Operating systems II
CS 2506

Dr. Dan Grigoras
Computer Science Department

Today's topics

- The role of the operating system.
- What is the relationship between the applications/programs and "the computing system"?

- What do we need to learn and why?

Computing systems

- What is a computer now?

- Computer configuration and architecture changed:
  - CPU/processor/core ➔ many cores, homogeneous or heterogeneous
  - Main memory (plain) ➔ memory hierarchy (cache, main memory)
  - Storage (local or remote), disk ➔ array of large disks
  - Bus ➔ multiple buses
  - Peripherals ➔ + Sensors

- ...or a cloud instance provided as a service.

- *Question*: how are these resources allocated to applications?

Allocation of computing resources

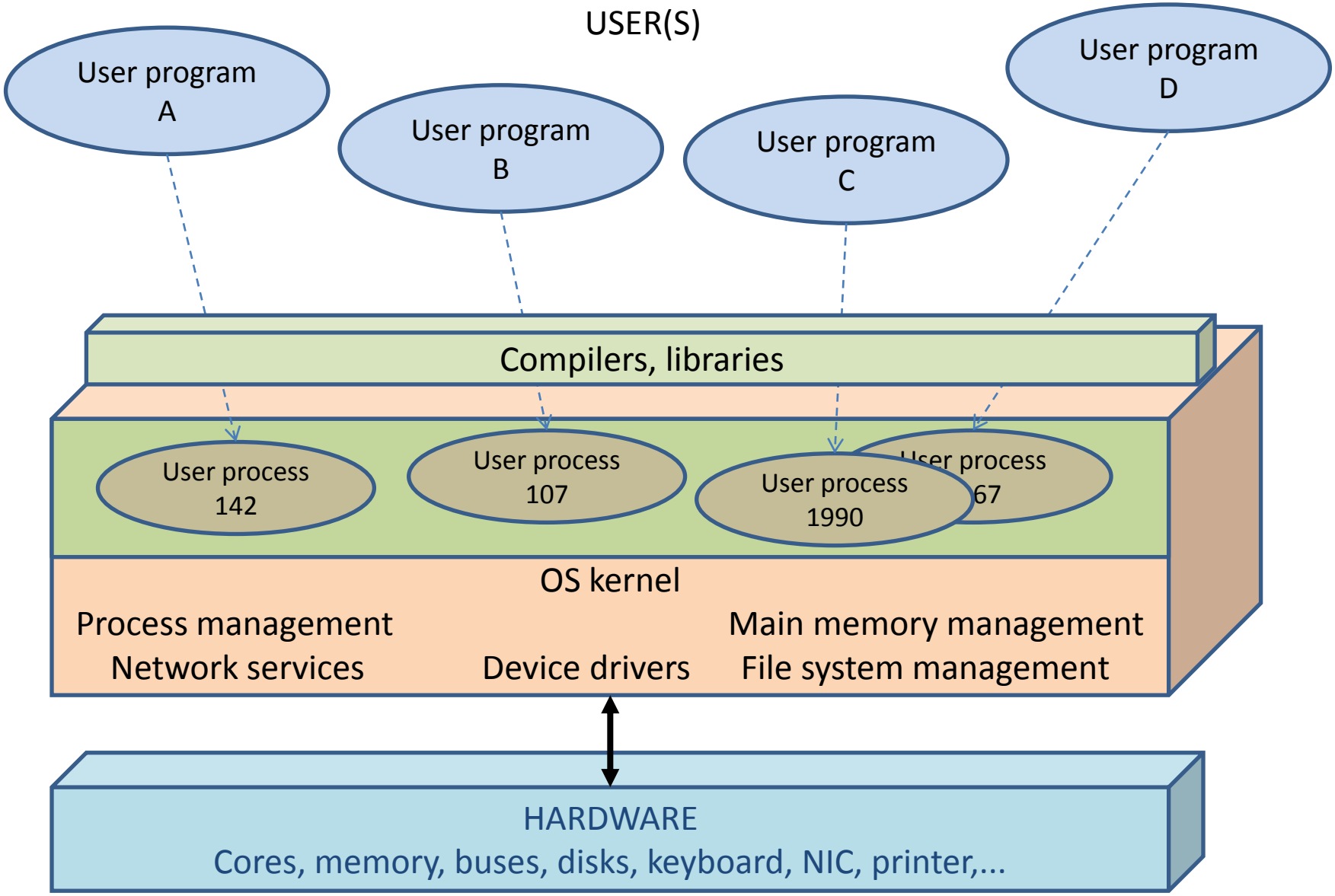
- Generally, there is more than one physical unit of a type of resource in a computer configuration. Examples: cores, memory pages, disk sectors, networking cards, cameras.

- Resource allocation to processes is carried out in terms of units and managed by specific OS services during processes life time.

- Many operating systems are using virtualization of resources (cores, memory pages) for better utilization.

Computing resources management

- OS goals:
  - To meet user requirements
    - Least execution time, quick response;
    - Least cost;
    - Fairness;
    - Best possible user experience.
    - Security.
  - To meet system admin requirements
    - Optimal use of resources;
    - Least energy use (green computing);
    - Maximise revenue.
- The system admin can be the user/owner but not necessarily in all cases – see rack-scale and cloud computing.


USER(S)
User program A
User program B
User program C
User program D
Compilers, libraries
User process 142
User process 107
User process 1990
User process 67
OS kernel
Process management
Main memory management
Network services
Device drivers
File system management
HARDWARE
Cores, memory, buses, disks, keyboard, NIC, printer,...

Computing system layers

- The bottom layer is the hardware; it accepts primitive commands such as “seek the disk arm to track 79, select head 3 and read sector 5”. Software that interacts directly with the hardware is non-portable and contains low level details: data for control and state registers, interrupt priorities, DMA starting address,…

- The OS kernel has several key components:
  - *Device drivers* are the hardware units managers; they hide the low level details.
  - *File sys manager* is the code that organizes the data storage into files and directories, hiding low level details re disk blocks, for example.
  - *Process management* handles processes, allocating them resources and scheduling their execution.
  - *Memory management* is responsible for physical memory and virtual memory management.
  - *Network services* provide host-to-host and process-to-process communication across network.

Access to kernel services

- The kernel can be viewed as a collection of services that user programs may call. They offer functionality and a higher level of abstraction of the computer.
- The repertoire of commands supported by the kernel defines the “virtual machine” which is platform-independent.
- To enter the kernel, the user program makes a *system call by executing a trap instruction of some sort.*
- This instruction switches the processor into a privileged operating mode (*kernel mode) and jumps into the kernel through a well-defined trap address.*
- Parameters passed with the trap instruction tell the kernel what service is requested.
- When the function is completed, the processor flips back to its normal operating mode (*user mode) and returns control to the user program.*

Trap instructions

- There are functions that require specific knowledge of handling resources – control registers, state register, sequence of operations, and a certain degree of protection – resources shared by several users/programs.
- These functions are coded as service routines; they are also known as *system calls*.

- The sequence of steps for a system call is:
  - The system call is invoked by the user program;
  - OS function is performed;
  - Control returns to the user program.

- Trap instructions are used to implement system calls.

Library functions

- User programs have access to libraries and include in their code functions of these libraries – linked in the executables.
- Some library functions use system calls. The function itself parcels up the parameters correctly and then performs the trap.

- The function works as a wrapper for the system call.
- Example:

printf(“hello world”); ➔ write(1, “hello world”, 11);

OS classes

- There isn't one OS that fits all computing devices!

- The large range of computing devices requires customized OS.
- General purpose computers run complex OS: Unix, Linux, Windows,...
- Mobile devices, such as smart phones or sensors, run OS like iPhone OS, Android, TinyOS, concerned with power saving and user experience.

- Embedded systems run scaled down versions of OS, real-time, event-driven.

Trends

- Rapid advances in hardware:
  - peripheral devices are integrated onto a die, each with complex, varying programming models, that execute specialized firmware with little OS integration.
  - for energy saving, cores, devices, and memory can be powered on/off at any point during execution.
  - large, non-volatile main memories will likely become prevalent and might have greater capacity than today's disks and disk arrays.
- Applications are changing:
  - many applications use cloud (data center) services, individual users' mobile devices, and sensors and actuators in the Internet of Things.
  - enterprise applications such as file servers, relational databases, and big data analytics now come pre-packaged in a rack-scale software appliance, delivered to customers who have only to plug in and use the system.
  - Clouds are running applications in containers with their own OS.

Suggested reading

- D. Milojicic and T. Roscoe: *Will OSs in 2025 still resemble the Unix-like consensus of today, or will a very different design achieve widespread adoption?*, Computer, Volume: 49, Jan. 2016, pp. 43-51.
- Available in Boole Library: http://ieeexplore.ieee.org.ucc.idm.oclc.org/document/7383138

- My comment: good personal overview of existing operating systems and technology advances that will have a great impact on operating systems architecture and operation. The paper can suggest interesting avenues of further study and research.

Course goals and methodology

- Goals
  - learn the main services of an operating system:
    - processes and threads management, including scheduling and load balancing, IPC;
    - physical and virtual memory management;
    - device drivers;
    - file management system.
  - learn features of different models of operating systems:
    - general purpose;
    - mobile;
    - sensor.

- Methodology
  - attending the lectures.
  - carrying out the lab work.
  - using recommended references to learn more about specific topics.

- Text book:

A. S. Tanenbaum and H. Bos: Modern Operating Systems, Pearson, 4th edition, 2014.

Course philosophy:

Collaborative learning process

Resources: cs4.ucc.ie/moodle

- Grading
  - Continuous assessment: 20% Labs
  - Written exam: 90 min 80%
- Lecture 50 min + 5 min review & questions

- Office: G65, Western Gateway Bldg
- Email: grigoras@cs.ucc.ie, Subject: CS2506
- Office hours: by appointment

Plagiarism

1. Plagiarism is presenting someone else's work as your own. It is a violation of UCC Policy and there are strict and severe penalties.
2. You must read and comply with the UCC Policy on Plagiarism www.ucc.ie/en/exams/procedures-regulations/
3. The Policy applies to *all* work submitted, including software.
4. You can expect that your work will be checked for evidence of plagiarism or collusion.
5. In some circumstances it may be acceptable to reuse a small amount of work by others, but *only* if you provide explicit acknowledgement and justification.
6. If in doubt ask your module lecturer *prior* to submission. Better safe than sorry!