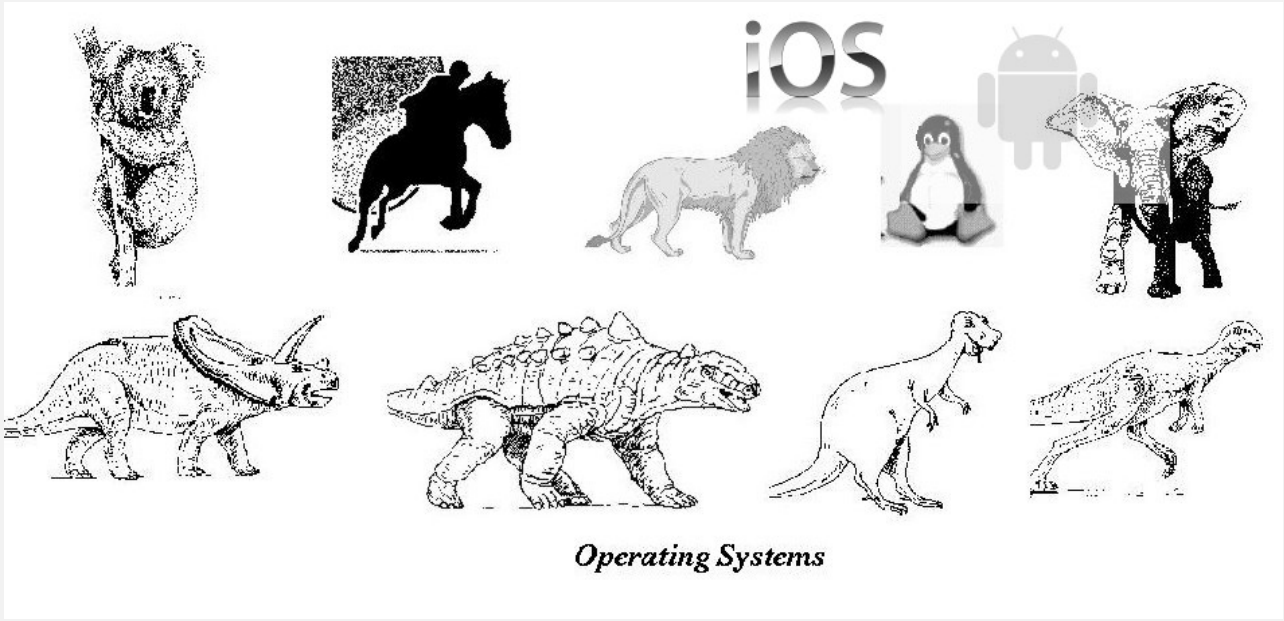
CS 453: Operating Systems: Introduction



Learning Objectives

- What is an Operating System (OS) and why is it needed?
- Understand the high-level organization of an OS.
- Introduce the concepts of a kernel, privilege levels and system calls used to implement an OS.
- Introduce Monolithic versus Microkernel design for an OS.
- How to observe the behavior of an OS.

Introduction

- An *Operating System* is a system software that acts as an intermediary between:
  - user and resources (could be hardware or abstract)
  - application software and resources
  - other system software and resources
- Application software versus System software
- An operating system has *two* major functions:
  - resource abstraction
  - resource sharing

Resource Abstraction

- provide abstract models of hardware components
- a good abstraction (or interface) is general across resources, yet easy to use
- abstraction can be carried out at several levels

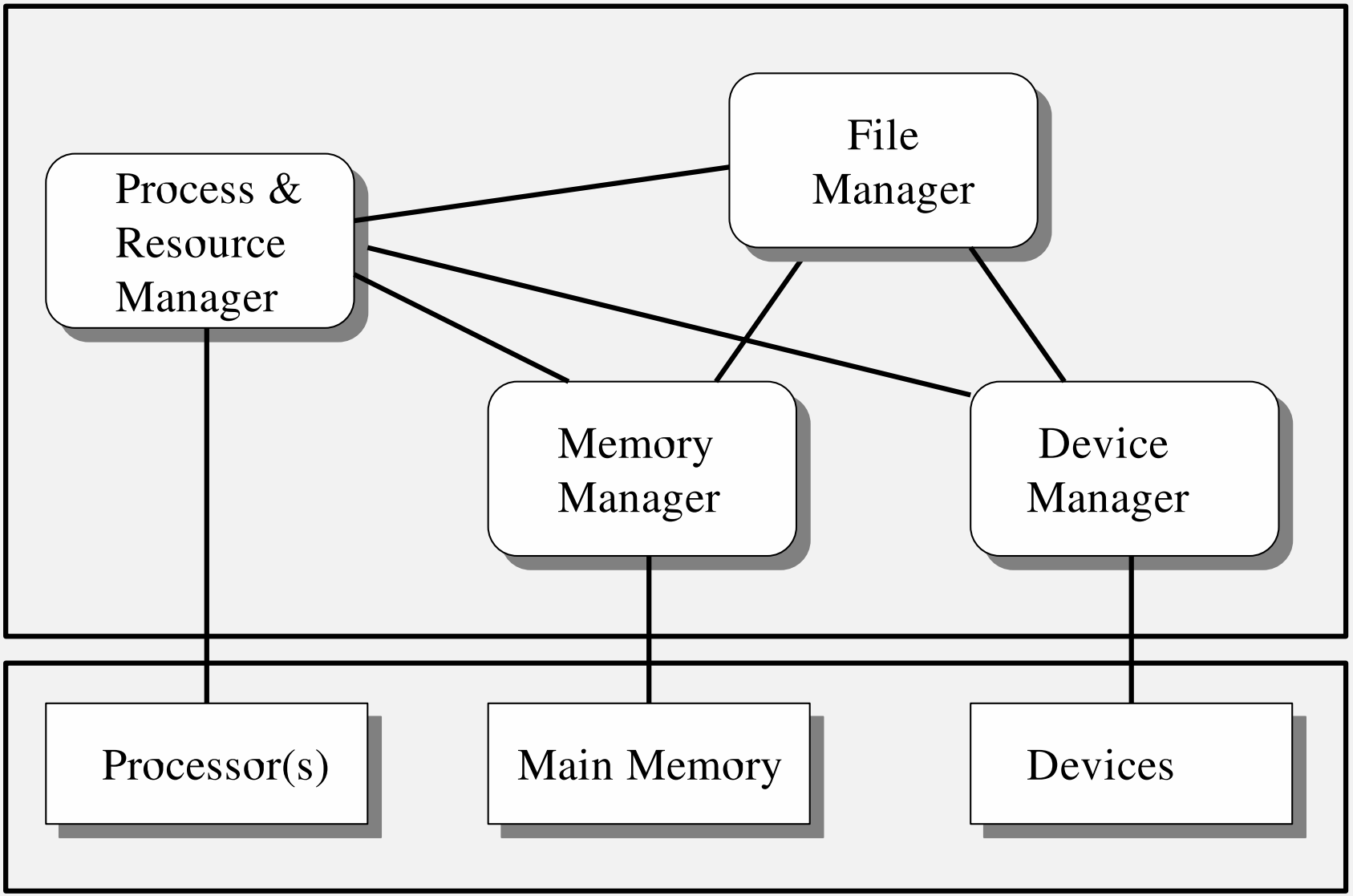
Resource Sharing

- Sharing can be of two types:

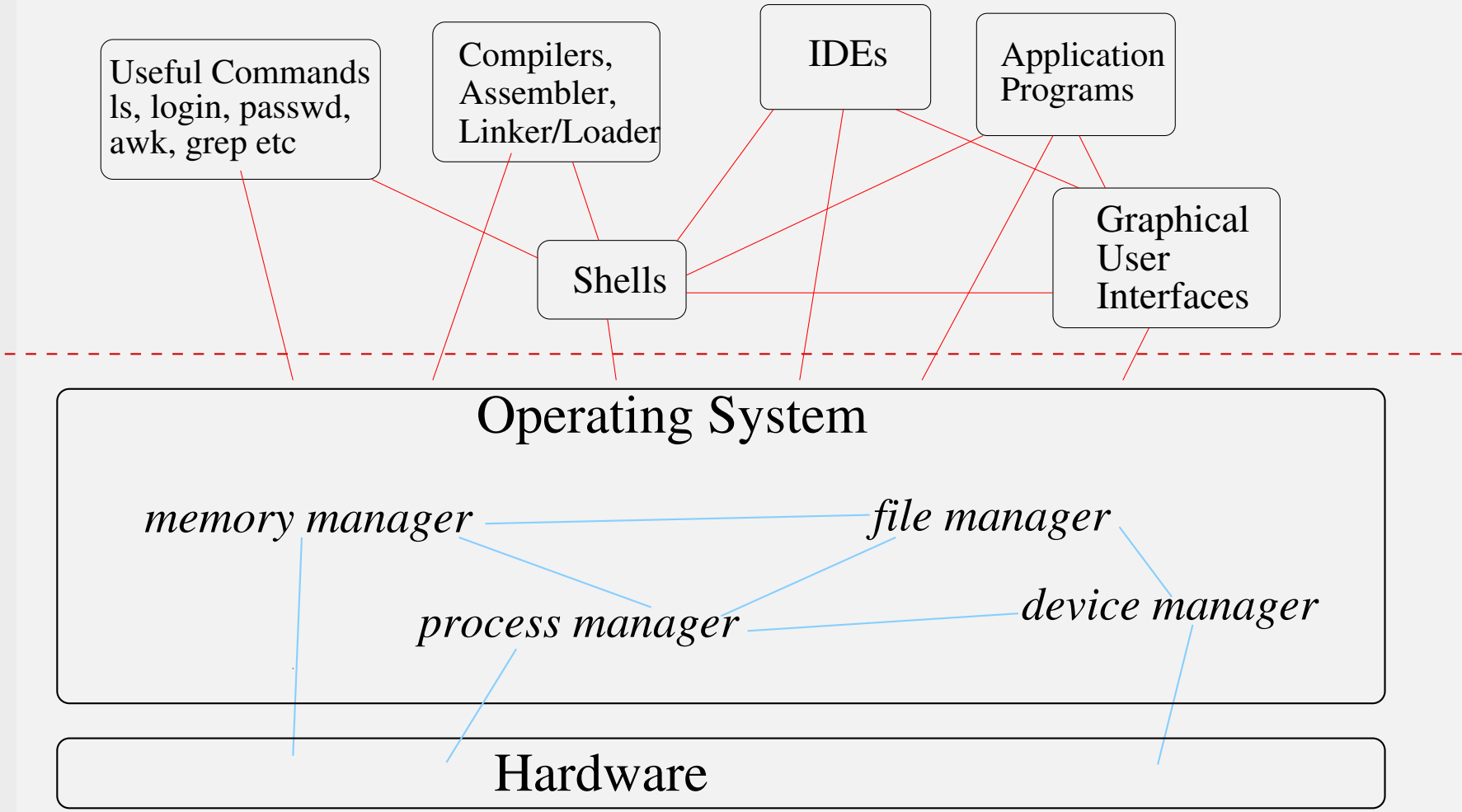
  Space-multiplexed : Resource can be divided into two or more units. e.g. memory, disks

  Time-multiplexed : Resource must be given exclusively. e.g. processor
- Should prevent unauthorized sharing while still allowing authorized sharing
- Resource isolation relies on the operating system being trustworthy. The operating system, in turn, relies on hardware for protection.

Logical Organization of an Operating System



The Big Picture



Why do we need an Operating System?

- Can your computer run without an Operating System?

- What would it take to use such a computer?

Linux Source Code

- Download the latest stable Linux source code from the "The Linux Kernel Archives." Or examine the code on onyx at `~amit/tmp/linux-4.9.3`
- How many files does it have? How many lines of code does it have?

- Can you identify folders corresponding to the major parts of the Operating System such as the kernel, memory management, process management, file system, device drivers.

Implementation Strategies/Issues

- The Operating System kernel as the trusted software module.
- Hardware provides supervisor mode versus user mode to provide protection.
- How do applications and users request services from the operating system?

Privilege Levels

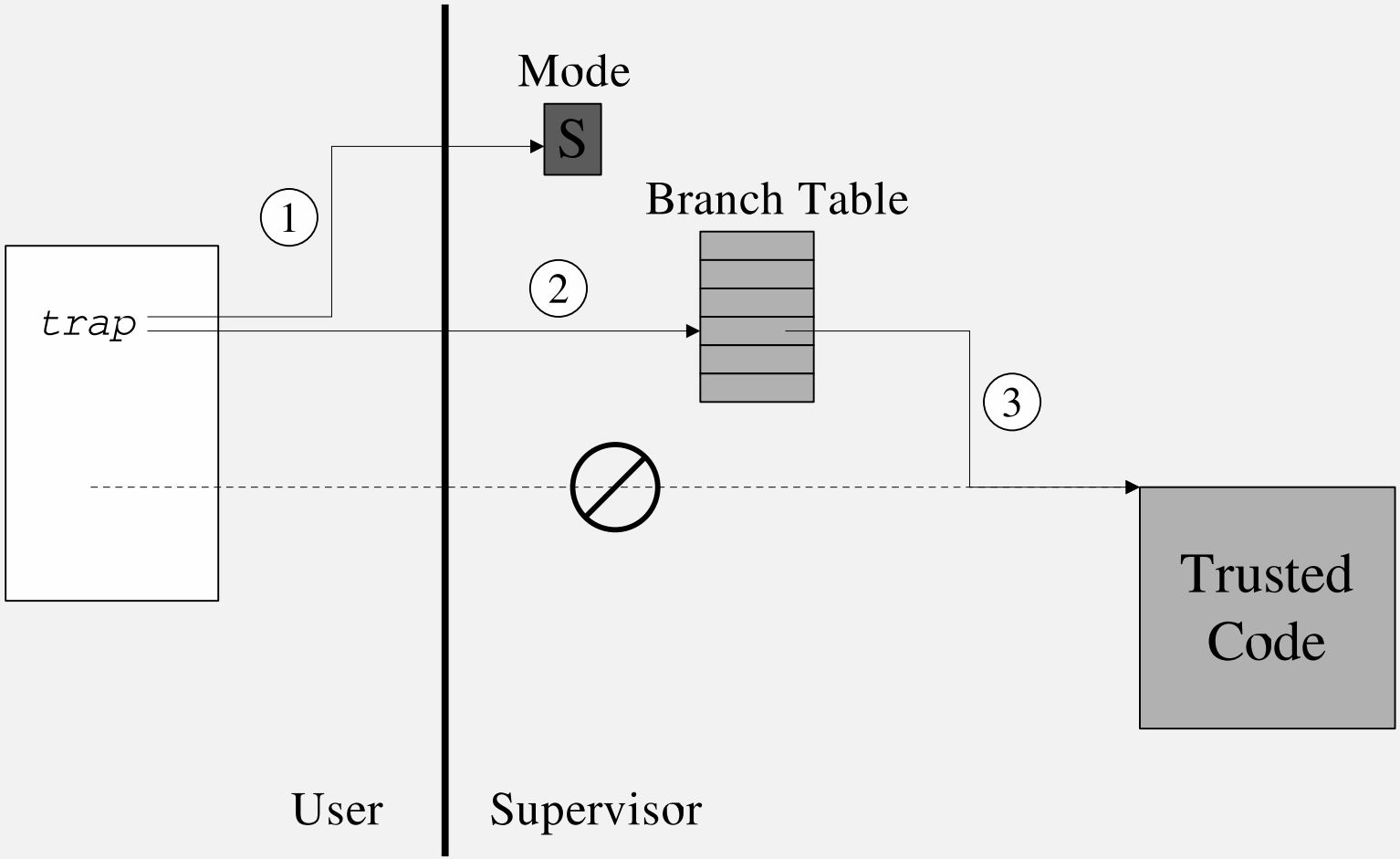
The mode bit is used to determine the privilege level by the hardware. The hardware must support at least two separate modes.

- Supervisor mode.
  - All machine instructions are available.
  - All memory addresses are available.
- User mode.
  - A subset of the instructions are available. Typically I/O instructions and instructions that change the mode bit are not available. Using those instructions causes the process to fault and stop.
  - A subset of the memory addresses are available.

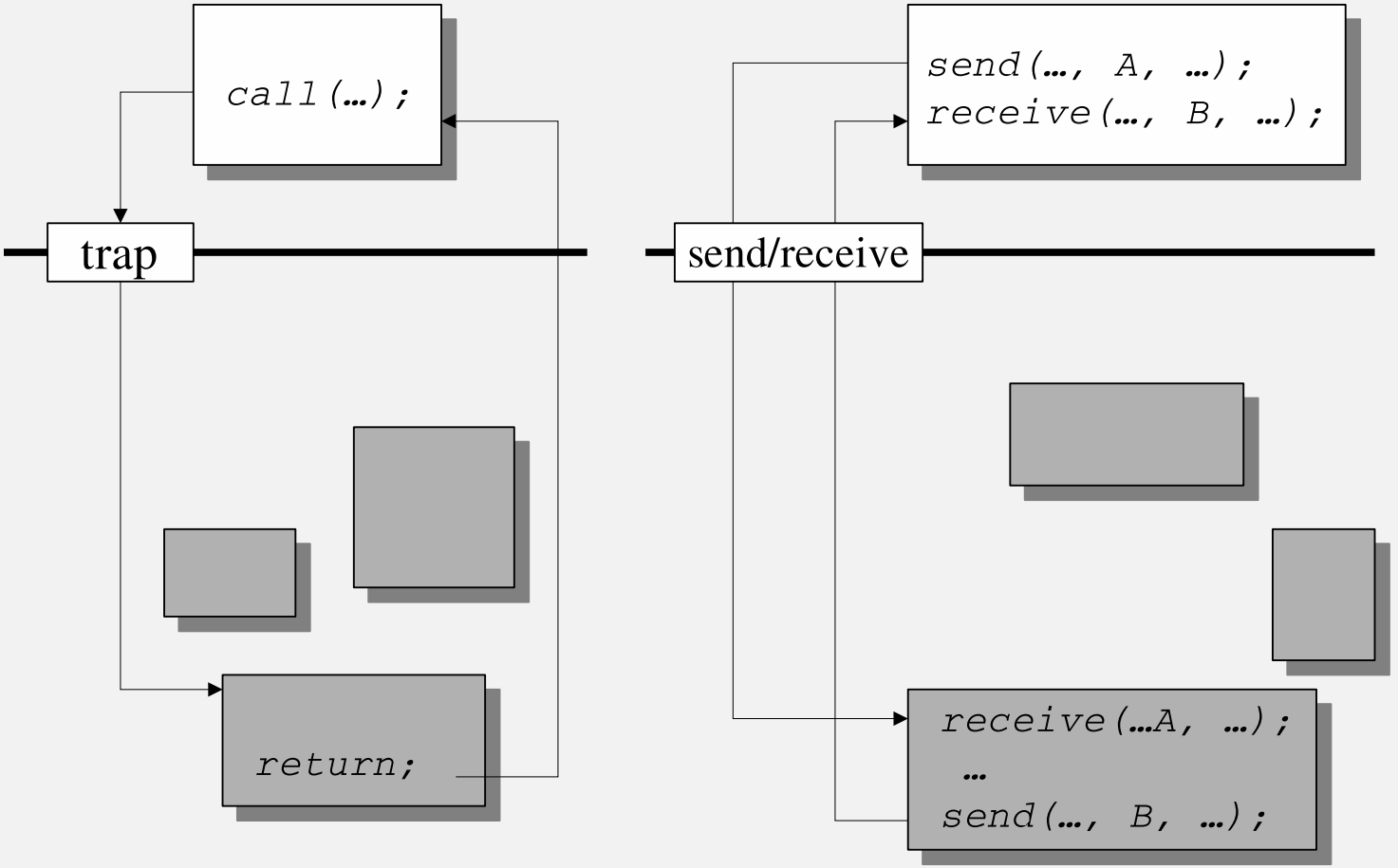
Kernel

- The **kernel** is the trusted part of the operating system. It runs in the supervisor mode.
- The *trap* machine instruction is used to switch from user mode to the supervisor mode. It is used to implement system calls.
- An alternative to *trap* instruction is using messages to get service from the operating system.

The *trap* Instruction



Requesting OS Service



In-class Exercise

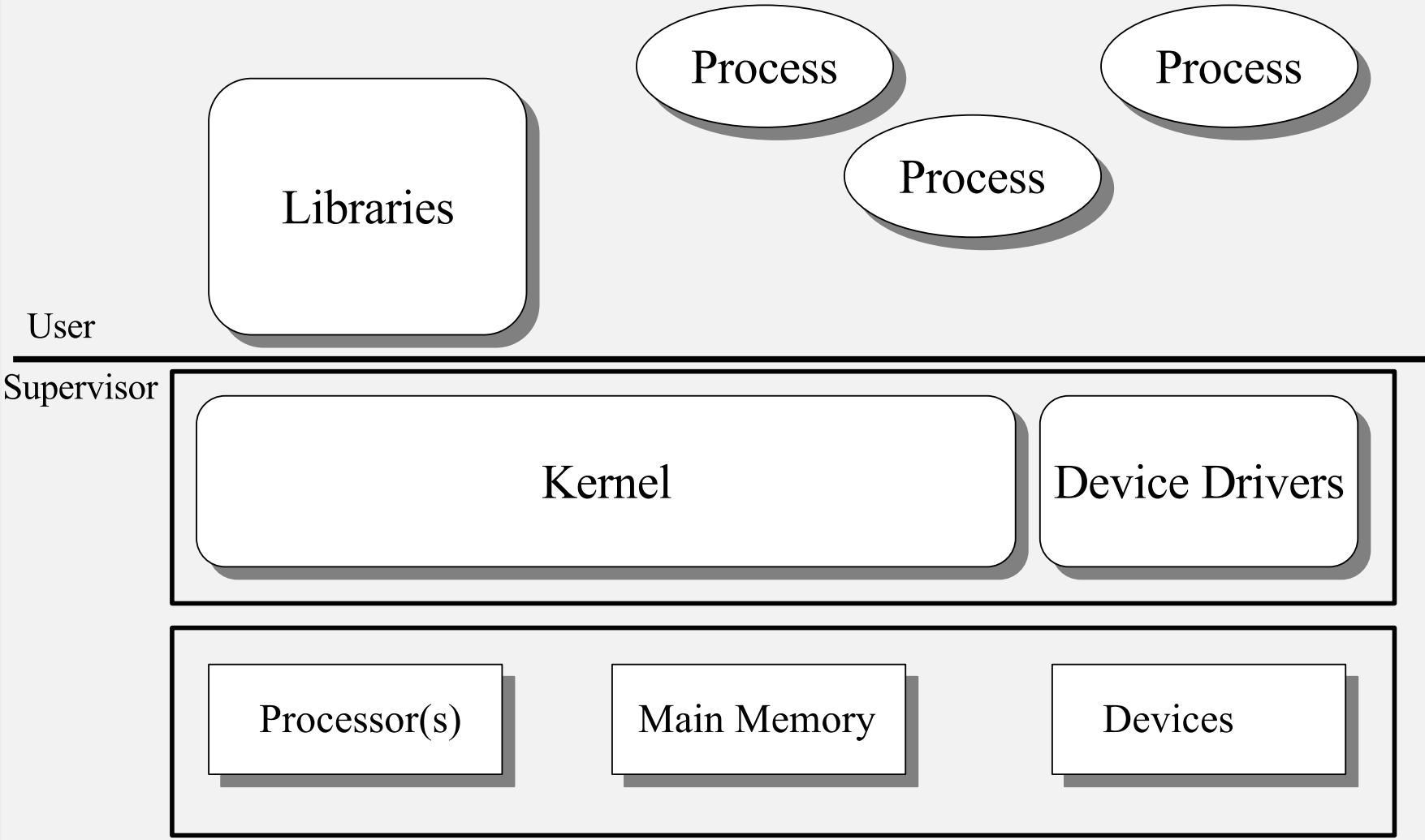
- In particular, see if you can locate the system call branch table in the Linux kernel code. *Hint: It will be architecture specific. Look for kernel code specific to an architecture, such as Intel x86. Use* `find` *and* `grep -r` *to help you!*

Operating System Kernel Design Options

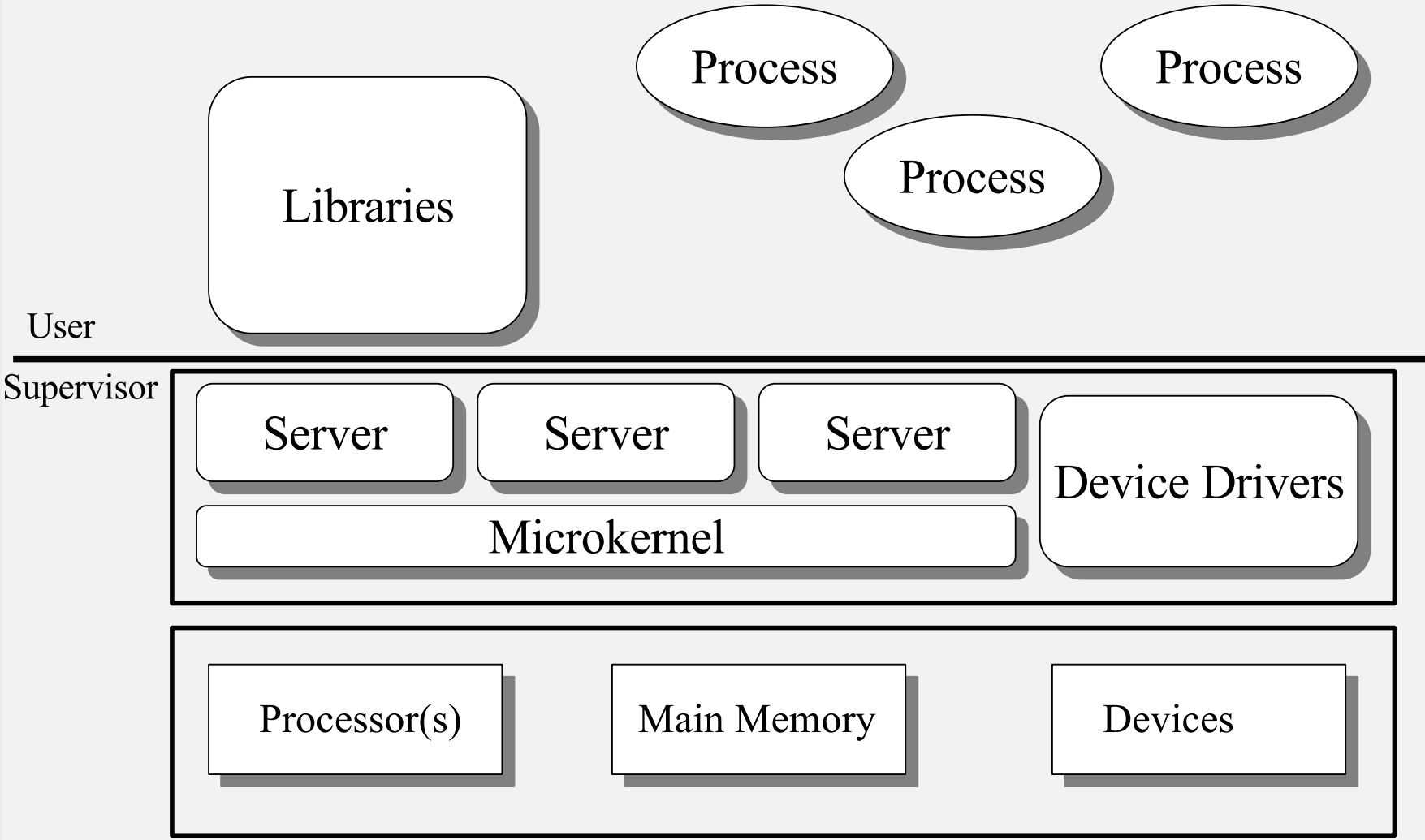
- **Monolithic**. There is a single large kernel that contains most of the operating system code. The device drivers can be separate.
  - UNIX was conventionally a monolithic design. Linux started as a monolithic kernel but with the increasing use of modules, the kernel can be made smaller and less monolithic.
- **Microkernel**. The kernel is very small and only implements some fundamental things like processes and scheduling. The operating system then consists of several subsystems along with the kernel.
  - MACH operating system is an example of a microkernel design. MS Windows NT was based on MACH. In turn MS Windows 2000, XP, Vista, 7, 8, and 10 are based on the NT design although they have moved somewhat away from the microkernel approach for performance reasons.

For more on the debate of monolithic versus microkernel approach, see the link on the class website under the *Links* section titled "Microkernel versus Monolithic kernel."

Typical Monolithic Kernel Organization



Typical Microkernel Organization



Factors in Operating System Design

- Performance
- Protection and Security
- Correctness
- Maintainability
- Commercial factors
- Standards and open systems. For example:
  - IEEE POSIX.1 open systems standard: specifies a set of system calls and their semantics that must be supported by a compliant operating system. (official standard)
  - TCP/IP: Internet network protocol. Supported by most operating systems. (de-facto standard)

Observing Operating System Behavior

- For Linux: Use ps, top or KDE System Guard ksysguard for user level tools. These tools are all based on the /proc virtual filesystem.
  - The /proc virtual filesystem under Linux provides a window into the internals of the operating system. For example:
    - /proc/cpuinfo gives us the details of the CPU in the system.
    - The file /proc/meminfo gives us the details of the memory in the system.
    - The file /proc/stat gives statistics about the system such as how long the system has been up, number of processes that have been created since the system was booted up etc.
    - The proc filesystem also has a folder for each process that is running. The folder contains relevant information that the OS keeps for that process.

    See man proc for more details on what all information is available.
- For Microsoft Windows: Use the *Task Manager*. For more detailed observations, see tools from http://www.sysinternals.com.