Anthony Ventresque
anthony.ventresque@ucd.ie

Operating System Components and Structures

School of Computer Science, UCD

Scoil na Ríomheolaíochta, UCD

Announcements

- Everybody should have access to a command line interface now
  - at least on a remote server (ssh/PuTTY)
  - and/or on your own machine (having a network connection will be important too)
  - if you don't, make sure to catch up at this Friday's lab session
- First quiz (***1%*** of the final mark) will be open from this Friday until before the lecture next Tuesday
  - no time limit
  - no limit in the number of attempts
- ***No lecture*** next Tuesday (26th September)
- ***Extra session*** on Friday 29th September (time TBC; Bash commands and scripts)

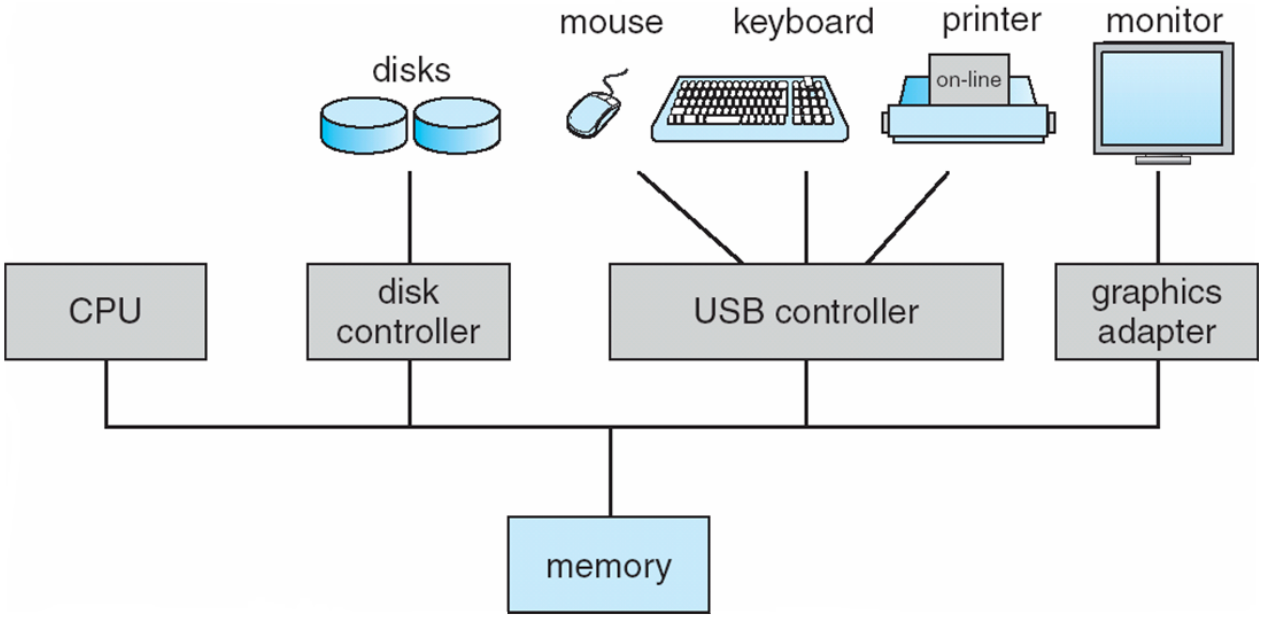
Last Week…

- An OS is a ***program*** that acts as an ***intermediary*** between a user of a computer and the computer hardware
- Goals: Execute user programs, make the computing system easy to use, utilise hardware efficiently
- OS is:
  - ***Resource allocator***: decides between conflicting requests for efficient and fair resource use
  - ***Control program***: controls execution of programs to prevent errors and improper use of computer

Last Week…

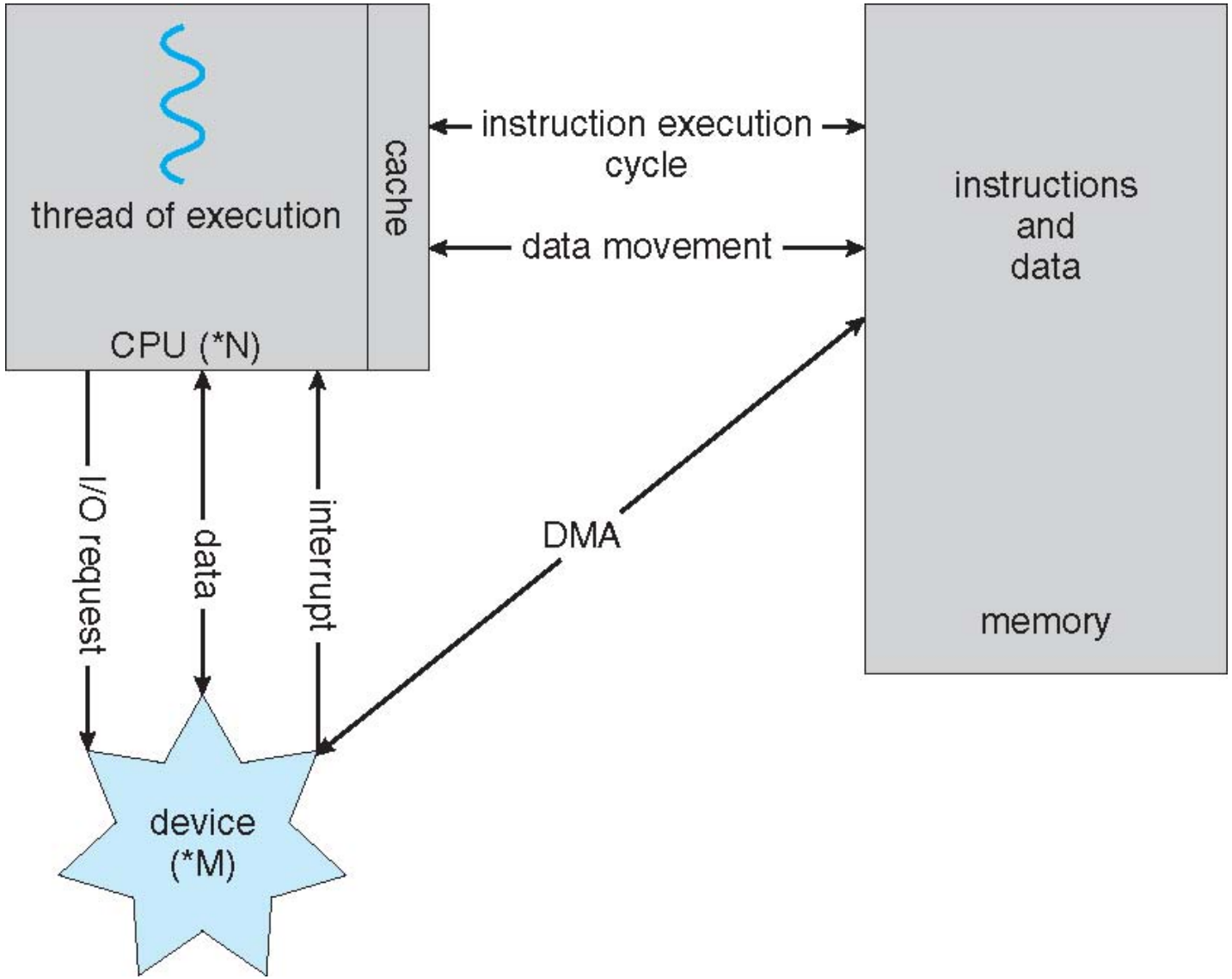
- ***Kernel***: the one program running at all times on the computer
- ***Bootstrap program***: loaded at power-up or reboot

Last Week…



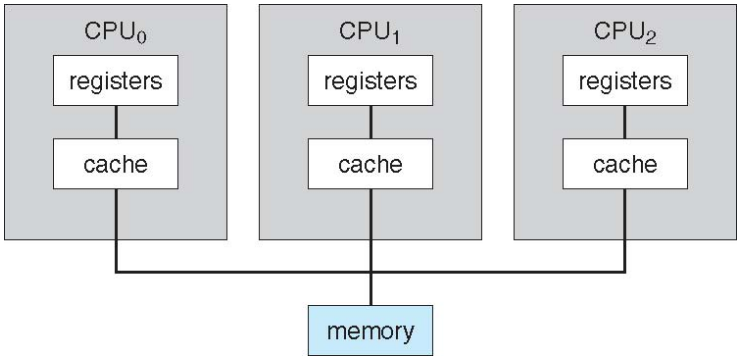
- Device controllers inform CPU that it is finished by causing an ***interrupt***
  - ***Trap*** is a software generated interrupt caused by error or user request
- ***System call***: request to the operating system to allow user to wait for I/O completion

Last Week…

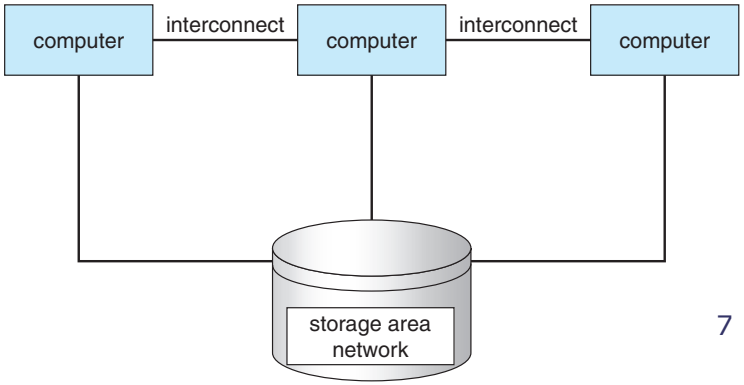

thread of execution
CPU (*N)
cache
instruction execution cycle
data movement
instructions and data
memory
I/O request
data
interrupt
DMA
device (*M)

Last Week…

- ***Multiprocessor*** Systems: Increased throughput, economy of scale, increased reliability
  - Can be asymmetric (1 processor=1 task) or symmetric (each processor performs all tasks)

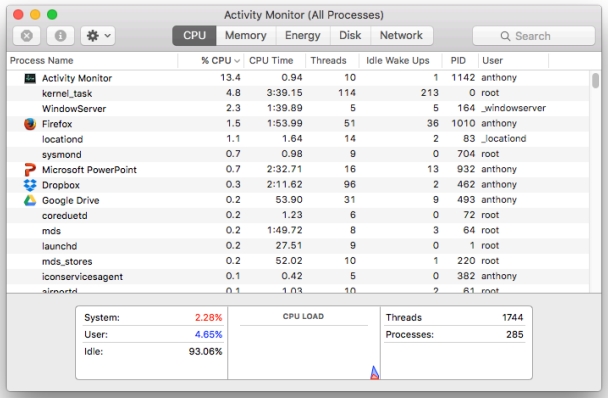


  - Clustered systems – Linked multiprocessor systems



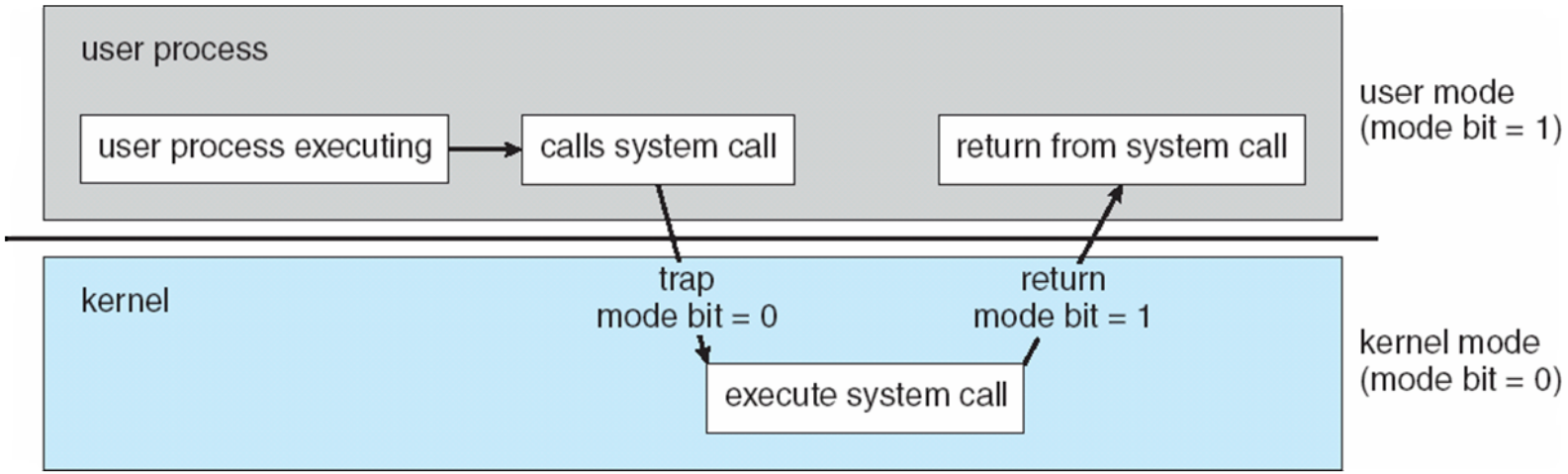
Last Week…

- ***Multiprogramming*** (multitasking) – Provides efficiency via job scheduling
  - When OS has to wait (ex: for I/O), switches to another job
- ***Timesharing*** – CPU switches jobs so frequently that each user can interact with each job while it is running (interactive computing)



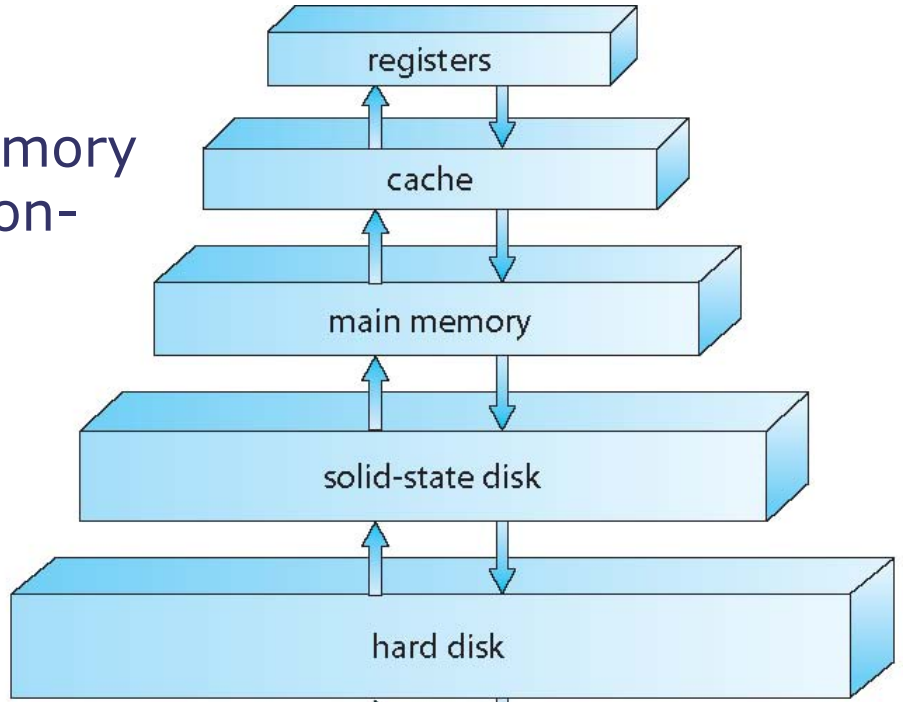
Last Week…

- ***Dual-mode operation*** allows OS to protect itself and other system components – User mode and kernel mode
  - Some instructions are only executable in kernel mode, these are privileged
  - Nowadays most CPUs can support multi-mode operations (e.g., VM managers)



Last Week…

- Storage structure:
  - Main memory – random access, volatile
  - Secondary storage – extension of main memory That provides large non-volatile storage
- ***Caching*** – copying information into faster storage system



…

Outline

- To describe the services an OS provides to users, processes and other systems
- To discuss the various ways of structuring an OS
- Understanding the different principles of OS organisation including abstractions and layering

Take home message:

An OS provides a number of components to users, processes, and other systems

Operating systems can be structured using different principles of organisation (i.e., Simple, Monolithic, Layered, Microkernel, Modular)

OS Structure

- How to organise an OS?
  - What are the essential components involved?
  - Where do they exist?
  - How do they cooperate?
- The task is not trivial if all system goals must be met:
  - Efficiency (high throughput)
  - Interactivity
  - Robustness (fault tolerance & reliability), security
  - Scalability, extensibility, portability

OS Services

- Operating systems provide an environment for execution of programs and services to programs and users
- One set of OS services provides functions that are helpful to the user:
  - ***User interface*** - Almost all operating systems have a user interface (UI).
    - Varies between ***Command-Line*** (CLI), ***Graphics User Interface*** (GUI), ***Batch***
  - ***Program execution*** - The system must be able to load a program into memory and to run that program, end execution, either normally or abnormally (indicating error)
  - ***I/O operations*** - A running program may require I/O, which may involve a file or an I/O device

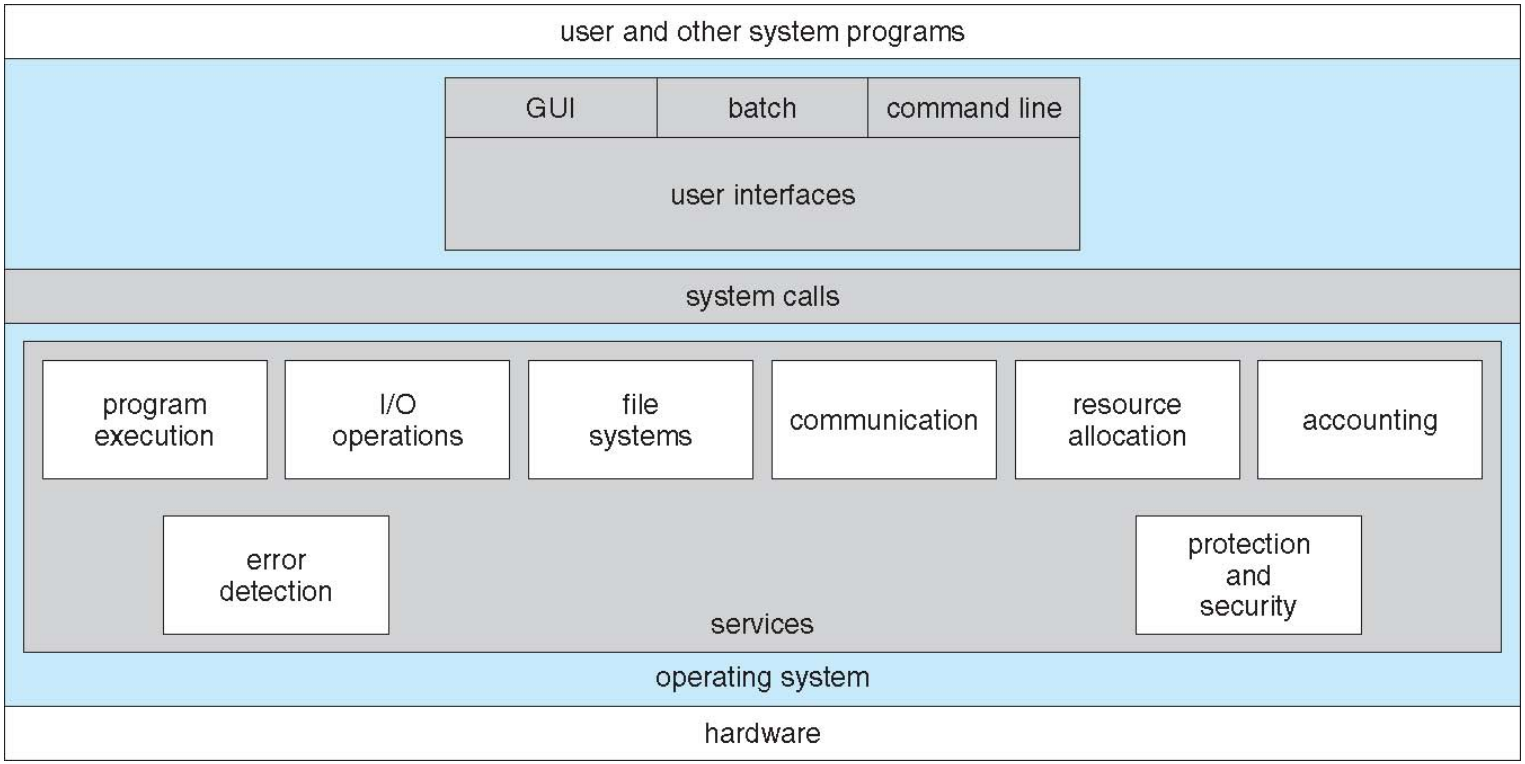
OS Services

- One set of operating-system services provides functions that are helpful to the user (Cont.):
  - ***File-system manipulation*** - Programs need to read and write files and directories, create and delete them, search them, list file Information, permission management.
  - ***Communications*** – Processes may exchange information, on the same computer or between computers over a network
    - Communications may be via shared memory or through message passing (packets moved by the OS)
  - ***Error detection*** – OS needs to be constantly aware of possible errors
    - May occur in the CPU and memory hardware, in I/O devices, in user program
    - For each type of error, OS should take the appropriate action to ensure correct and consistent computing
    - Debugging facilities can greatly enhance the user's and programmer's abilities to efficiently use the system

OS Services

- Another set of OS functions exists for ensuring the efficient operation of the system itself via resource sharing
  - ***Resource allocation*** - When multiple users or multiple jobs running concurrently, resources must be allocated to each of them
    - Many types of resources - CPU cycles, main memory, file storage, I/O devices.
  - ***Accounting*** - To keep track of which users use how much and what kinds of computer resources
  - ***Protection and security*** - The owners of information stored in a multiuser or networked computer system may want to control use of that information, concurrent processes should not interfere with each other
    - Protection involves ensuring that all access to system resources is controlled
    - Security of the system from outsiders requires user authentication, extends to defending external I/O devices from invalid access attempts

A View of OS Services

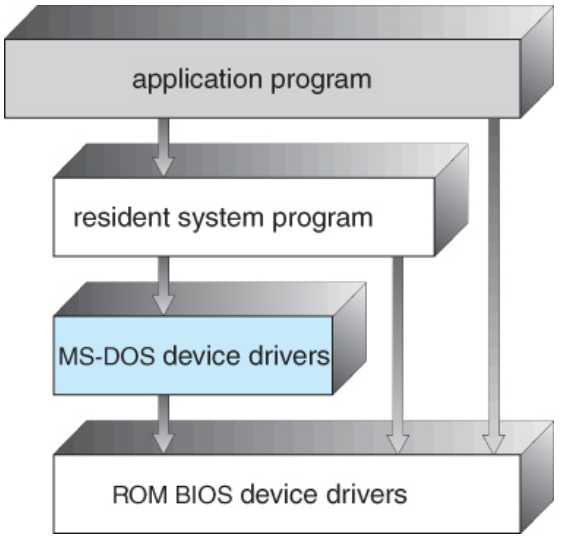


Different OS Organisational Principles

- General-purpose OS is very large program
- Various ways to structure OS
  - ***Simple*** Structure: Only one or two levels of code (e.g MS-DOS)
  - ***Monolithic*** Architectures: More complex (e.g. UNIX)
  - ***Layered***: Lower levels independent of upper levels (an abstraction)
  - ***Microkernel***: OS built from many user-level processes (Mach)
  - ***Modular***: Core kernel with Dynamically loadable modules

Simple Structure: MS-DOS

- MS-DOS – written to provide the most functionality in the least space
  - Not divided into modules
  - Although MS-DOS has some structure, its interfaces and levels of functionality are not well separated



Monolithic Architecture

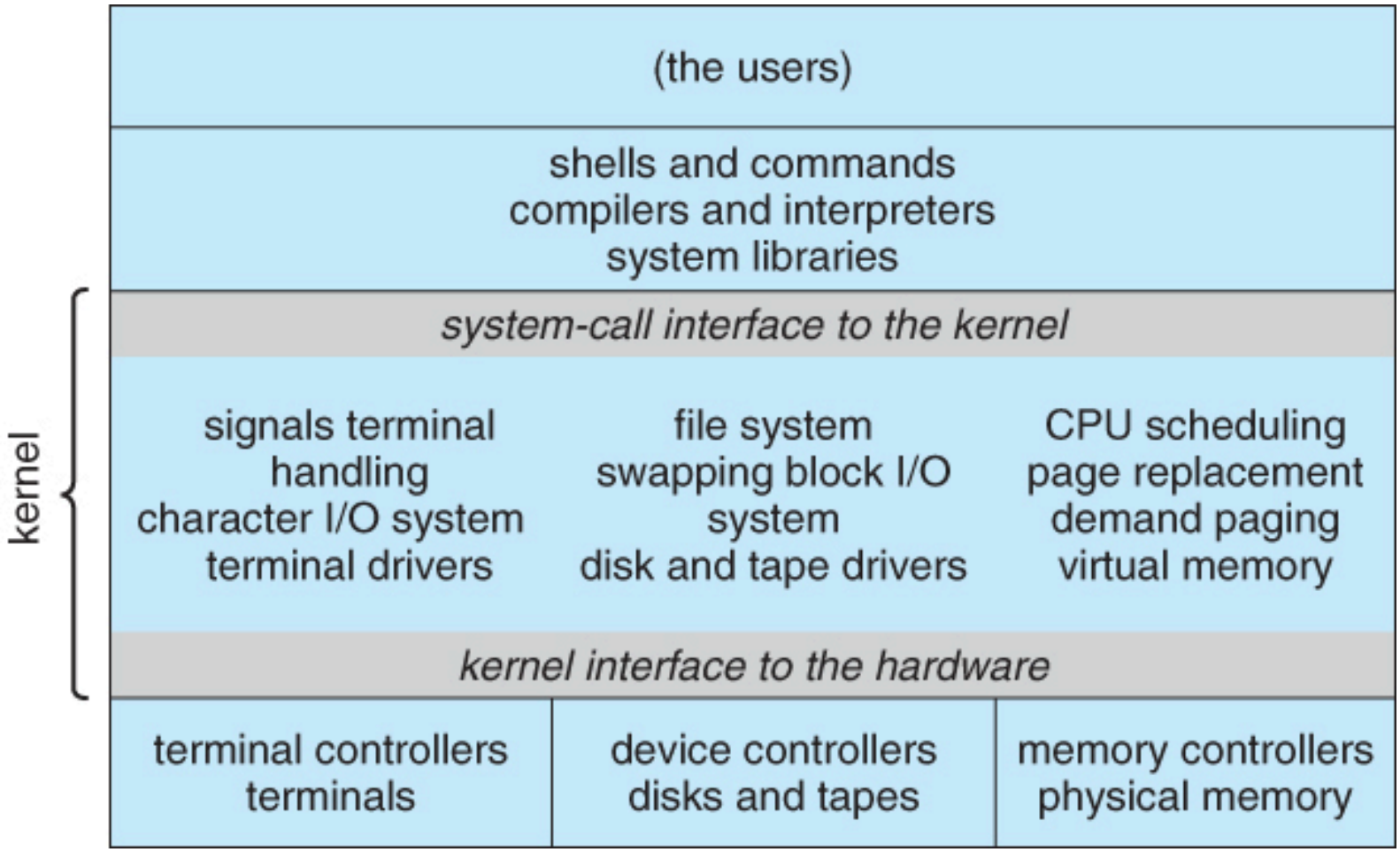
- Traditionally, systems were built around monolithic kernels
  - every OS component is contained in the kernel
  - any component can directly communicate with any other (by means of direct function calls)
    - due to this they tend to be highly efficient (high performance)

Monolithic Architecture

- Problems:
  - Unstructured → scalability makes them unwieldy
    - hard to understand, modify & maintain (e.g., debug)
  - Susceptible to damage from errant or malicious code, as all kernel code runs with unrestricted access to the system

- Examples: OS/360, VMS, Unix-like OSs (Linux, BSD)

Unix System Structure



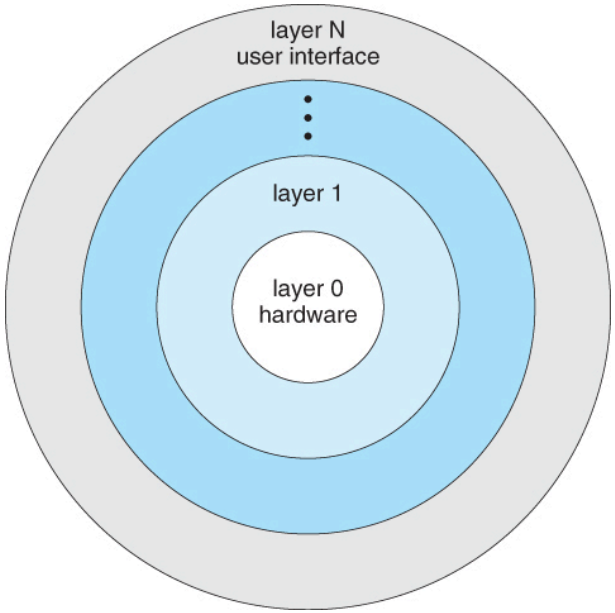
Layered Architecture

- Improves on monolithic kernel designs by adding structure
  - Components are grouped into layers that perform similar functions
  - Each built on top of lower layers
  - Bottom layer (layer 0) is hardware
  - Highest layer (layer N) is the user interface

- Advantages: modularity imposes structure and consistency (easier validation, debugging, modification, reuse)

Layered Architecture

- Each layer communicates only with layers immediately above and below it
  - each layer is a virtual machine to the layer above
  - a higher layer provides a higher-level virtual machine



First Layering-based OS

Layer	Function
5	The operator
4	User programs
3	Input/output management
2	Operator-process communication
1	Memory and drum management
0	Processor allocation and multiprogramming

Structure of the THE operating system.

- Layering was first used in Dijkstra's THE OS (1968) layer 4 (user space) sees virtual I/O drivers
  - each layer "sees" a logical machine provided by lower layers:
    - layer 4 (user space) sees virtual I/O drivers
    - layer 3 sees virtual console
    - layer 2 sees virtual memory
    - layer 1 sees virtual processors 9/26 2nd Chapter: OS Structure
  - Based on a static set of cooperating processes
  - Each process can be tested and verified independently

Problems with Layering

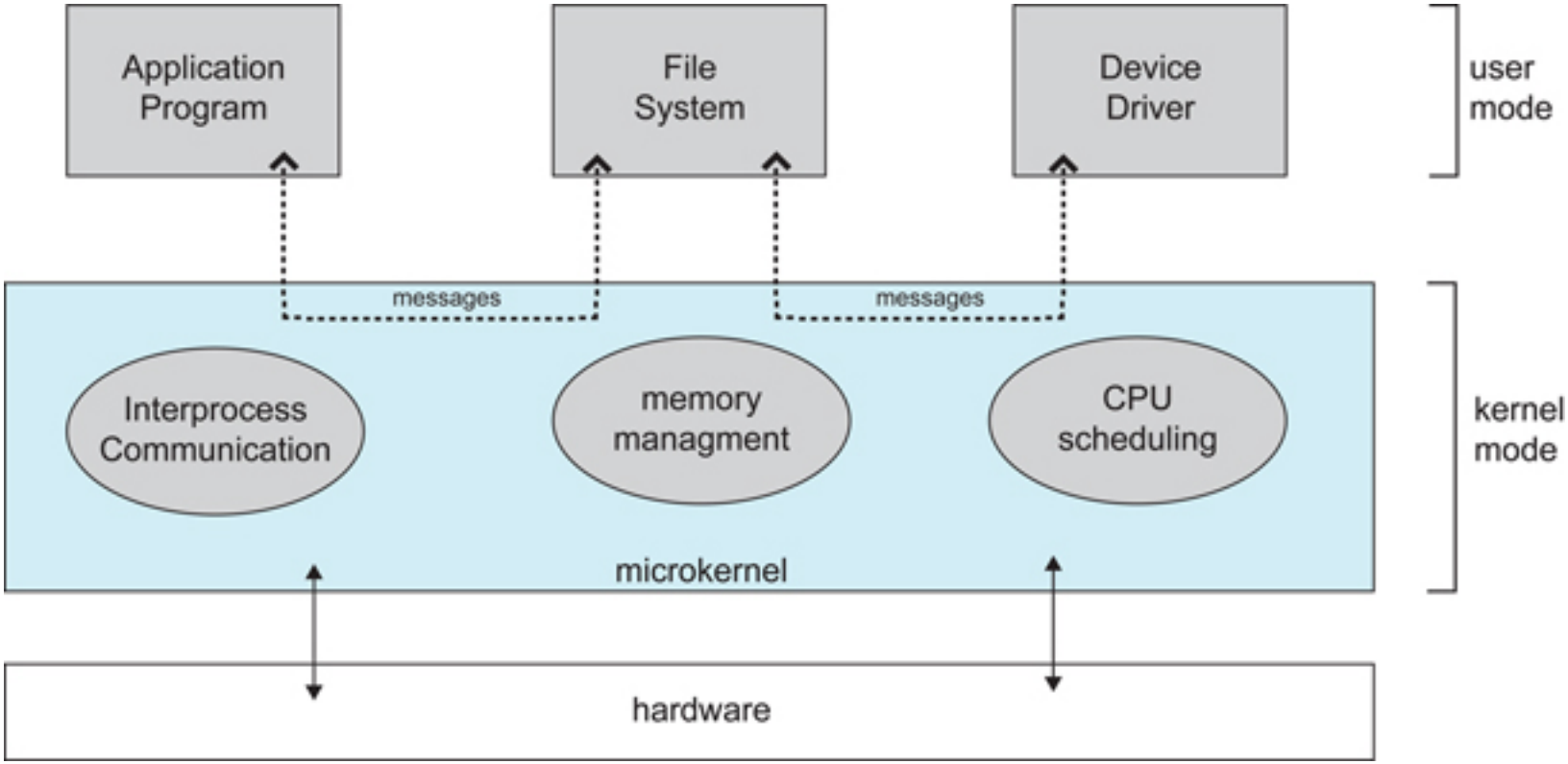
- Appropriate definition of layers is difficult
  - A layer is implemented using only those operations provided by lower-lever layers
  - A real system structure is often more complex than the strict hierarchy required by layering
- Performance issues
  - Processes' requests might pass through many layers before completion (layer crossing)
  - System throughput can be lower than in monolithic kernels

Microkernel System Architecture

- Minimise the services offered by the kernel, in an attempt to keep it small and scalable
  - Small core OS running at kernel level
  - OS Services built from many independent user-level processes
  - No consensus about minimal set of services inside micro- kernel
    - At least: minimal process and memory management capabilities, plus inter-process communications
  - Services such as networking and file system tend to run non-privileged at the user process level (i.e. out of the micro-kernel)

- Communication takes place between user modules using ***message passing***

Microkernel System Structure



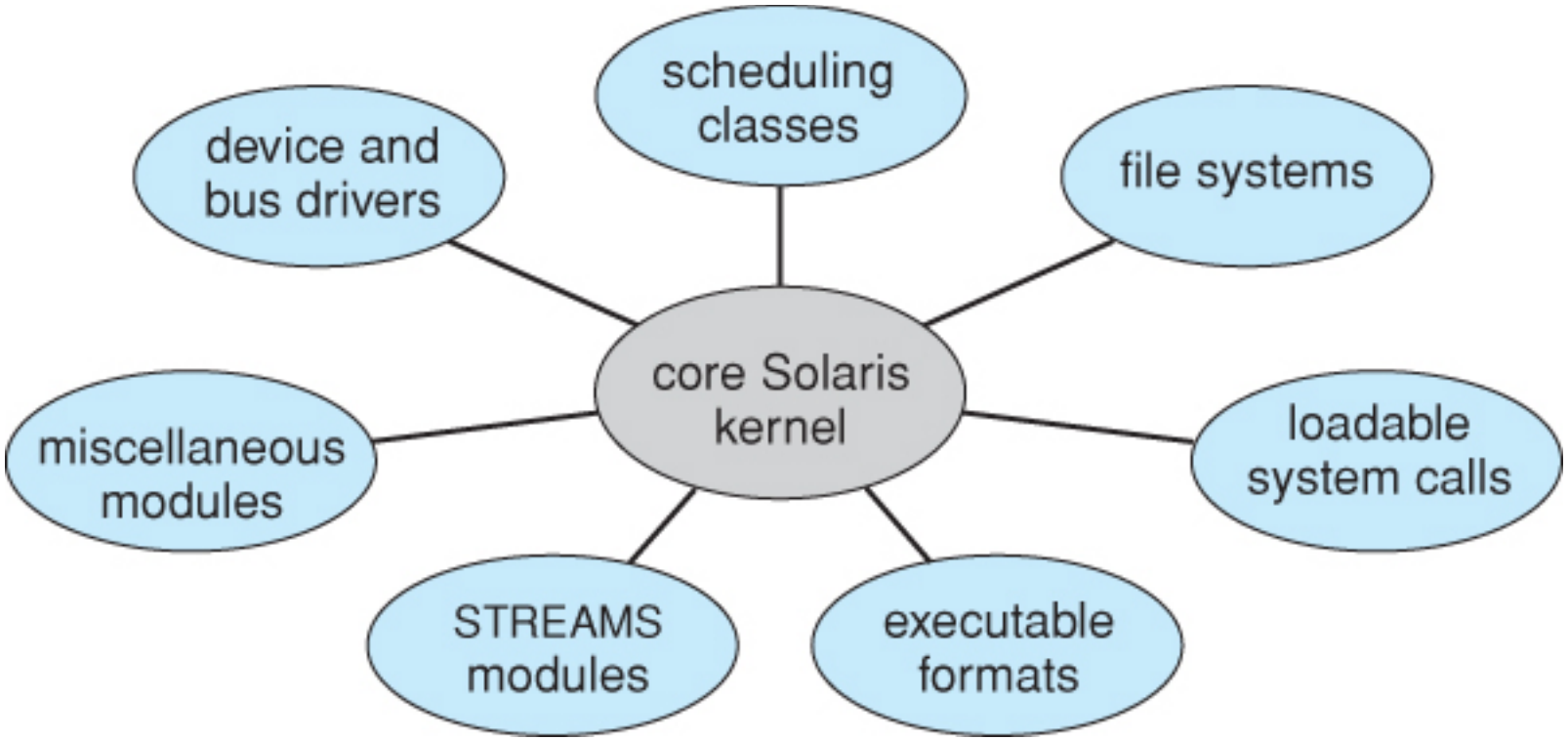
Microkernel System Structure

- Benefits:
  - Easier to extend a microkernel
  - Easier to port the operating system to new architectures
  - More reliable (less code is running in kernel mode)
  - More secure
- Drawbacks:
  - Performance overhead of user space to kernel space communication
- Examples:
  - Hydra (CMU, 1970): first µ-kernel
  - Mach (CMU)
  - Mac OS X kernel (Darwin) partly based on Mach
  - Chorus/MiX (Unix-like distributed OS)
  - Windows NT (originally a layered microkernel architecture)
  - Symbian

Modules-based Structure

- Many modern operating systems implement ***loadable kernel modules***
  - Uses object-oriented approach
  - Each core component is separate
  - Each talks to the others over known interfaces
  - Each is loadable as needed within the kernel
- Overall, similar to layers but with more flexibility
  - Linux, Solaris, etc.

Modules-based Structure



Conclusion

- An operating system provides a number of services/ components to users, processes, and other systems (i.e. memory management, I/O, scheduling, etc.)
- Various ways to structure OS
  - ***Simple*** Structure: Only one or two levels of code (e.g., MS-DOS)
  - ***Monolithic*** Architectures: More complex (e.g., UNIX)
  - ***Layered***: Lower levels independent of upper levels (an abstraction)
  - ***Microkernel***: OS built from many user-level processes (Mach)
  - ***Modular***: Core kernel with Dynamically loadable modules