey's Law, Nielsen's Law, Metcalfe's Law,
     - volume and diversity of (peripheral) devices,
     - ...
  2. *societal* trends:
     - use-cases, e.g., mobile vs. not, or interactive vs. not,
     - volume and diversity of data,
     - users-per-computer, users-per-resource (and so on) ratios,
     - ...

---

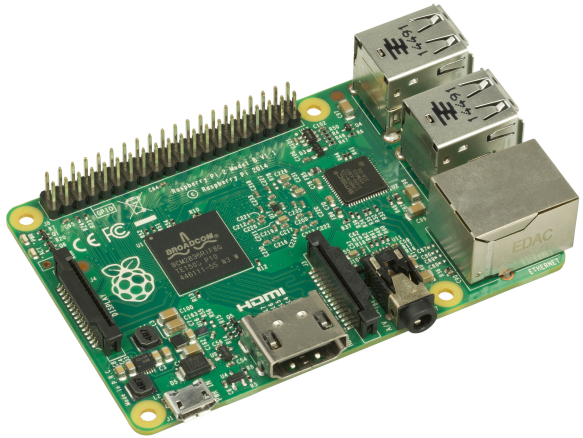
http://www.cl.cam.ac.uk/Relics/jpegs/edsac99-9.jpg

http://en.wikipedia.org/wiki/File:Raspberry-Pi-2-Bare-BR.jpg

EDSAC (circa 1949)



RaspberryPi2 (circa 2015)



- Question: what's changed?
- Answer: *massively* increased **complexity**
  - complex use-cases,
  - complex quality metrics and requirements,
  - complex software,
  - complex hardware,
  - complex interactions and failure modes,
  - ...

which form motivation for utilising an **operating system**.

---

http://www.cl.cam.ac.uk/Relics/jpegs/edsac99-9.jpg

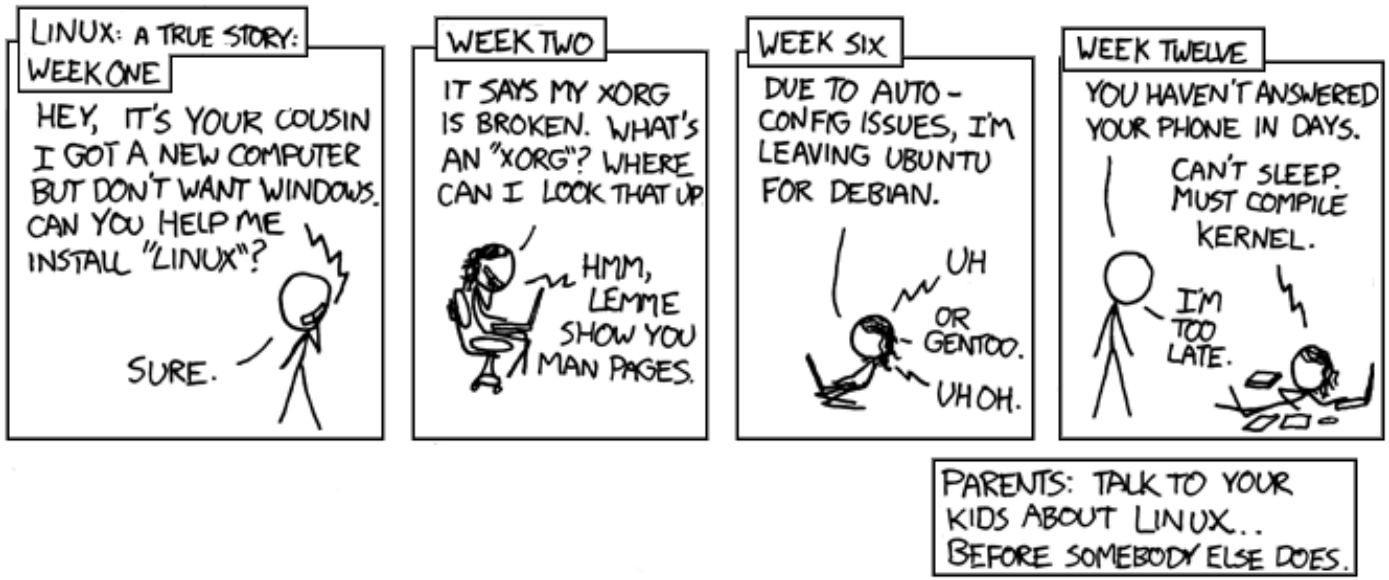
http://en.wikipedia.org/wiki/File:Raspberry-Pi-2-Bare-BR.jpg

Concepts (1)
What *is* an operating system?

- Question: what *is* an operating system?

Concepts (1)
What *is* an operating system?

- Question: what *is* an operating system?
- Answer: maybe you define it via *experience*



but that's not *so* useful.

---

http://xkcd.com/456/

- Question: what *is* an operating system?
- Answer: a more technical definition might be

Definition

> **operating system**, *n.* *the low-level software that supports a computer's basic functions, such as scheduling tasks, controlling peripherals, and allocating storage.*
>
> – *OED (http://www.oed.com)*

but, in practice, we often find that

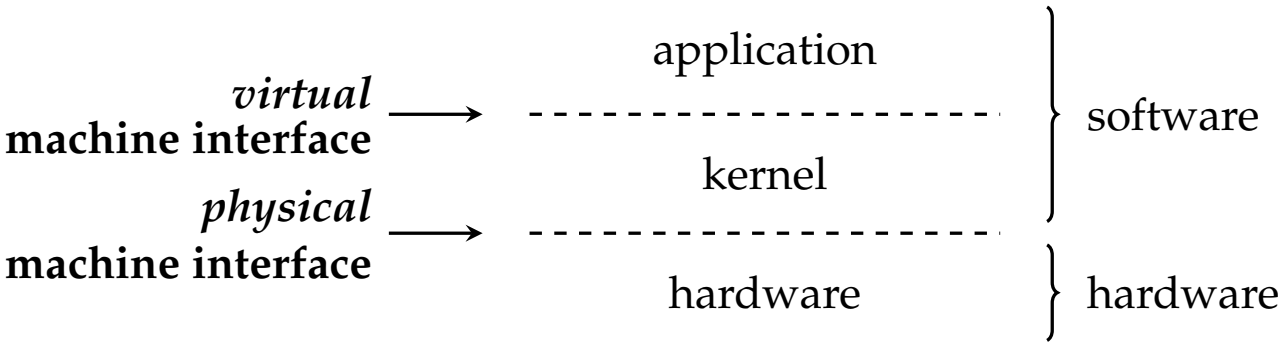
$$\text{operating system } \textit{distribution} = \{\text{kernel}, \text{system software}, \text{application software}, \ldots\}$$

so, from here on, we'll make the strict assumption that

$$\textbf{operating system} \equiv \textbf{kernel}.$$

Concepts (1)
What *is* an operating system?

- Question: what *is* ~~an operating system~~ a kernel?
- Answer: a more technical definition might be



so the kernel is a layer of software that delivers

1. **management** : allocate, multiplex, and protect access to resource
2. **abstraction** : offer appropriate interface to resource
3. **virtualisation** : make it look like resource has features you want

plus various standard services.

- Question: what is an abstraction?

- Question: what is an abstraction?
- Answer: a method of managing (e.g., hiding) complexity.

- Question: what is an abstraction?
- Answer: a method of managing (e.g., hiding) complexity.
- Question: what constitutes a *good* abstraction?

- Question: what is an abstraction?
- Answer: a method of managing (e.g., hiding) complexity.
- Question: what constitutes a *good* abstraction?
- Answer: one that affords

1. simplification by
   - hiding unattractive properties,
   - adding new functionality, and/or
   - organising information
2. an separation [8] between (or decoupling of)

$$\begin{aligned} \text{policy} &\equiv \text{how the interface works abstractly (i.e., } \textit{semantics}) \\ \text{mechanism} &\equiv \text{how the interface works concretely (i.e., } \textit{implementation}) \end{aligned}$$

Definition

An **address space** is abstraction of memory.

Definition

An **address space** is abstraction of memory.

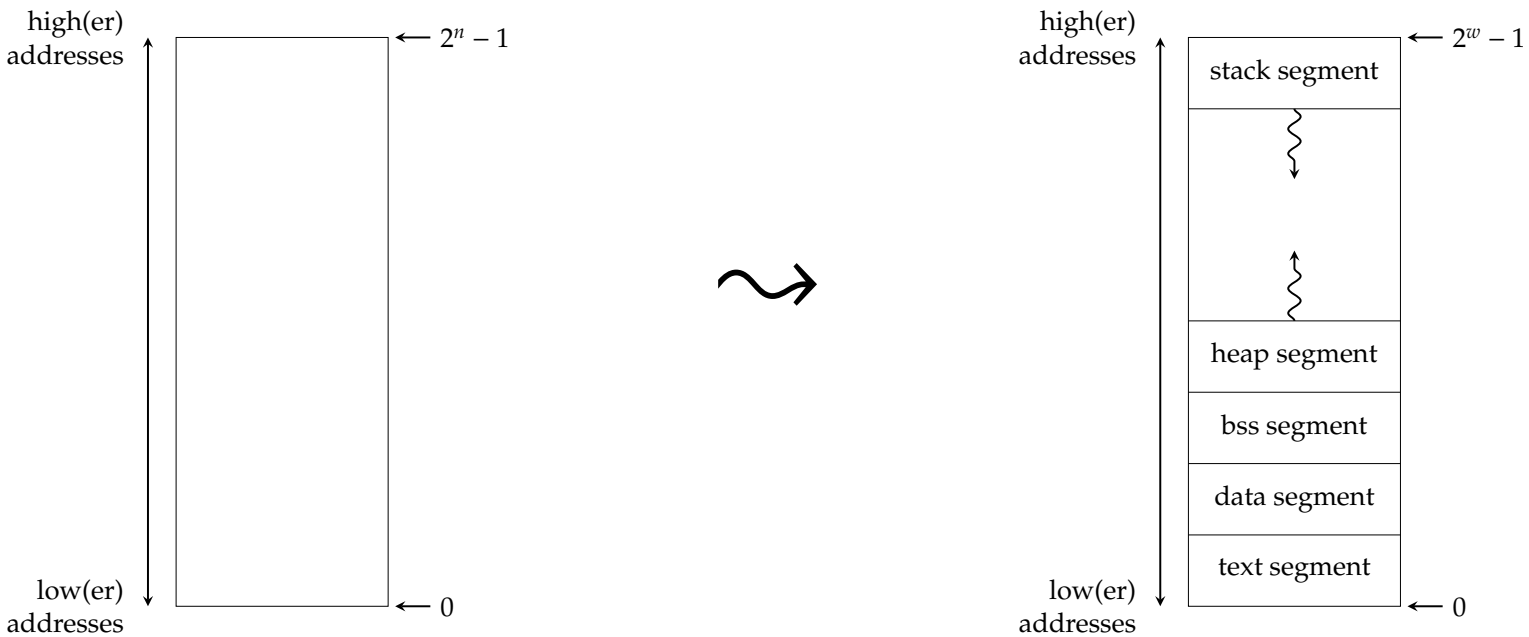
- Question:
  - unattractive properties : **?**
  - new capabilities : **?**
  - organise information : **?**

Definition

An **address space** is abstraction of memory.

- Question:
  - unattractive properties : 1 memory vs. n processes, fixed size, sparse, ...
  - new capabilities : virtualisation, protection, ...
  - organise information : ...

  st. it represents a (structured) set of accessible addresses, e.g.,



Definition

A **process** is abstraction of the processor.

Definition

A **process** is abstraction of the processor.

- Question:
  - unattractive properties : **?**
  - new capabilities : **?**
  - organise information : **?**

Definition

A **process** is abstraction of the processor.

- Question:
  - unattractive properties : 1 processor vs. n processes, ...
  - new capabilities : virtualisation, protection, communication, ...
  - organise information : execution context(s), address space, resources, ...

  st. it represents an executing instance of some program, acting as

  1. a "container" to organise other abstractions, and
  2. a boundary for privilege or protection, which demands hardware support:
     - in the simplest case, two **processor modes** exist
       a. **kernel mode** (i.e., a privileged mode), *or*
       b. **user mode** (i.e., a non-privileged mode),
     - the terms
       a. **kernel space**, and
       b. **user space**

       describe the associated set of accessible resources, and
     - switching *between* modes is carefully controlled.

Definition

A **file** is abstraction of the disk.

Definition

A **file** is abstraction of the disk.

- Question:
  - unattractive properties : **?**
  - new capabilities : **?**
  - organise information : **?**

Definition

A **file** is abstraction of the disk.

- Question:
  - unattractive properties : reliability, latency, fragmentation, ...
  - new capabilities : identifiers, hierarchy, dynamic size, ...
  - organise information : access control, ...

  or, actually, ... only *sort of*:
  - UNIX uses what is often termed an "everything is a file" philosophy [1],
  - *any* stream of bytes has a file-like interface, e.g.,
    - persistent storage,
    - pseudo-files (e.g., `/dev/random`),
    - I/O with devices (e.g., `/dev/sda`),
    - kernel configuration (e.g., `/proc`),
    - ...

Definition

A **socket** is abstraction of the network.

Definition

A **socket** is abstraction of the network.

- Question:
  - unattractive properties : **?**
  - new capabilities : **?**
  - organise information : **?**

Definition

A **socket** is abstraction of the network.

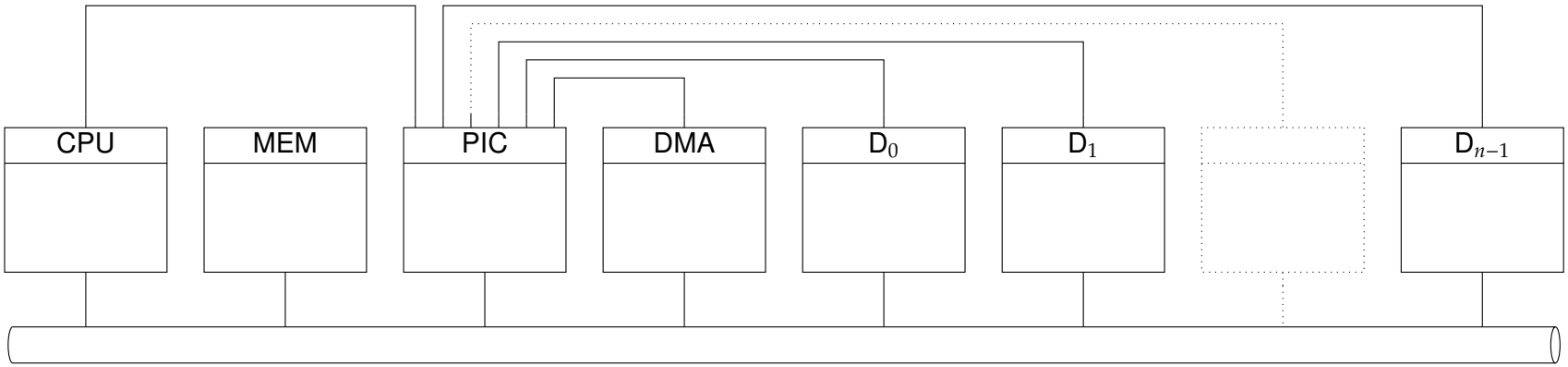
- Question:
  - unattractive properties : reliability, latency, topology, ...
  - new capabilities : name look-up, packet filtering, ...
  - organise information : ...

  or, actually, ... only *sort of*:
  - in reality, there are several *types* of socket
    - domain sockets $\rightsquigarrow$ local communication
    - internet domain sockets $\rightsquigarrow$ remote communication

    which are more like a generalisation of file abstraction,
  - abstraction of the network exists via *multiple* interfaces
    - OSI layers 4 + 5 $\rightsquigarrow$ TCP $\Rightarrow$ sockets
    - OSI layer 3 $\rightsquigarrow$ IP $\Rightarrow$ kernel configuration
    - OSI layer 2 $\rightsquigarrow$ NIC $\Rightarrow$ kernel network interface
    - OSI layer 1 $\rightsquigarrow$ NIC $\Rightarrow$ kernel device drivers

Conclusions

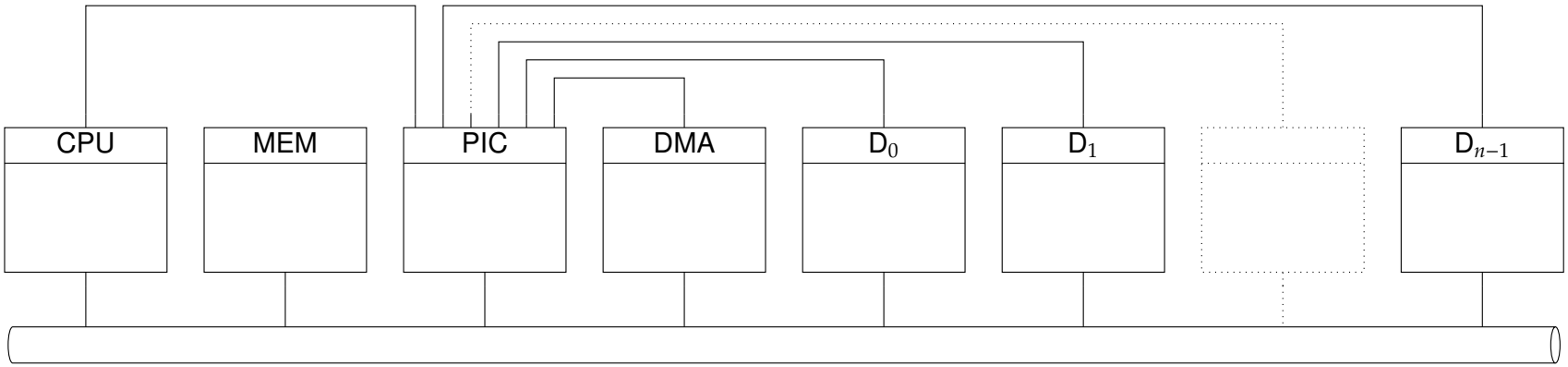
- Remit:
  - understand a simple(ish) computer system



  and how an operating system kernel supports processes executing on it, *but*
  - limit the detail and volume of coverage to fit allocated time.

Conclusions

- Remit:
  - understand a simple(ish) computer system



  and how an operating system kernel supports processes executing on it, *but*
  - limit the detail and volume of coverage to fit allocated time.
- *Why*?!
  1. technical curiosity:
     - to explain how things work,
     - to extract *general* principles from experience.
  2. practical utility:
     - *some* of you will develop an operating system,
     - *some* of you will work in system administration,
     - *most* of you will develop hardware or software that *depends* on an operating system.

Additional Reading

- *Wikipedia: Operating system*. URL: http://en.wikipedia.org/wiki/Operating_system.
- *Wikipedia: Kernel*. URL: http://en.wikipedia.org/wiki/Kernel_(operating_system).
- A. Silberschatz, P.B. Galvin, and G. Gagne. "Chapter 1: Introduction". In: *Operating System Concepts*. 9th ed. Wiley, 2014.
- A. Silberschatz, P.B. Galvin, and G. Gagne. "Chapter 2: System structures". In: *Operating System Concepts*. 9th ed. Wiley, 2014.
- A.S. Tanenbaum and H. Bos. "Chapter 1.1: What is an operating system". In: *Modern Operating Systems*. 4th ed. Pearson, 2015.
- A.S. Tanenbaum and H. Bos. "Chapter 1.5: Operating system concepts". In: *Modern Operating Systems*. 4th ed. Pearson, 2015.

References

[1] *Wikipedia: Everything is a file*. URL: http://en.wikipedia.org/wiki/Everything_is_a_file (see pp. 19–21).

[2] *Wikipedia: Kernel*. URL: http://en.wikipedia.org/wiki/Kernel_(operating_system) (see p. 27).

[3] *Wikipedia: Operating system*. URL: http://en.wikipedia.org/wiki/Operating_system (see p. 27).

[4] A. Silberschatz, P.B. Galvin, and G. Gagne. "Chapter 1: Introduction". In: *Operating System Concepts*. 9th ed. Wiley, 2014 (see p. 27).

[5] A. Silberschatz, P.B. Galvin, and G. Gagne. "Chapter 2: System structures". In: *Operating System Concepts*. 9th ed. Wiley, 2014 (see p. 27).

[6] A.S. Tanenbaum and H. Bos. "Chapter 1.1: What is an operating system". In: *Modern Operating Systems*. 4th ed. Pearson, 2015 (see p. 27).

[7] A.S. Tanenbaum and H. Bos. "Chapter 1.5: Operating system concepts". In: *Modern Operating Systems*. 4th ed. Pearson, 2015 (see p. 27).

[8] W. Wulf et al. "HYDRA: The Kernel of a Multiprocessor Operating System". In: *Communications of the ACM (CACM)* 17.6 (1974), pp. 337–345 (see pp. 9–12).