Computer System (More)

ARQCP Course

Arquitetura de Computadores
Licenciatura em Engenharia Informática

2025/26

Paulo Baltarejo Sousa
pbs@isep.ipp.pt

ISEP INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO

Disclaimer

Material and Slides

Some of the material/slides are adapted from various:

- Presentations found on the internet;
- Books;
- Web sites;
- ...

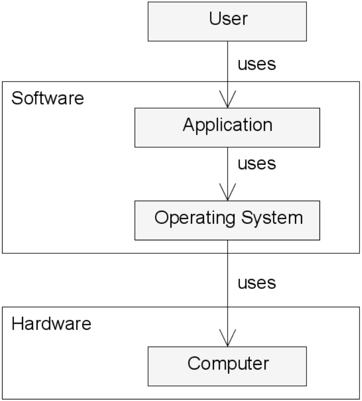
Outline

1. Application Execution

1 **Application Execution**

Operating System (OS) [1] (I)

- When the shell loaded and ran the `hello` program, and when the `hello` program printed its message, neither program accessed the keyboard, display, disk, or main memory directly.
  - Rather, they relied on the services provided by the OS.
- The OS has two primary purposes:
  - To protect the hardware from misuse by runaway applications
  - To provide applications with simple and uniform mechanisms for manipulating complicated and often wildly different low-level hardware devices.



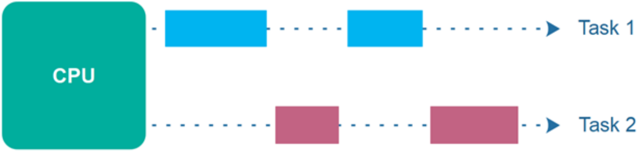
[1] OS issues are out of course scope

OS (II)

- A process is the OS's **abstraction for a running program**.
  - Whenever a program file (a file containing machine code) is loaded to be executed the OS creates a process.
- A process **is an active program and related resources**.
  - From the OS's point of view, the purpose of a **process is to act as an entity to which system resources (CPU time, memory, etc.) are allocated**.
- It provides two virtualisations, **giving the illusion that it alone monopolizes the system**.
  - **Virtualised processor**.
  - **Virtualised memory**.

Virtualised processor

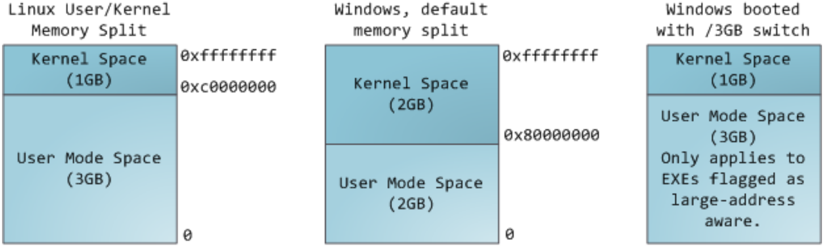
- Multiple processes **can run concurrently on the same system, and each process appears to have exclusive use of the hardware**, such as CPU.



  - The instructions of **one process are interleaved with the instructions of another process**.
  - Generally, **there are more processes to run than there are CPUs to run them**.
    - The OS performs this interleaving with a mechanism known as **context switching**.

Virtualised Memory

- Each process **runs in its own memory sandbox (virtual memory)**.
- Virtual memory is **an abstraction that provides each process with the illusion that it has exclusive use of the main memory**.



Linux User/Kernel Memory Split

Kernel Space (1GB)

User Mode Space (3GB)

0xffffffff

0xc0000000

0

Windows, default memory split

Kernel Space (2GB)

User Mode Space (2GB)

0xffffffff

0x80000000

0

Windows booted with /3GB switch

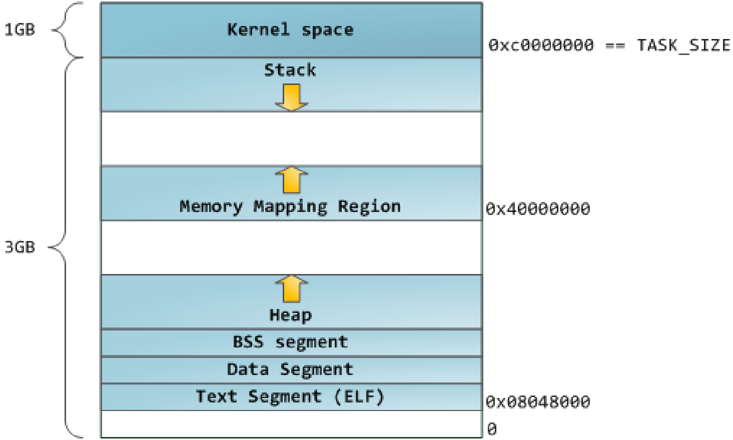
Kernel Space (1GB)

User Mode Space (3GB) Only applies to EXEs flagged as large-address aware.

- **OS Kernel code and data are always addressable**, ready to handle interrupts or system calls at any time.

Virtual Memory Layout

- Each process has the same uniform view of memory, which is known as its **virtual address space**.



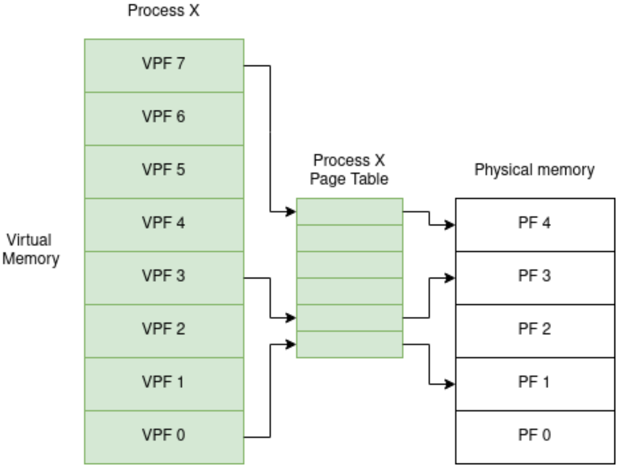
Memory Paging

- Paging is a memory management technique [2] in which the memory is divided into fixed size chunk of bytes, called **pages**.
- **Paging** is a memory management scheme:
    - That eliminates the need for a contiguous allocation of physical memory.
    - That improves the efficiency by moving pages in and out of memory as needed.

[2] There are other techniques such as segmentation.

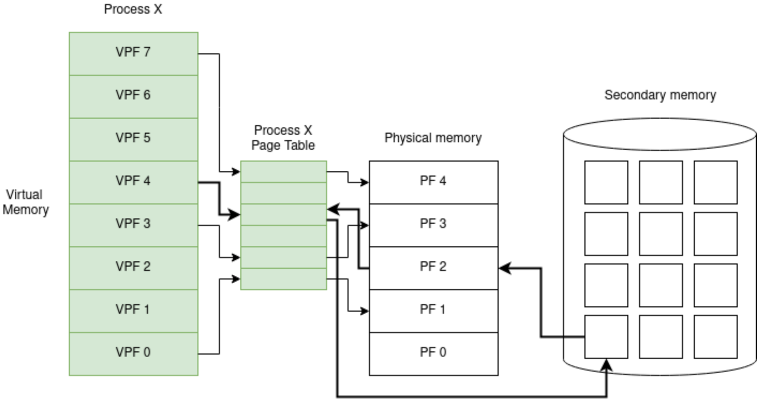
Memory Mapping (I)

- These **virtual addresses are mapped to physical memory by page tables**, which are maintained by the OS and consulted by the CPU.



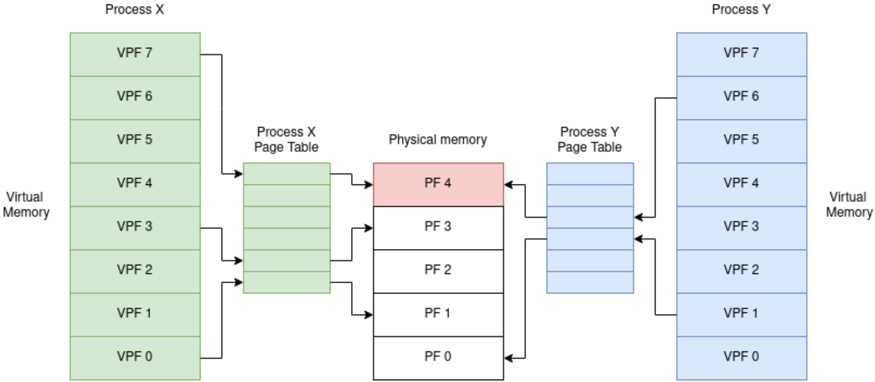
Memory Mapping (II)

- When a process needs to access a memory location that is not in physical memory, the **page table entry** for that location indicates that a **page fault** has occurred.
- The OS loads the required page into physical memory from disk, updating the page table accordingly.

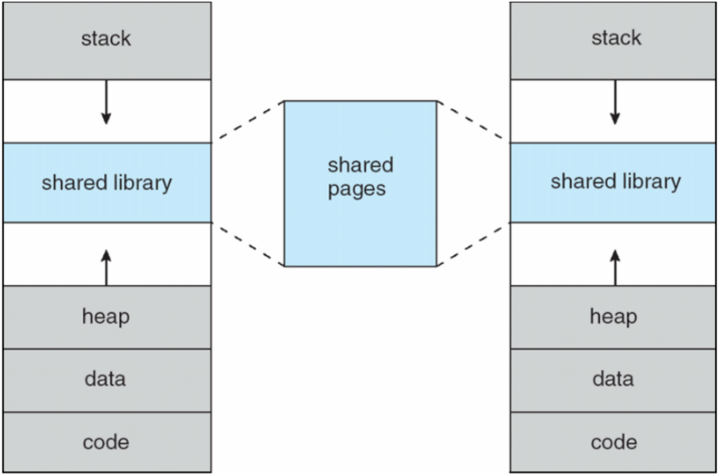


Memory Mapping (III)

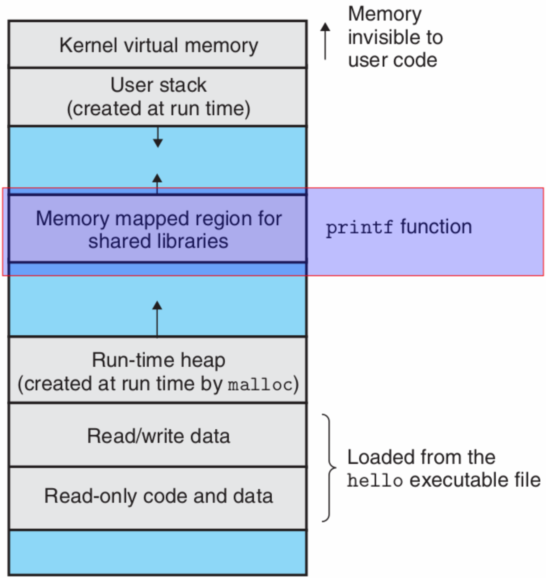
- Each process has its page table.
- This mechanism allows memory to be **shared across several processes**.



Shared Memory (I)



Shared Memory (II)


Kernel virtual memory
Memory invisible to user code
User stack (created at run time)
Memory mapped region for shared libraries
printf function
Run-time heap (created at run time by malloc)
Read/write data
Read-only code and data
Loaded from the hello executable file