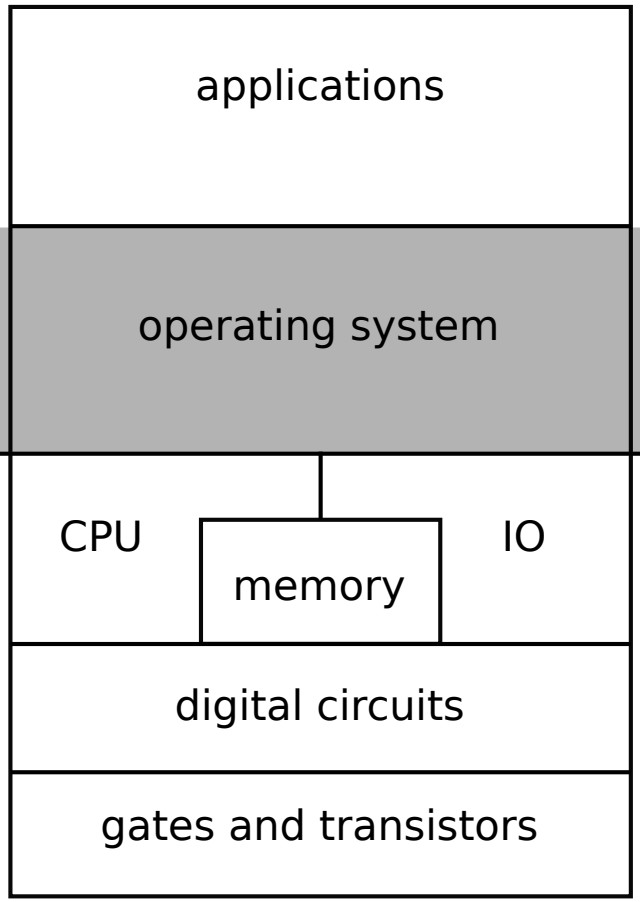
EssCS - Topic 3

Introduction to Operating Systems

Lecture 10, 05.11.2024

Nuša Zidarič

Introduction to Operating Systems

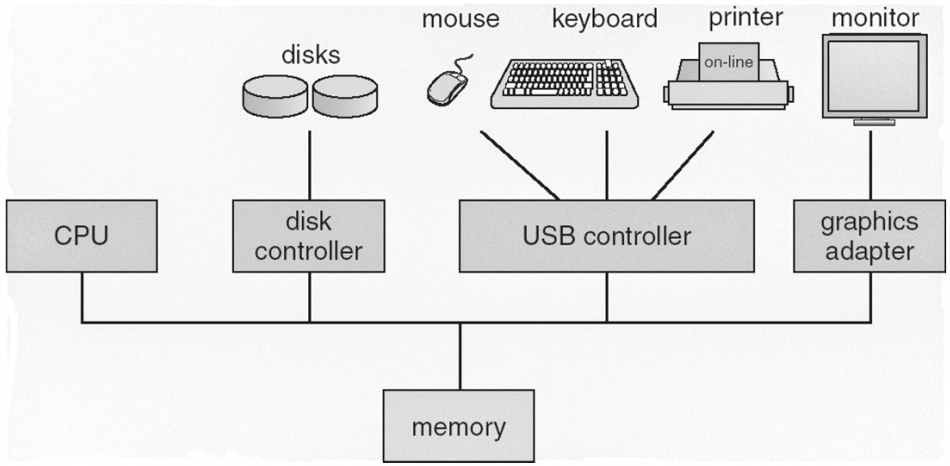


- OS = Operating System
  - OS is software that manages computer hardware and provides a user interface for interaction with the system
  - it coordinates the use of HW between various applications and various users
- OS as resource allocator (manager):

  CPU, memory (main memory & storage), IO, network, ...
- OS as control and isolation program:

  preventing errors and misuse
- goals: ease of use, performance, extensibility
- however: different environments/requirements $\rightarrow$ different OS (and different terminology!)

General-purpose computer system



CPU[1] and device controllers connected through common bus to a shared memory

- OS has a device driver for each device controller → uniform interface to the device
- CPU and device controllers can execute in parallel → competing for memory access

`https://www.youtube.com/watch?v=0dziYWEkDIM`

[1] one or more CPUs

System Boot

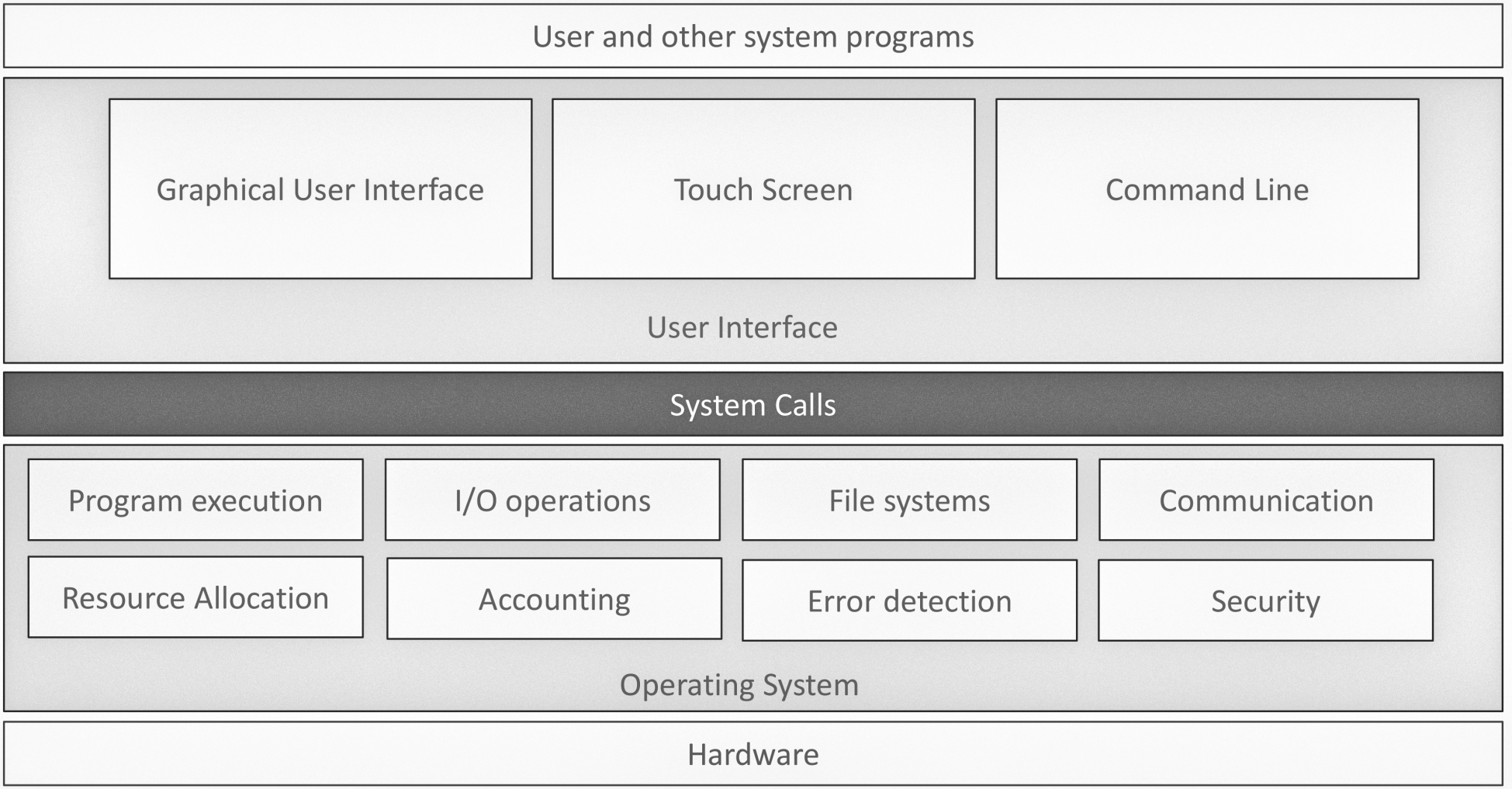
- recall memory hierarchy: registers, cache (L1, L2), main memory (also called primary storage) $\Leftarrow$ volatile memory (temporary, contents are gone when powered off)
- secondary storage (e.g., hard-disk drives) is nonvolatile (permanent storage)

  when powered-on, a special circuit forces a reset and the PC to an address mapped containing a bootstrap program
  - old example: BIOS[2], a small bootloader, stored in a firmware chip:
    - it runs POST (power on self test) and initializes device controllers
    - it loads and executes a second boot loader found in the boot block location[3] in secondary storage
    - second bootloader locates the kernel, loads it into main memory and starts executing
    - this is a multi-stage boot process!
  - modern systems: UEFI (Unified Extensible Firmware Interface) stored in a pre-defined location in secondary storage
    - a complete boot manager, it is faster, has better support for 64-bit systems and for larger drives

[2] Basic Input Output System

[3] wlog: pre-defined location

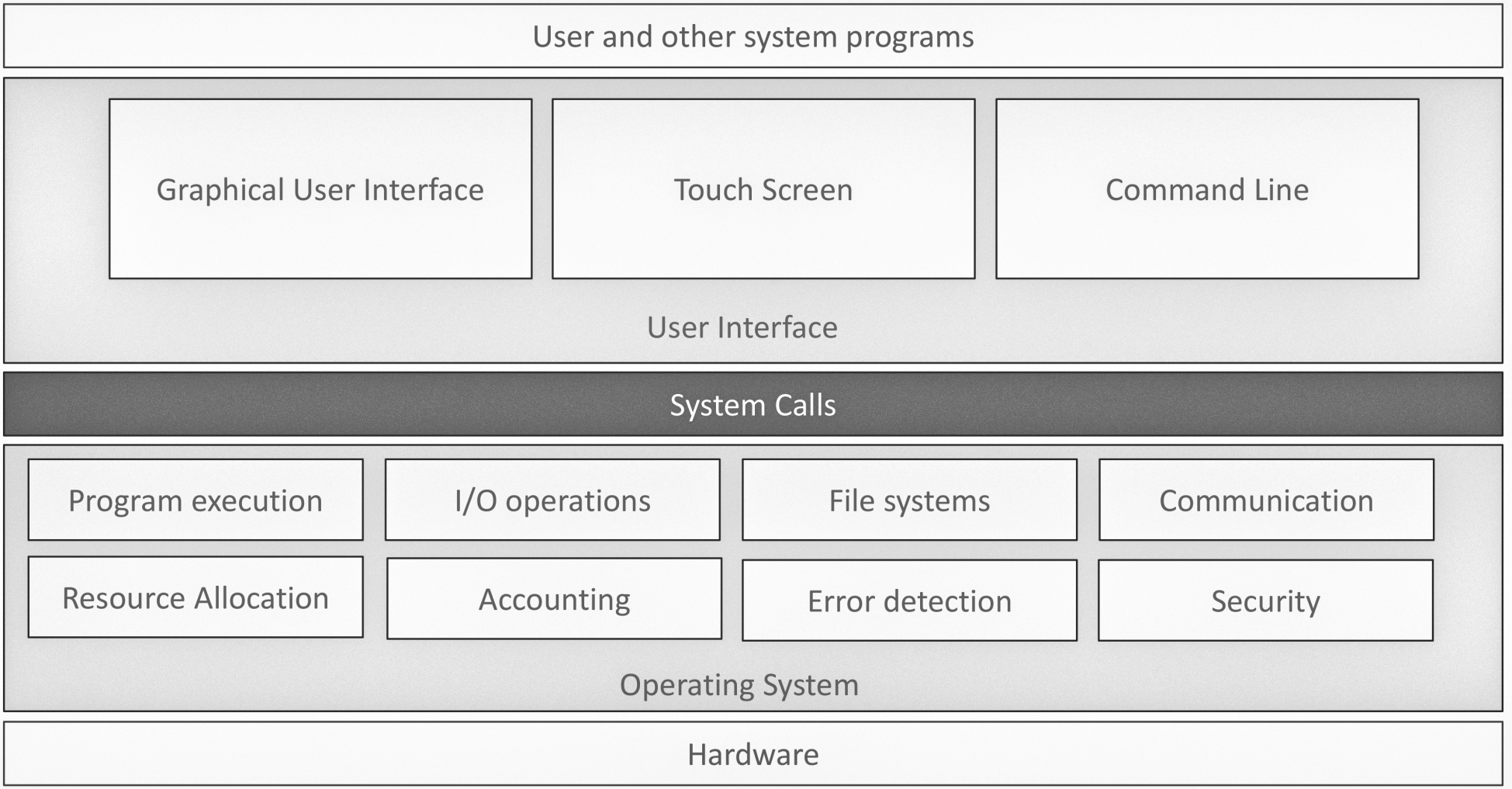
Introduction to Operating Systems



- OS "provides services" to

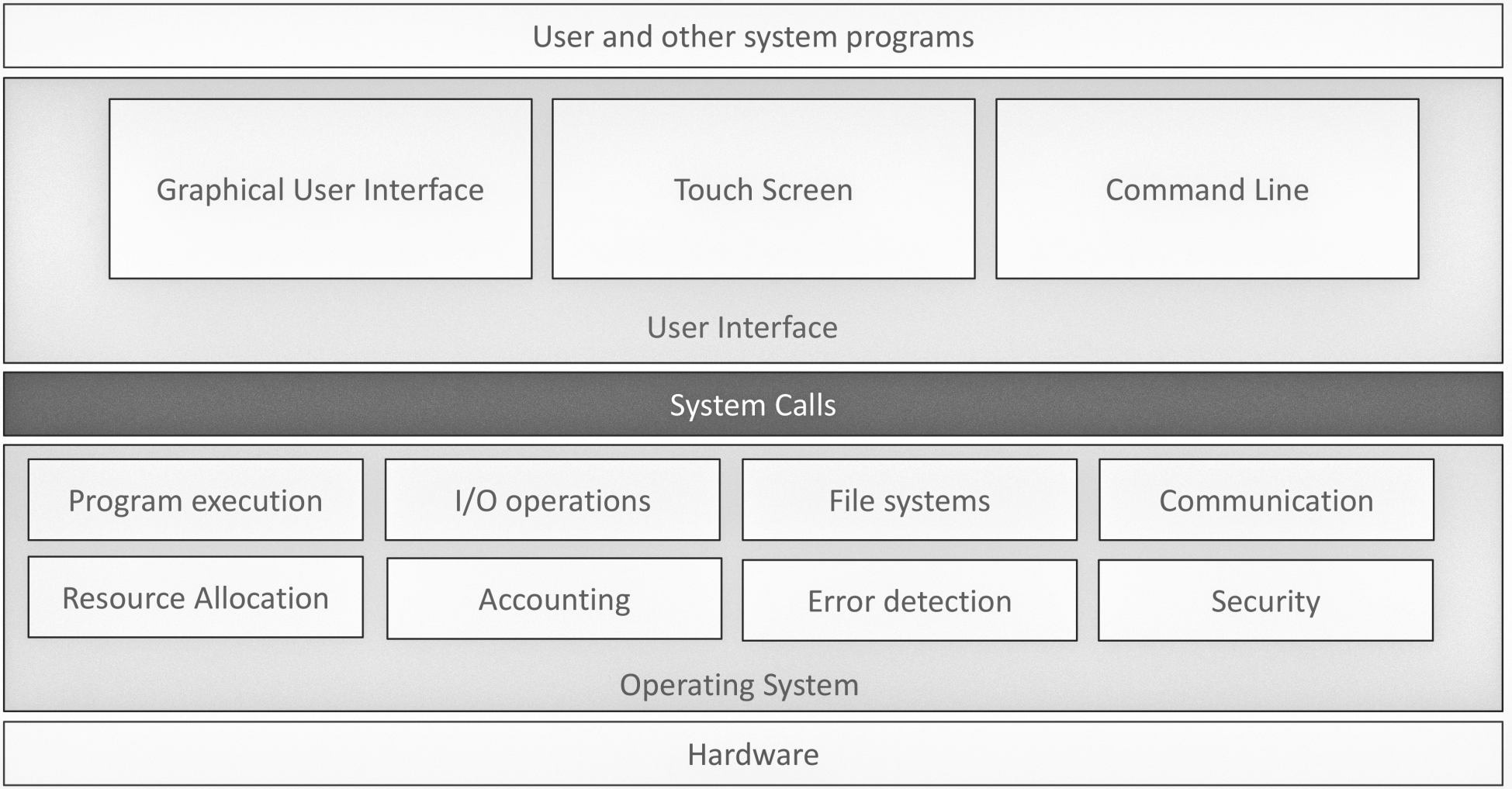
  system: resource allocation, accounting, security

  user: program execution, I/O operations, File systems, ...



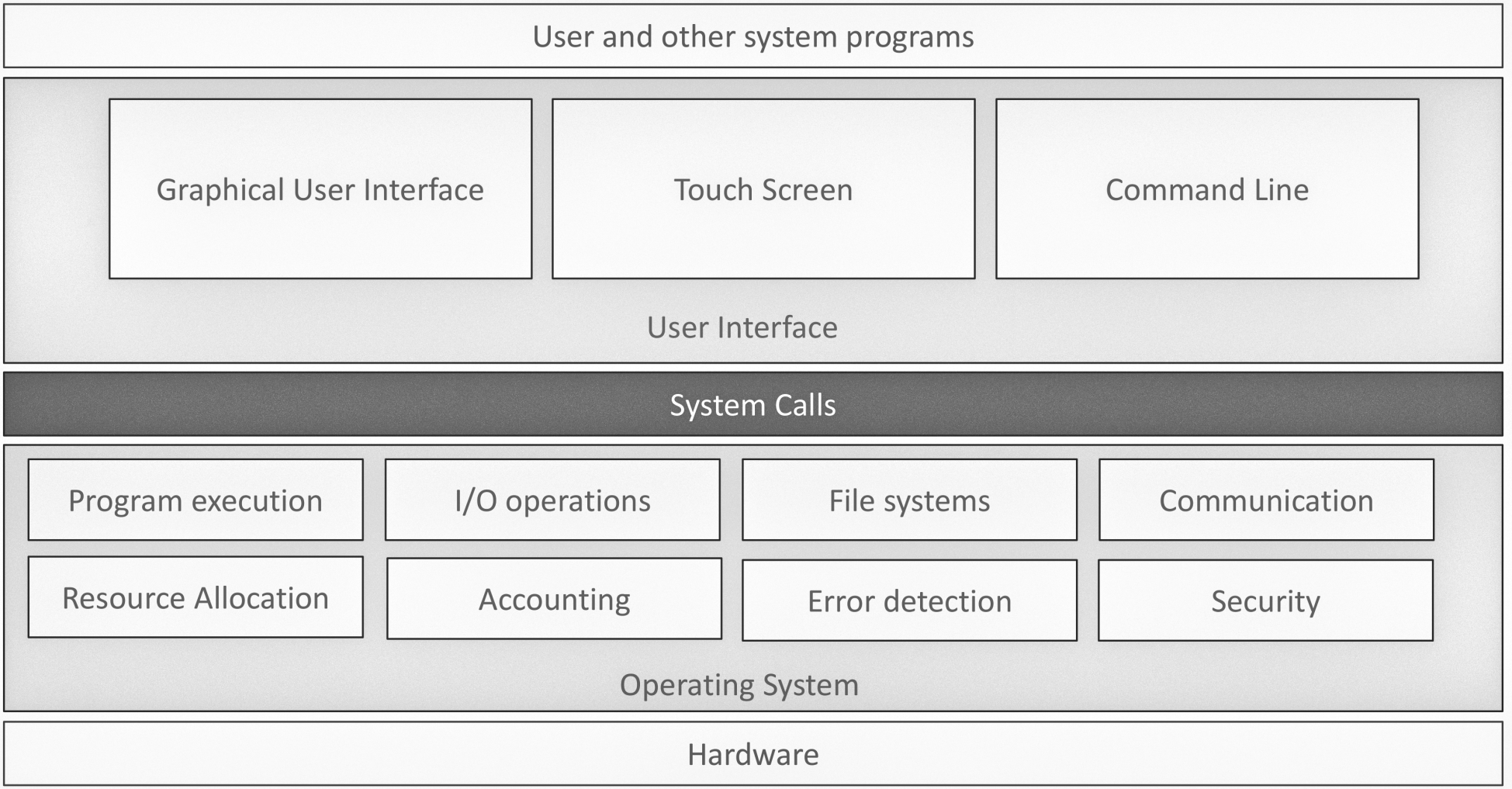
Program execution: the system must be able to load a programs into main memory, start its execution, and end its (normally or abnormally - error)

process = (a part of) a program in execution



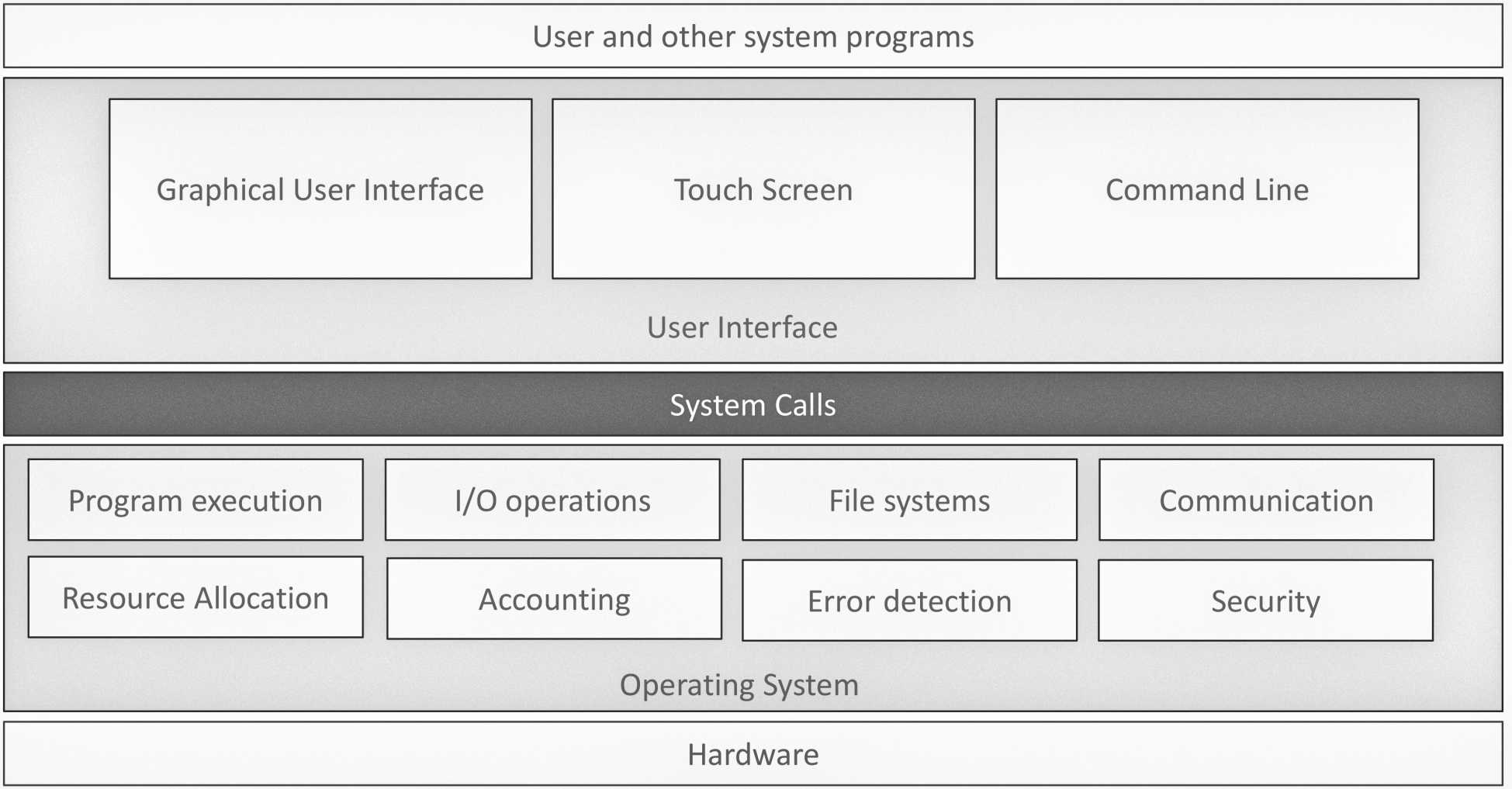
I/O operation: action of accessing devices (e.g., reading from a network interface)

Efficiency and protection: users cannot access hardware devices directly

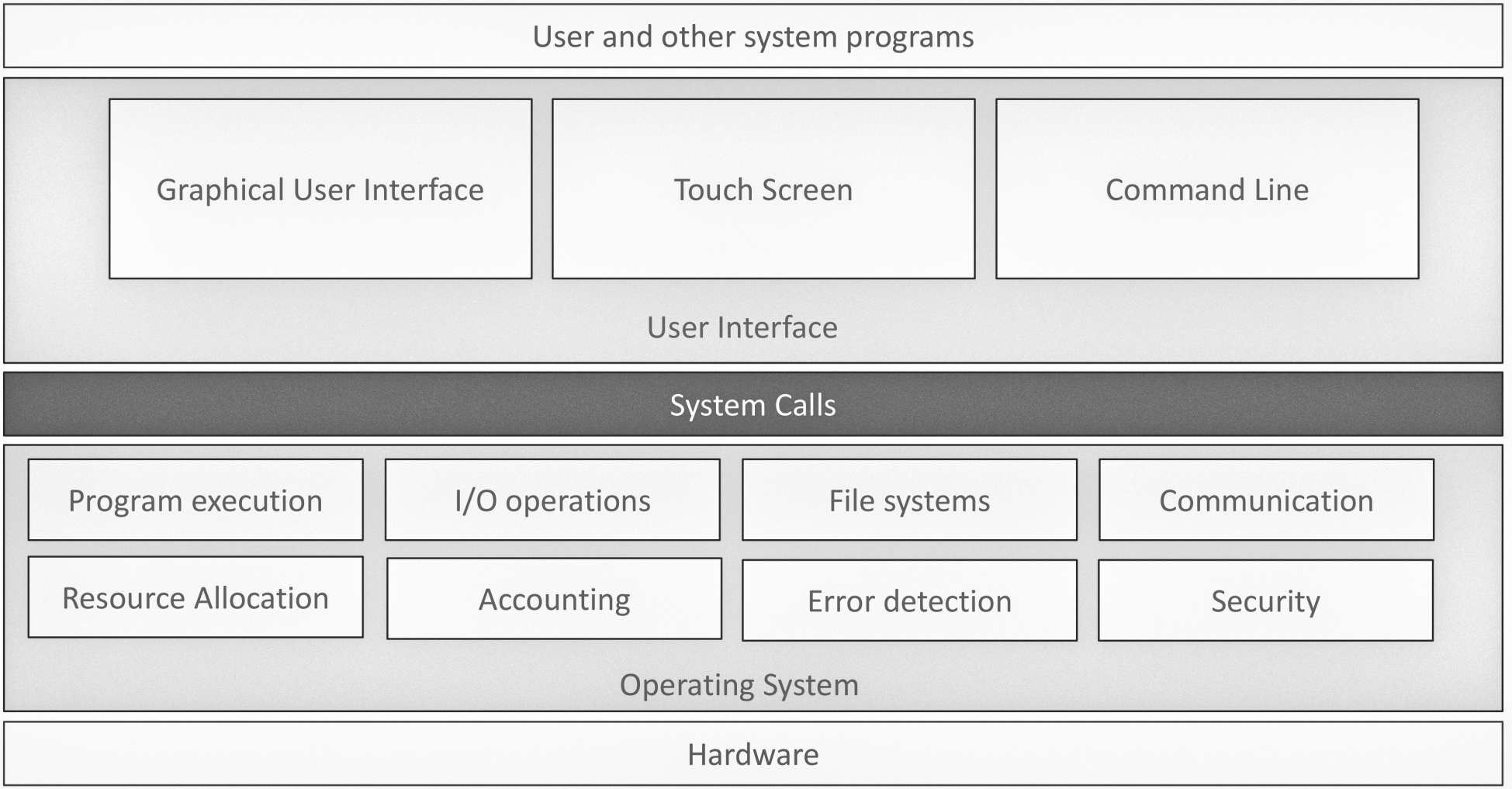


File systems: programs need to read/write files and directories

other tasks: create, delete, search, access control (based on permissions)



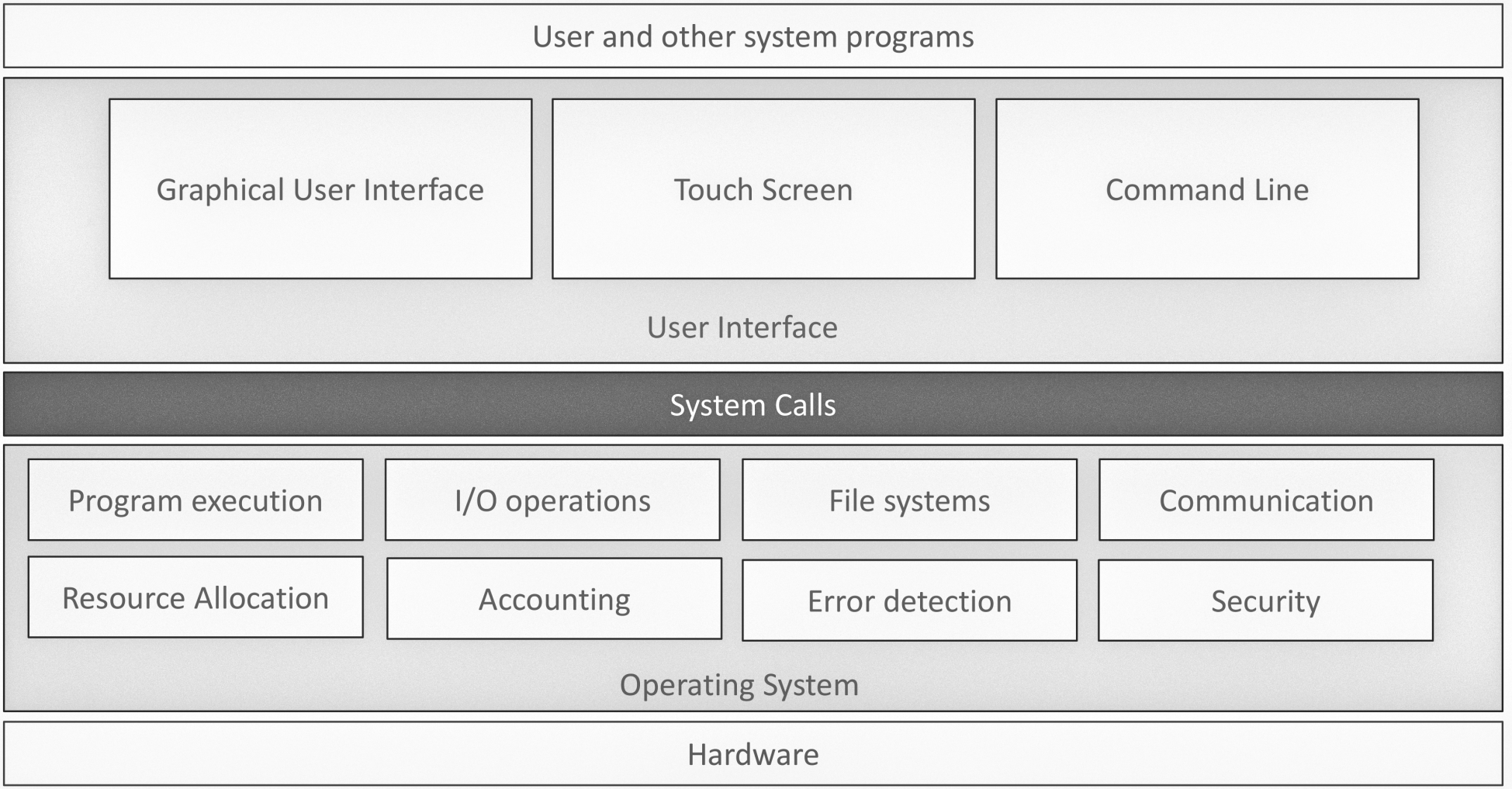
Communication: between processes (IPC = Inter Process Communication)
either between processes executing on the same computer
or between processes executing on different computers connected by a network



Error detection: detecting and correcting[4] errors (also report/log)

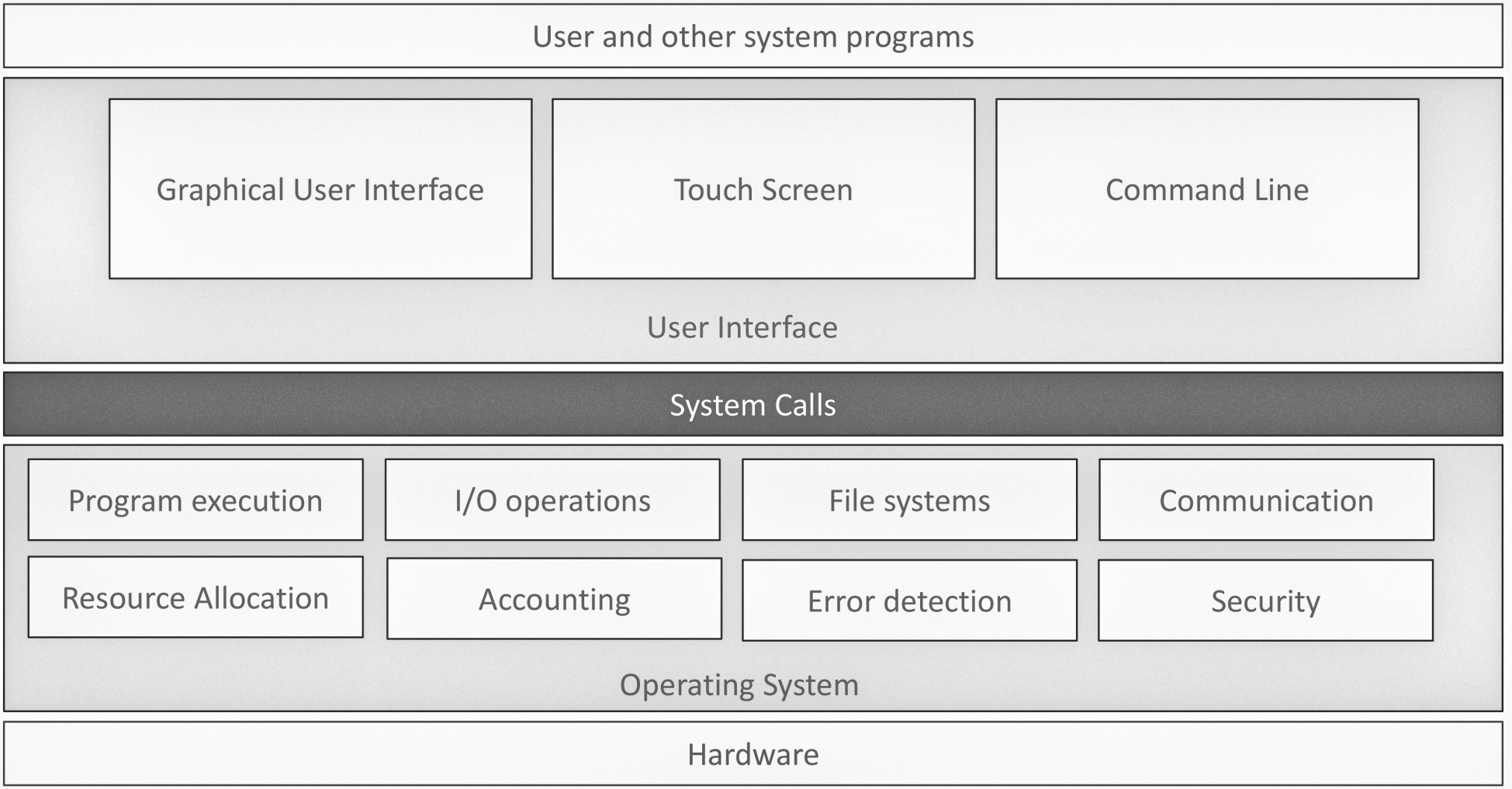
errors in CPU, memory, IO, user program

[4] if possible

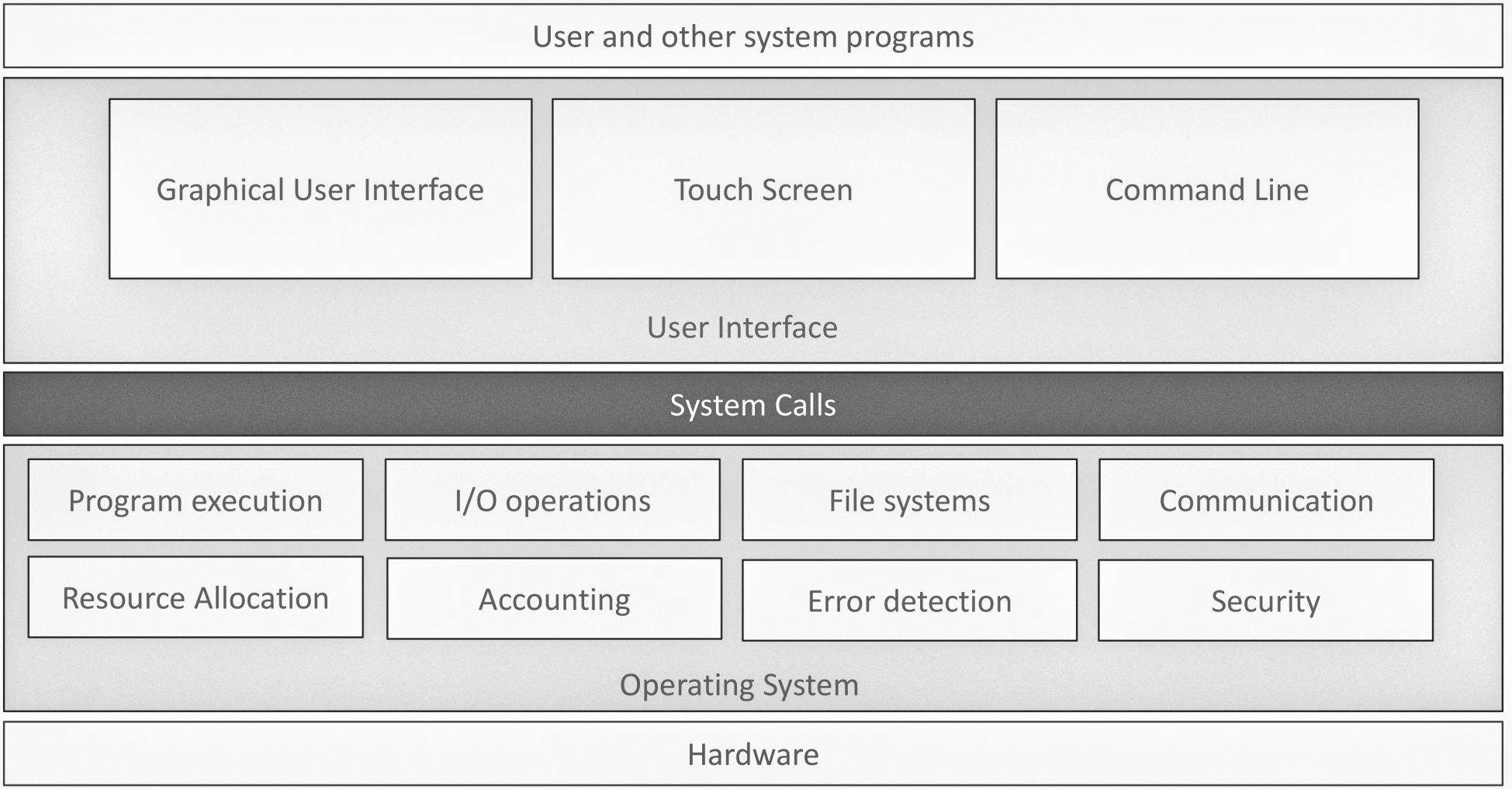


Resource allocation: when multiple processes running at the same time

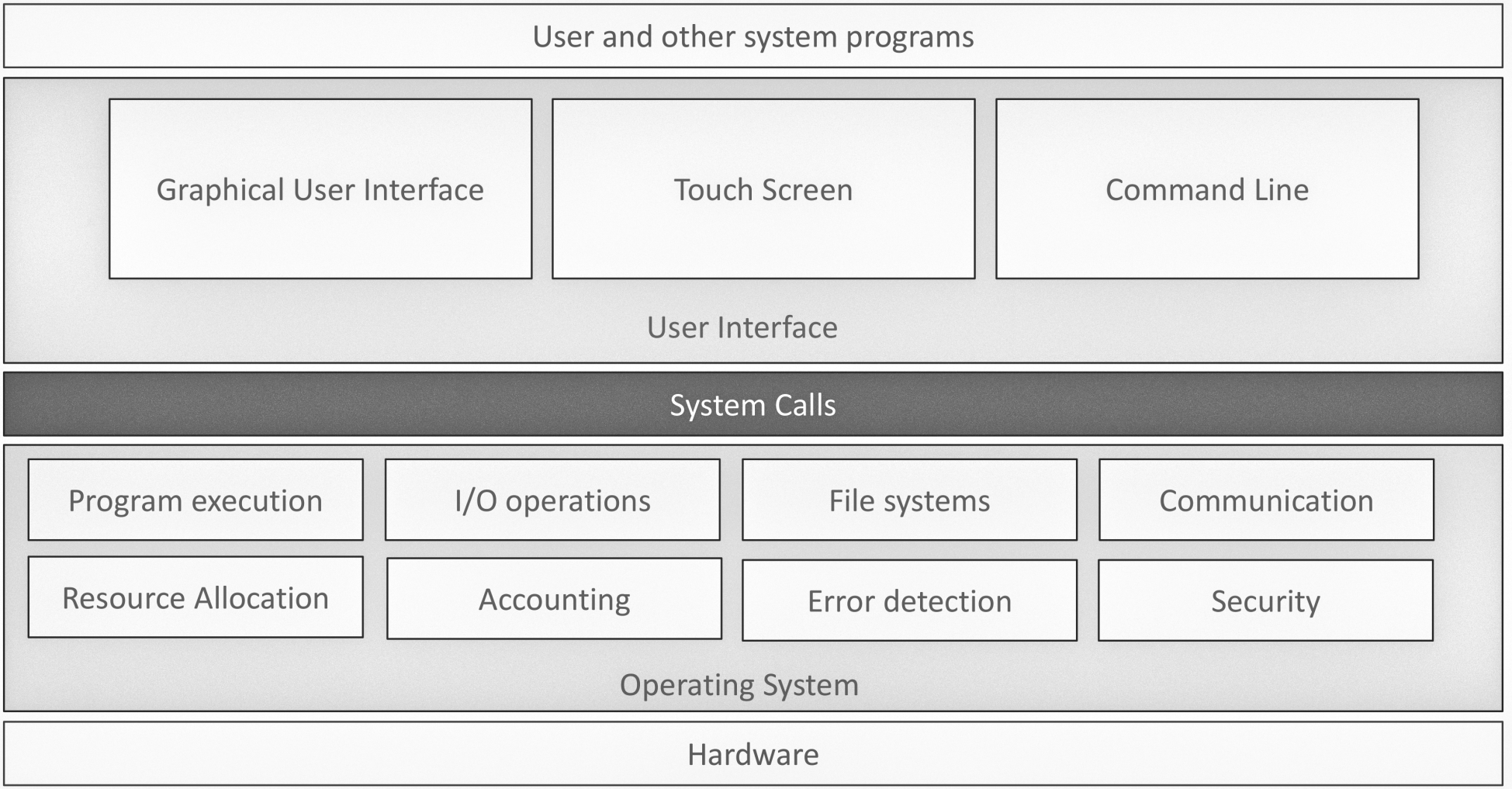
CPU scheduling, memory allocation, ...



Accounting: logging which programs use how much and what kind of resources

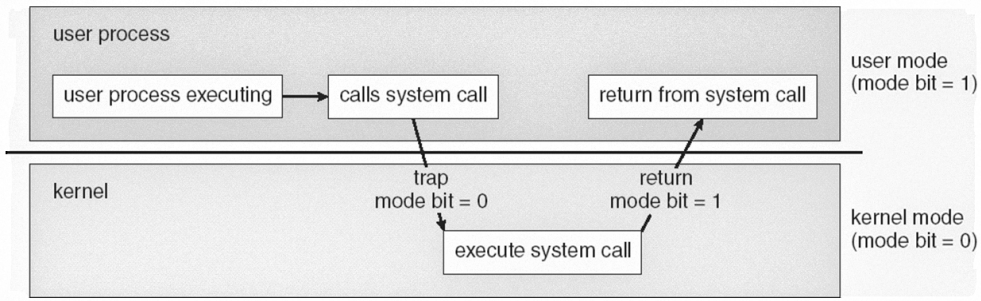


Security: access control to stored information and devices, user authentication



System calls: means by which programs request services from the OS

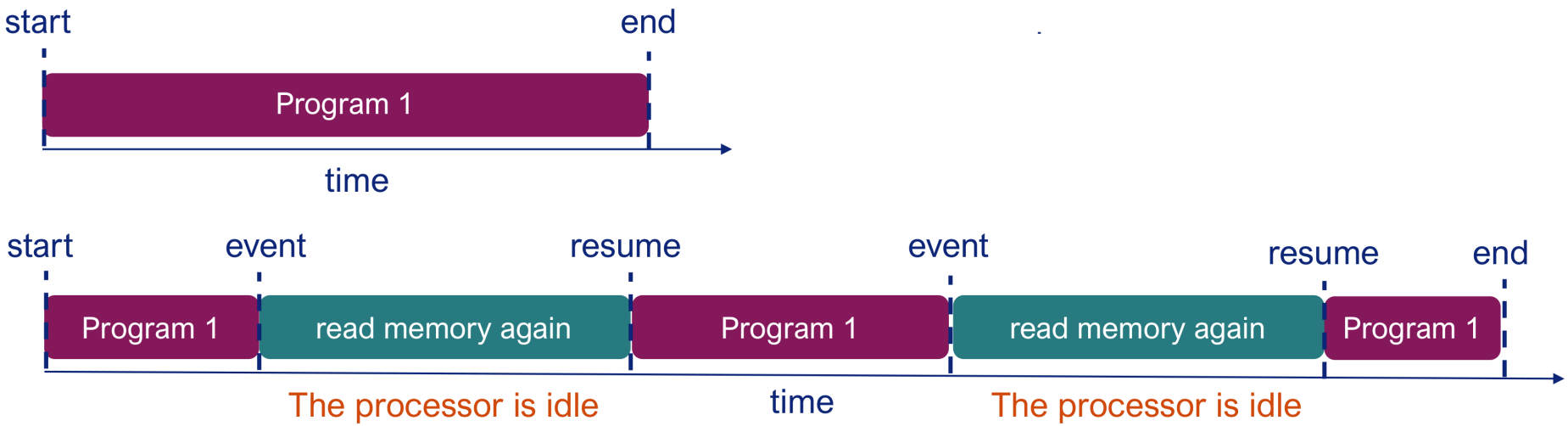
e.g., file management system calls open, read, write, close



- minimum two modes: user mode and kernel/supervisor/privileged mode
- in hardware: mode bit

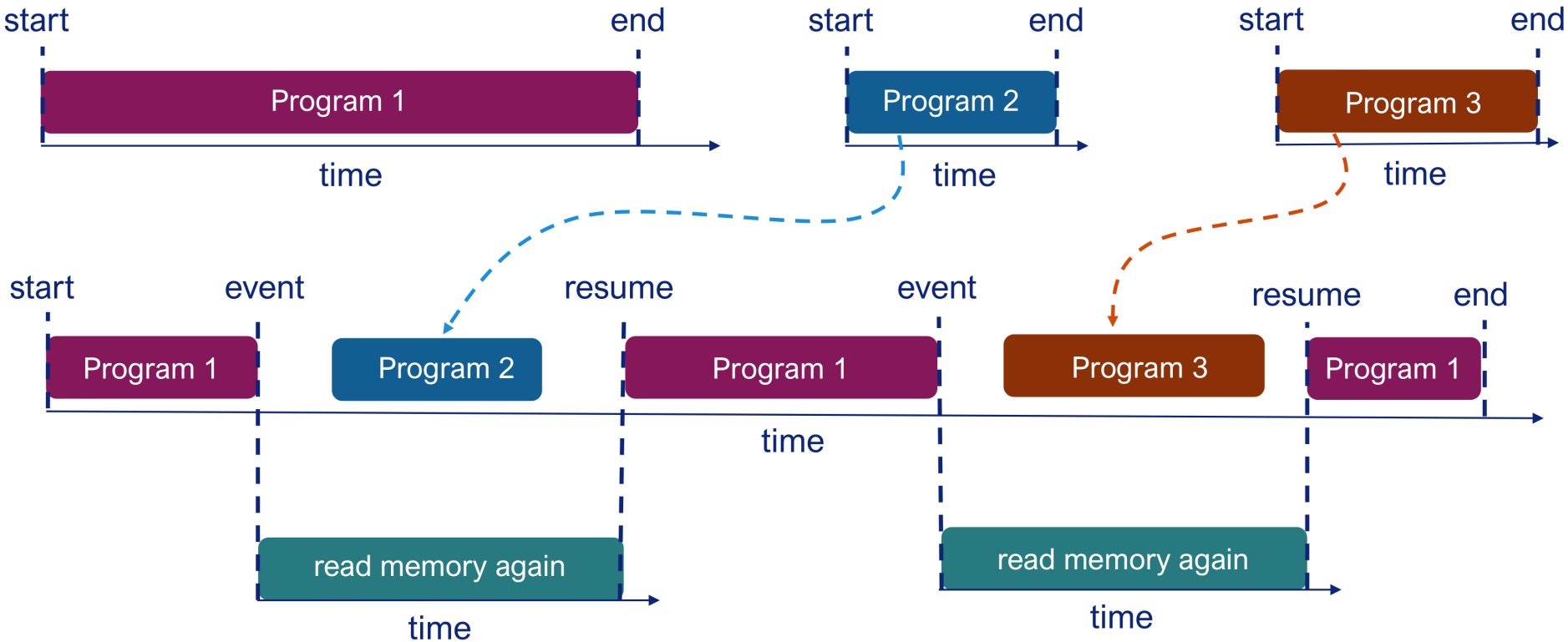
- System calls allow applications or user-level processes to request services (that require privileged access to the system's resources) from the operating system's kernel
- examples: files management, memory allocation, creating processes, IO control
- transition from user mode to kernel mode occurs during a system call

Multiprogramming



- ideal case: CPU executes program 1 without interruptions (events)
- reality: events like accessing main memory stall the execution → CPU is idle
  ⇒ how can OS minimize the idle time ?

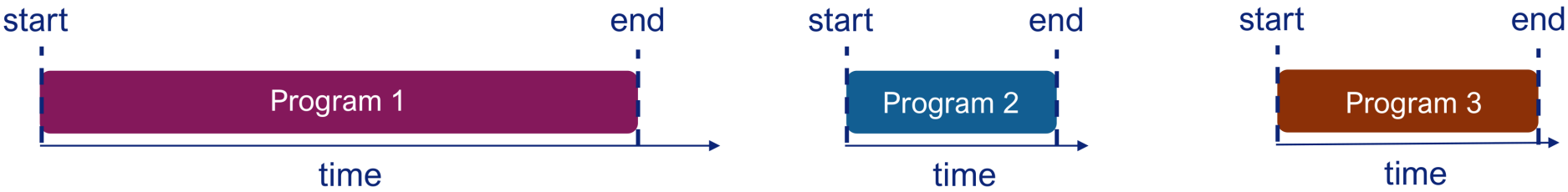
Multiprogramming - how can OS minimize the idle time?

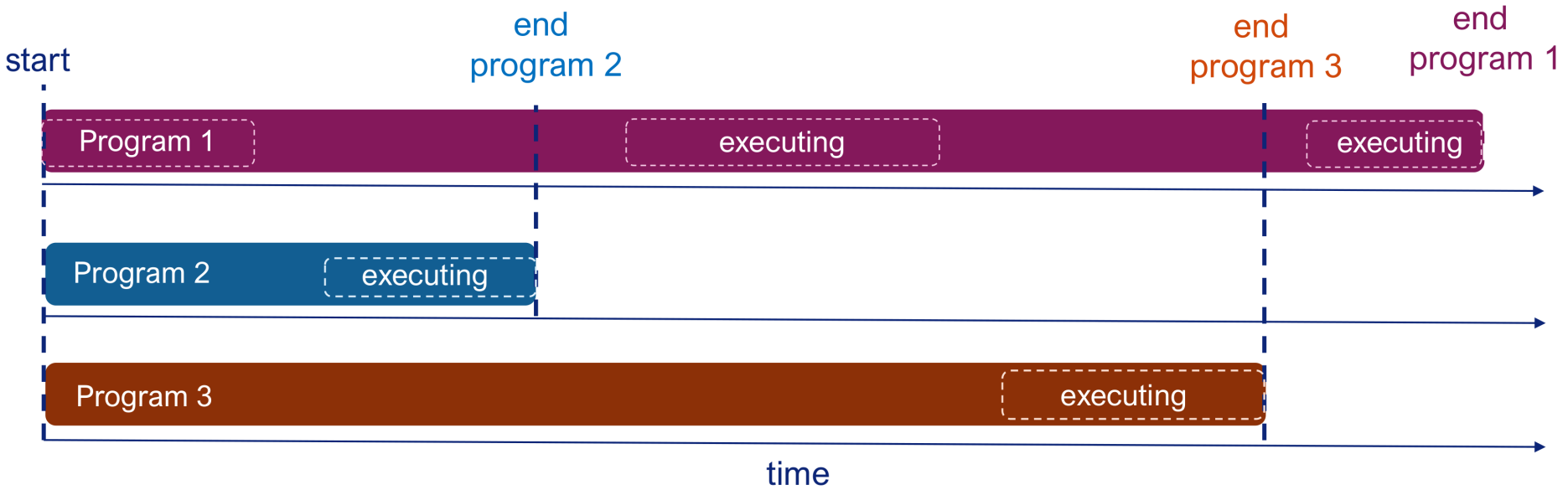


- we need more then one program in main memory at the same time[5]
- we need scheduling
- this creates the illusion of parallelism

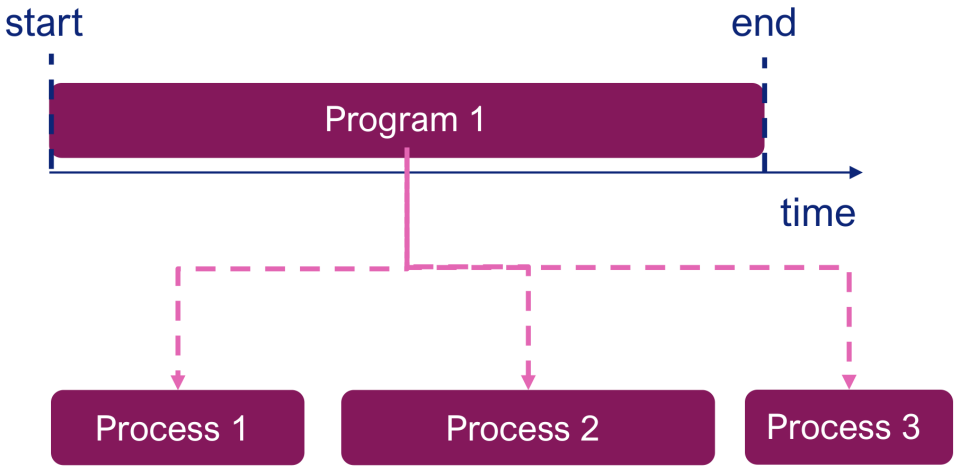
[5] CPU can only directly access the main memory

Multiprogramming - illusion of parallelism



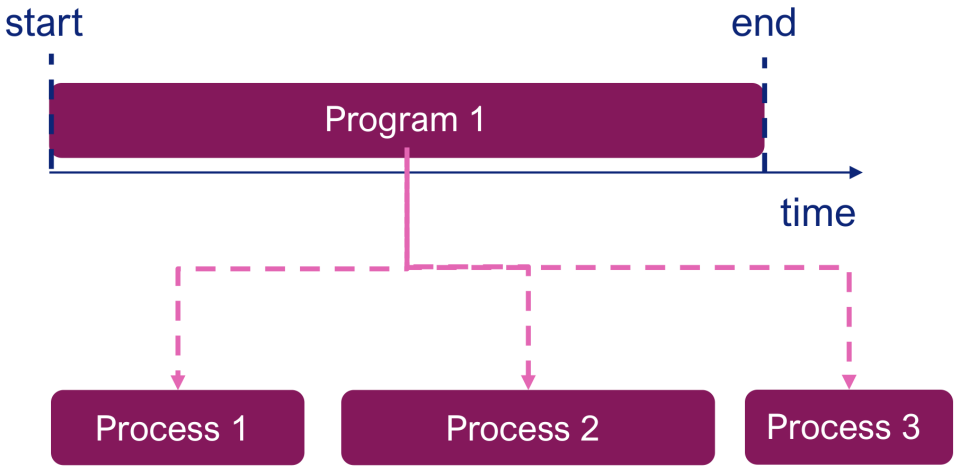


Multitasking



- recall: process is (a part of) a program in execution
- Program 1 as 3 processes (with different exec. times!)
- Switching between processes is faster then switching between "programs"
- timesharing approach

Multitasking

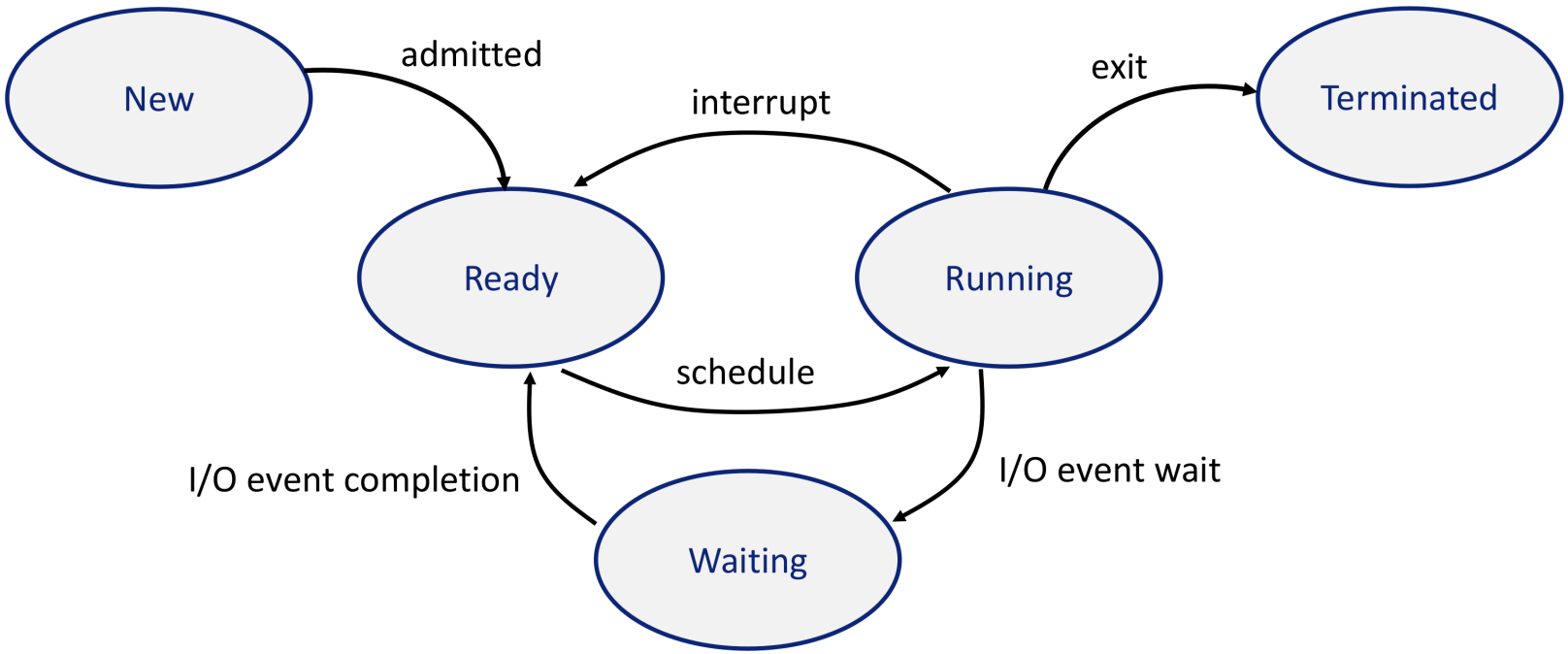


- recall: process is (a part of) a program in execution
- Program 1 as 3 processes (with different exec. times!)
- switching between processes is faster then switching between "programs"
- timesharing approach

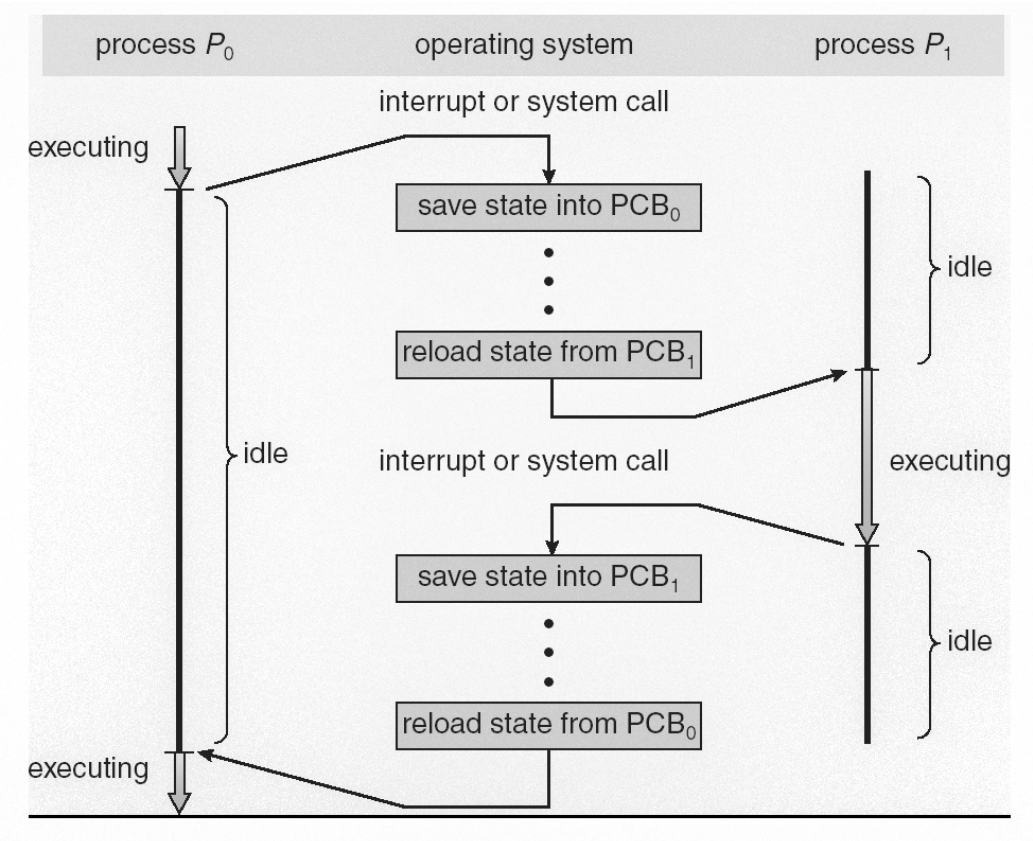
How OS manages processes

- process: insns and data
- the OS keeps a *process control block (PCB)* for each process
- PCB includes sufficient information for a process to be interrupted and resumed
  - process state (new, running, waiting, ready, terminated)
  - process number
  - PC (new insn to be executed)
  - CPU registers (depends on computer architecture)
  - memory limits (OS will allow access only if process within its limits)
  - I/O status (list of open files, allocated I/O devices, . . .)
  - Accounting information (processor time, time limits, . . .)
- Switching between processes is faster then switching between "programs"
- timesharing approach

How OS manages processes

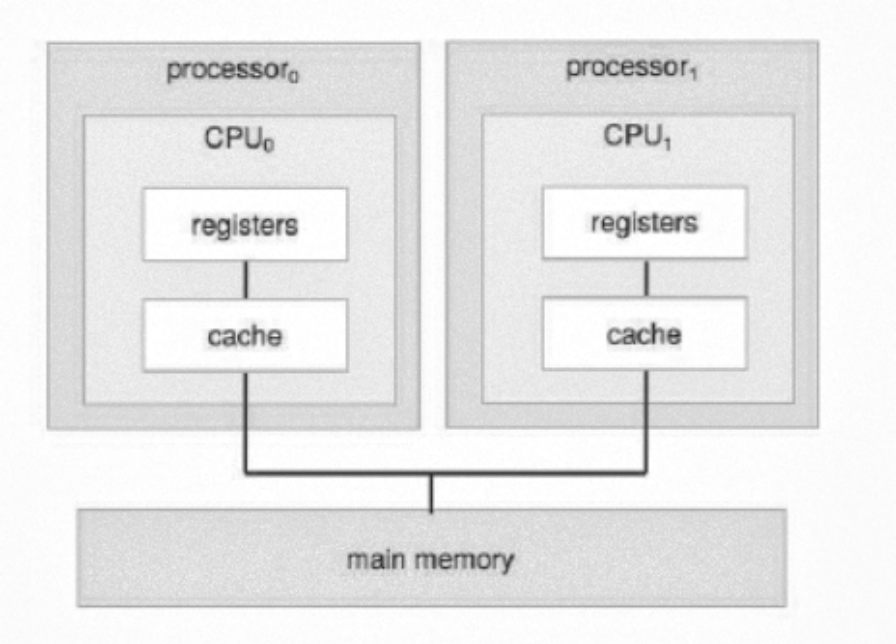


How OS manages processes

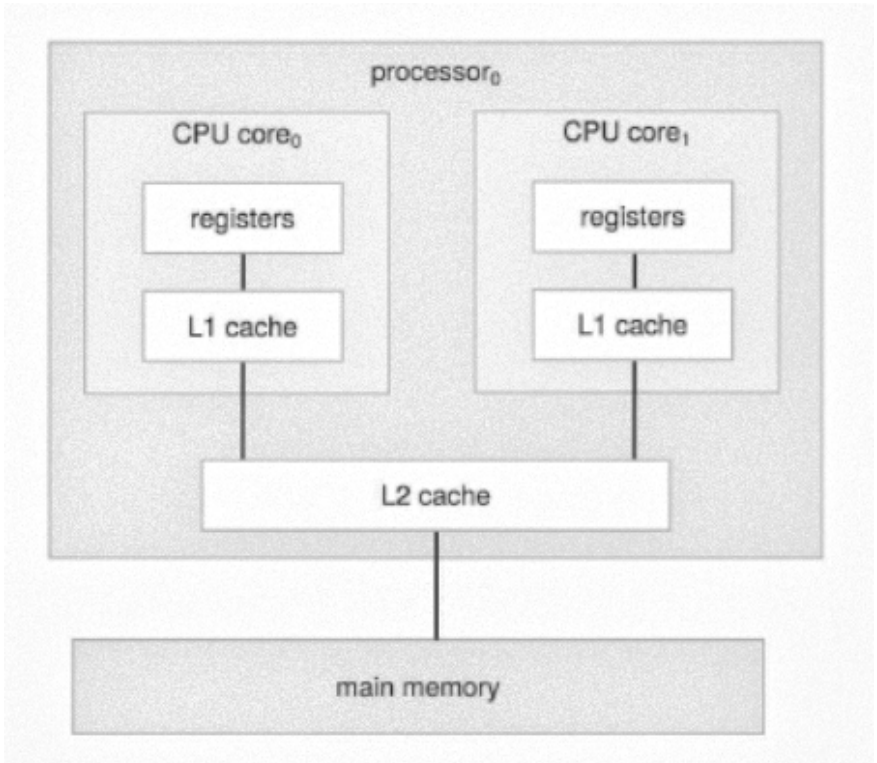


Some basic definitions

Symmetric multiprocessing architecture

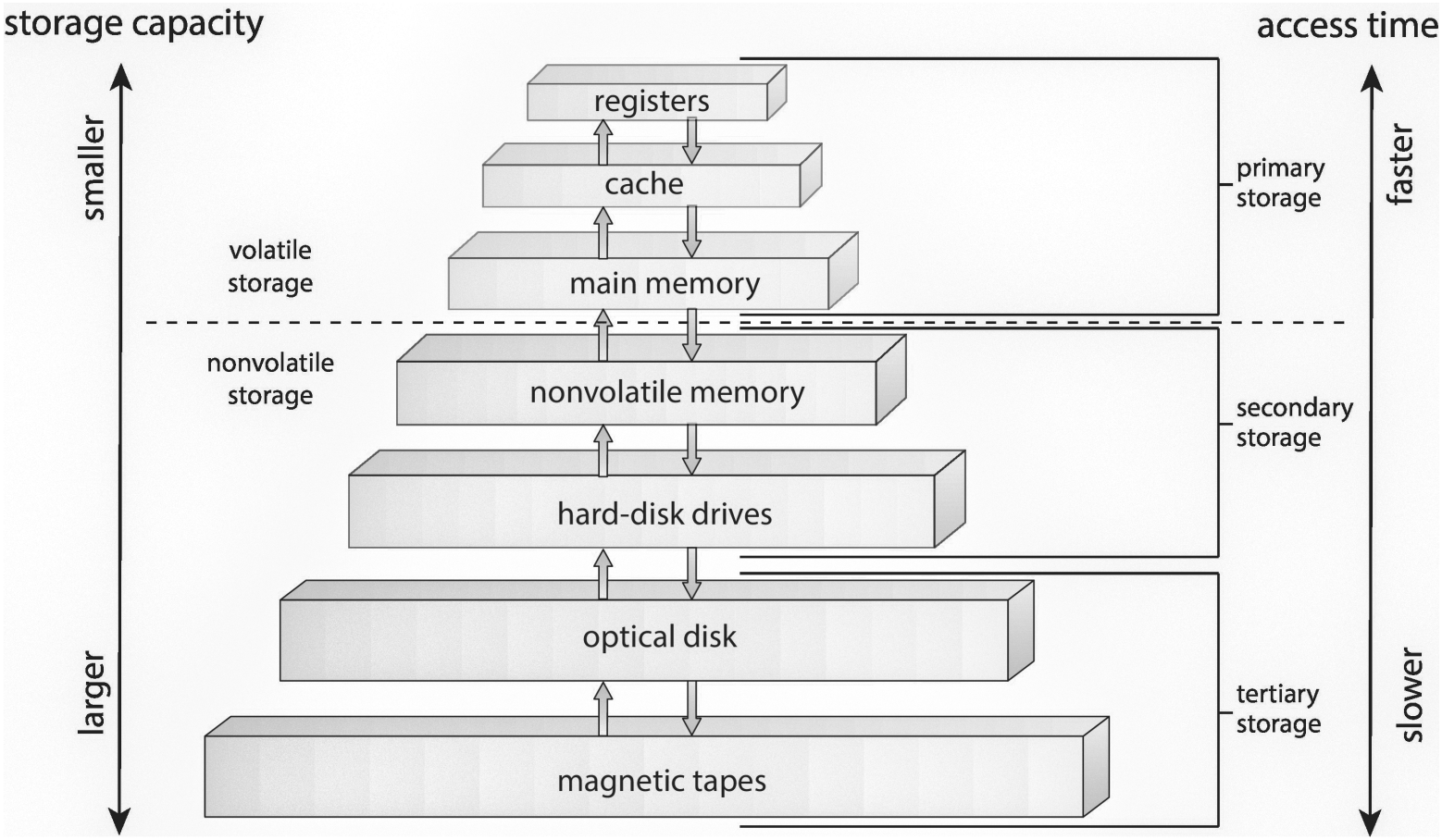


Multicore system



modern architectures: multiprocessor systems in which each CPU contains several computing cores

Some basic definitions



NOTE: figures from Silberschatz et al. Operating System Concepts and from slides by G.Perin