University of Pittsburgh

Introduction to Operating Systems
CS 1550

Spring 2023

Sherif Khattab

ksm73@pitt.edu

(Some slides are from **Silberschatz, Galvin and Gagne ©2013)**

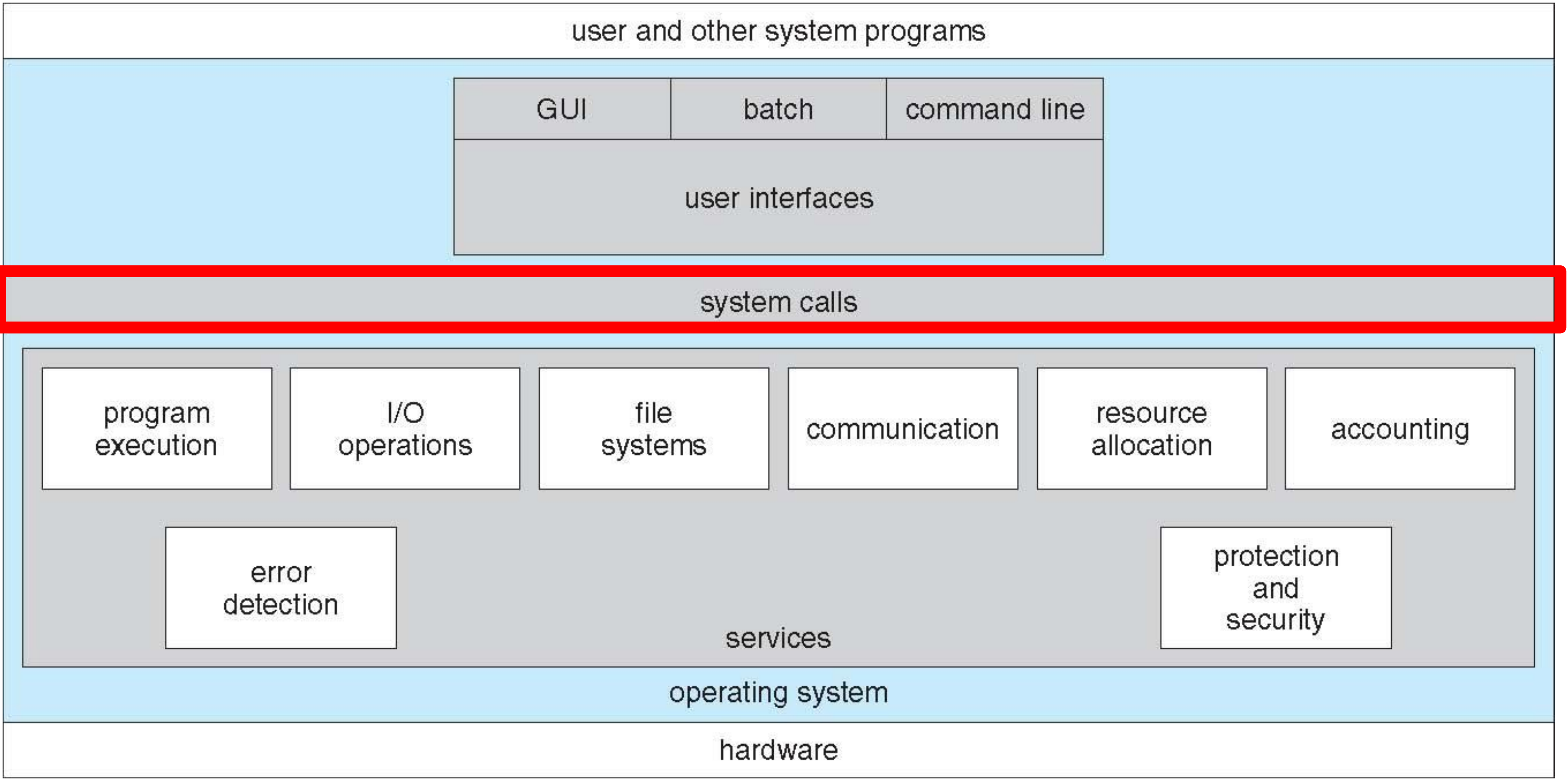
Announcements

- Lab 0 due next Friday
  - soft deadline; not graded; no deliverables
- Homework 1 will be posted on Canvas this Friday
  - 3 attempts
  - A practice homework with unlimited attempts
    - Special for Homework 1
- Recitations start next week
- VS Code setup tutorial on Piazza
  - (also linked from Canvas)
- Draft Slides repo linked from Canvas

Agenda

- Main tasks of an operating system
- System Calls
  - What an interrupt is
  - What happens when an interrupt occurs
  - What a system call is
  - How system calls implemented
  - Effect of OS structure on system calls

System Calls



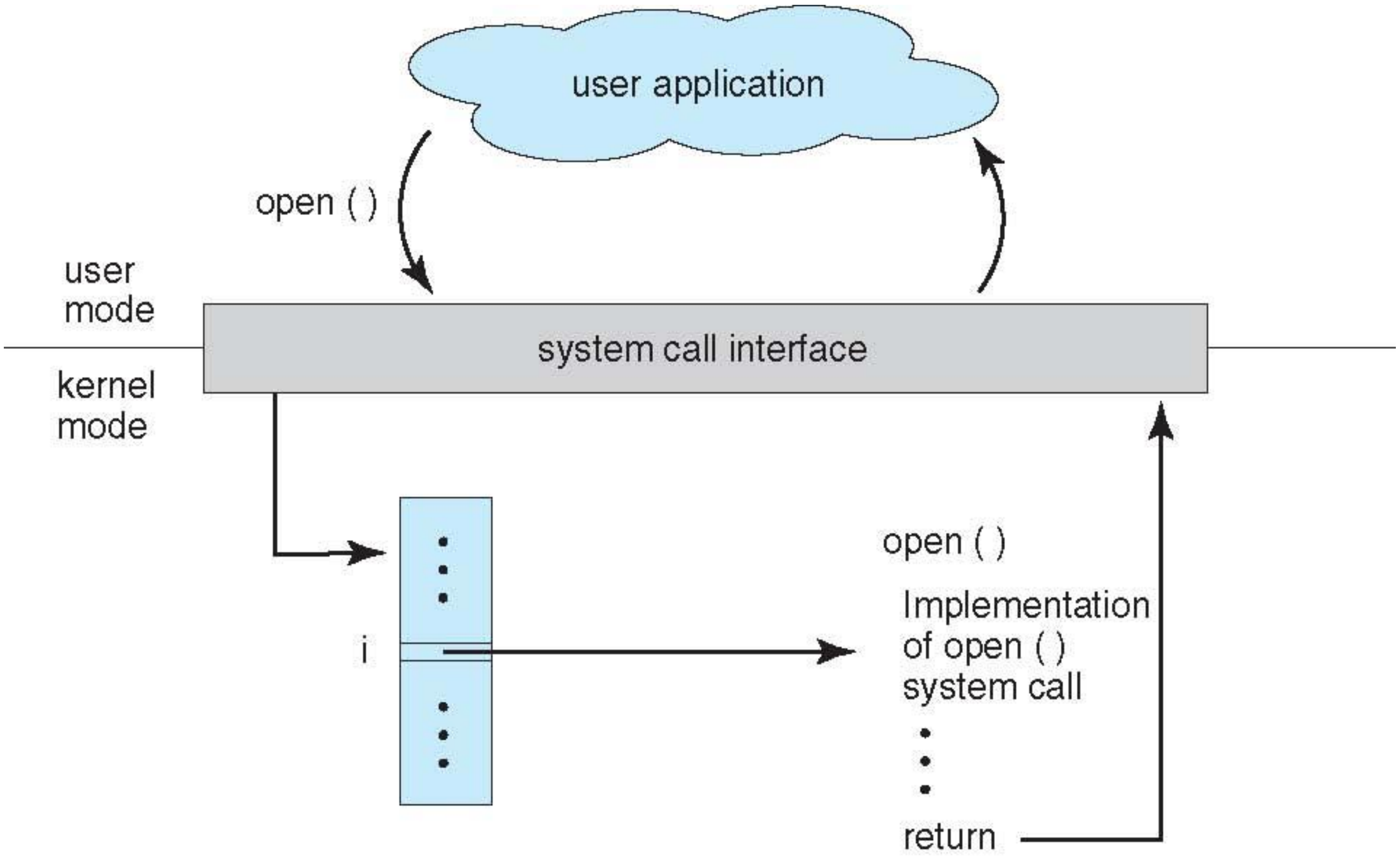
System Calls

- Programming interface to OS services
- Typically written in a high-level language (C or C++)
- Mostly accessed by programs via a high-level **Application Programming Interface (API)** rather than direct system call use
  - Win32 API for Windows
  - POSIX API for POSIX-based systems (including virtually all versions of UNIX, Linux, and Mac OS X), and
  - Java API for the Java virtual machine (JVM)

System Call Implementation

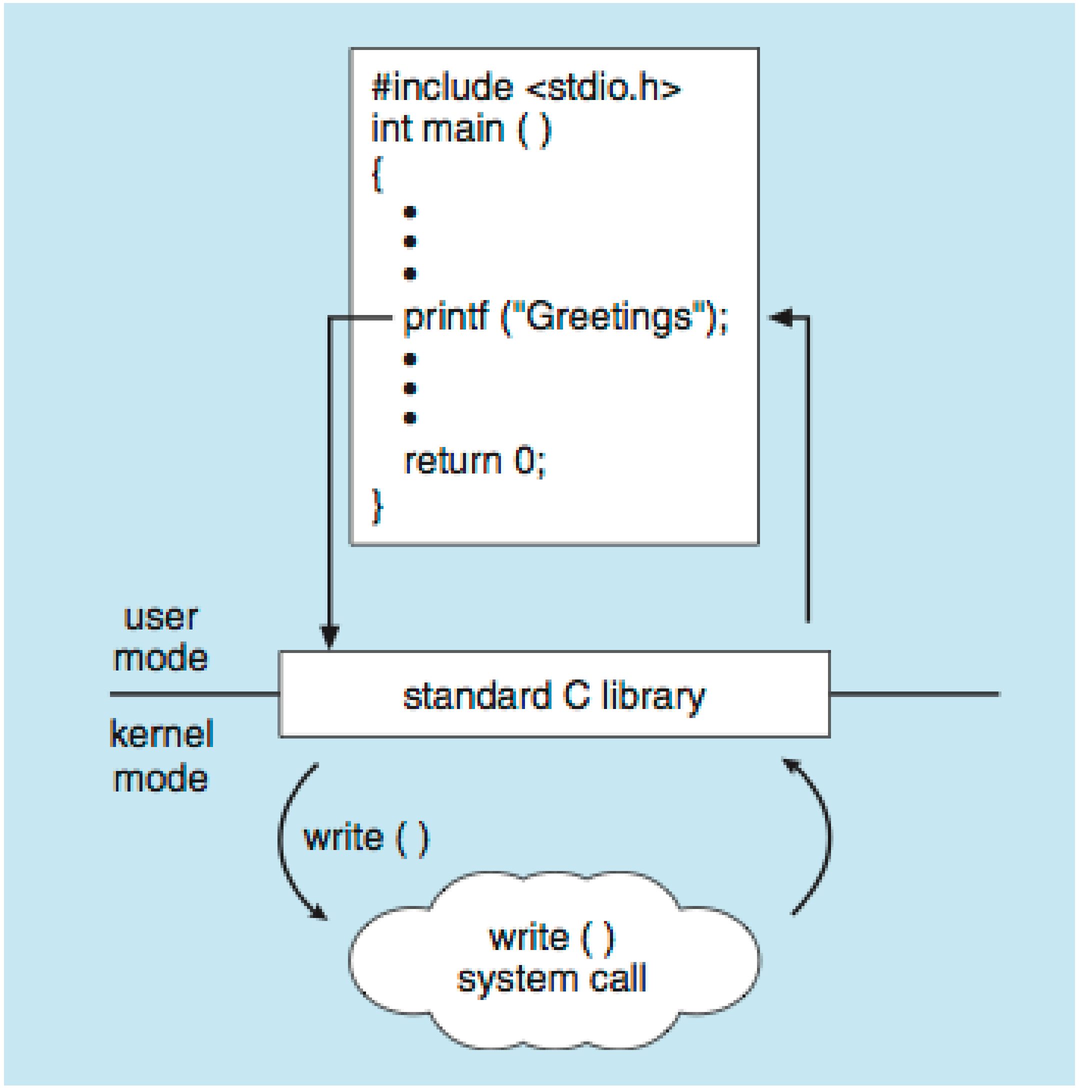
- Typically, there is a number associated with each system call
- Each system call has a corresponding system call implementation function (part of the OS kernel)
- **System-call table** indexed according to these numbers
  - Each entry in the table contains the address of the corresponding system call implementation function
- The system call interface is the ISR corresponding to the syscall software interrupt
  - invokes the intended system call in OS kernel,
  - passes arguments if needed, and
  - returns status of the system call and any return values
- The caller need know nothing about how the system call is implemented

API – System Call – OS Relationship

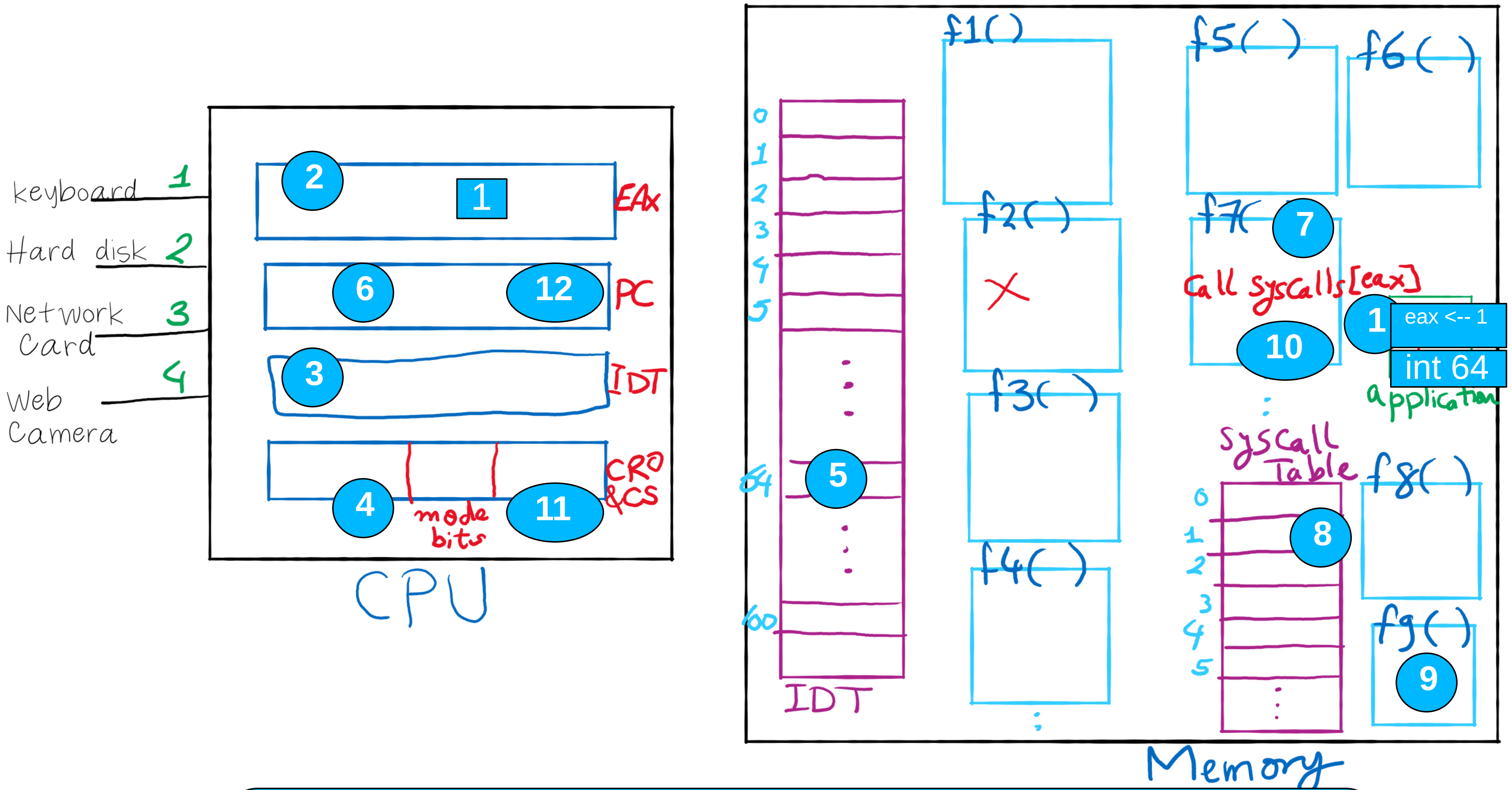


Standard C Library Example

- C program invoking printf() library call, which calls write() system call



What happens on a syscall?



Step 11-12: PC value restored from kernel stack

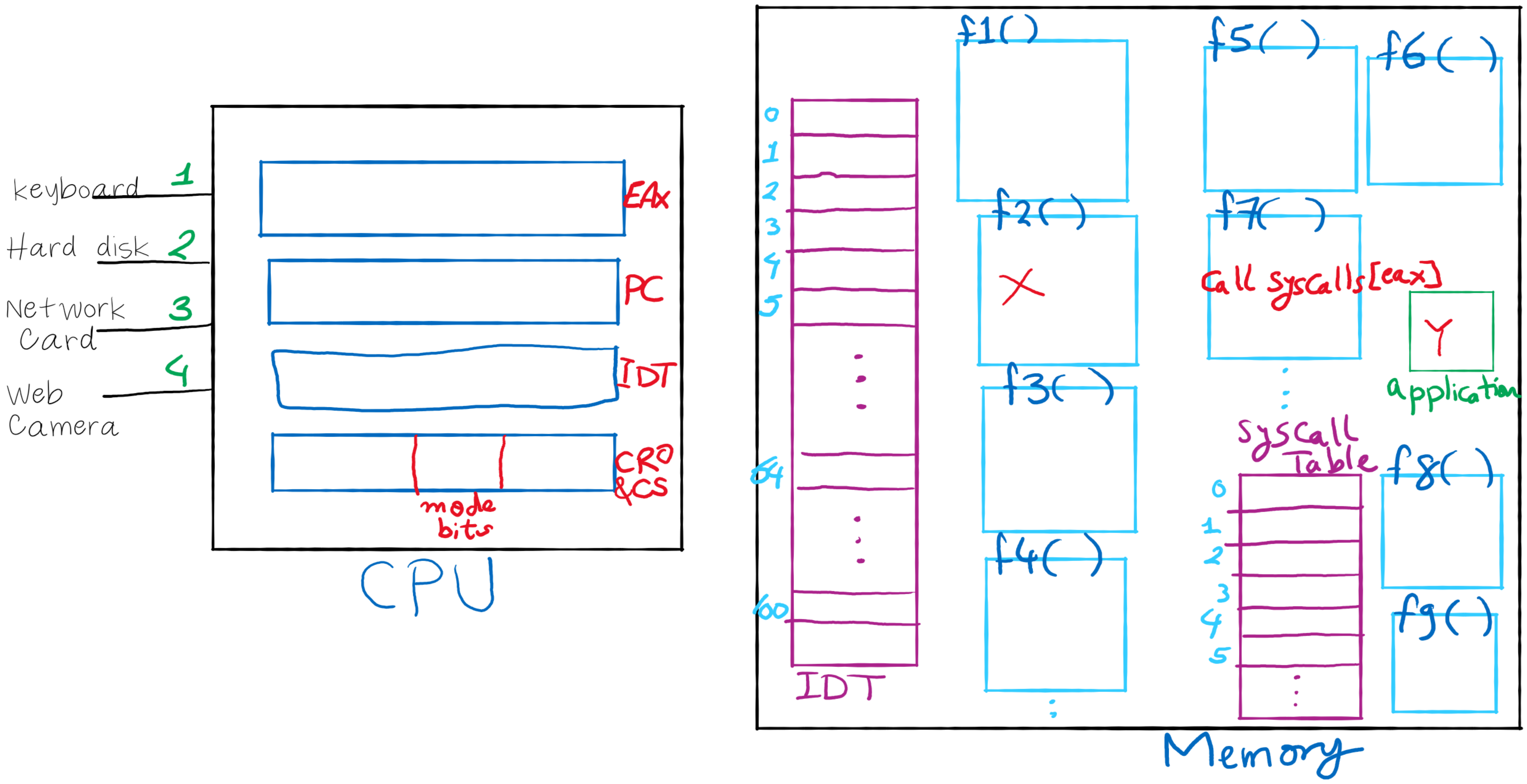
CPU switches back to user mode

Execution of the application resumes in user mode

System Call Parameter Passing

- Three general methods used to pass parameters to the OS
  - Simplest: pass the parameters in registers
    - In some cases, may be more parameters than registers
  - Parameters stored in a block, or table, in memory, and address of block passed as a parameter in a register
    - This approach taken by Linux and Solaris
  - Parameters placed, or **pushed**, onto the **stack** by the program and **popped** off the stack by the operating system
    - XV6
- Block and stack methods do not limit the number or length of parameters being passed

How to add a system call to an OS?

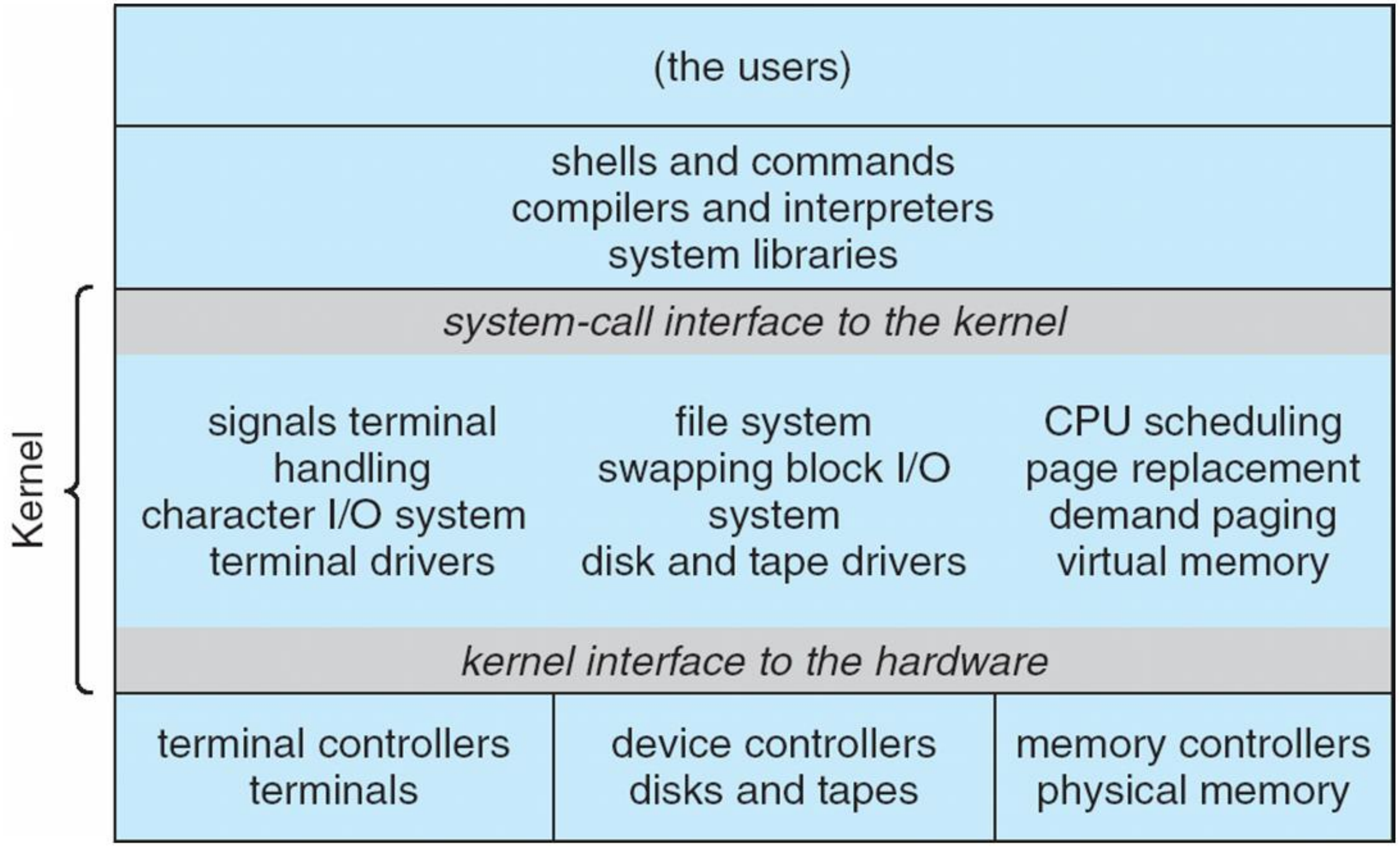


Xv6 Code Walkthrough

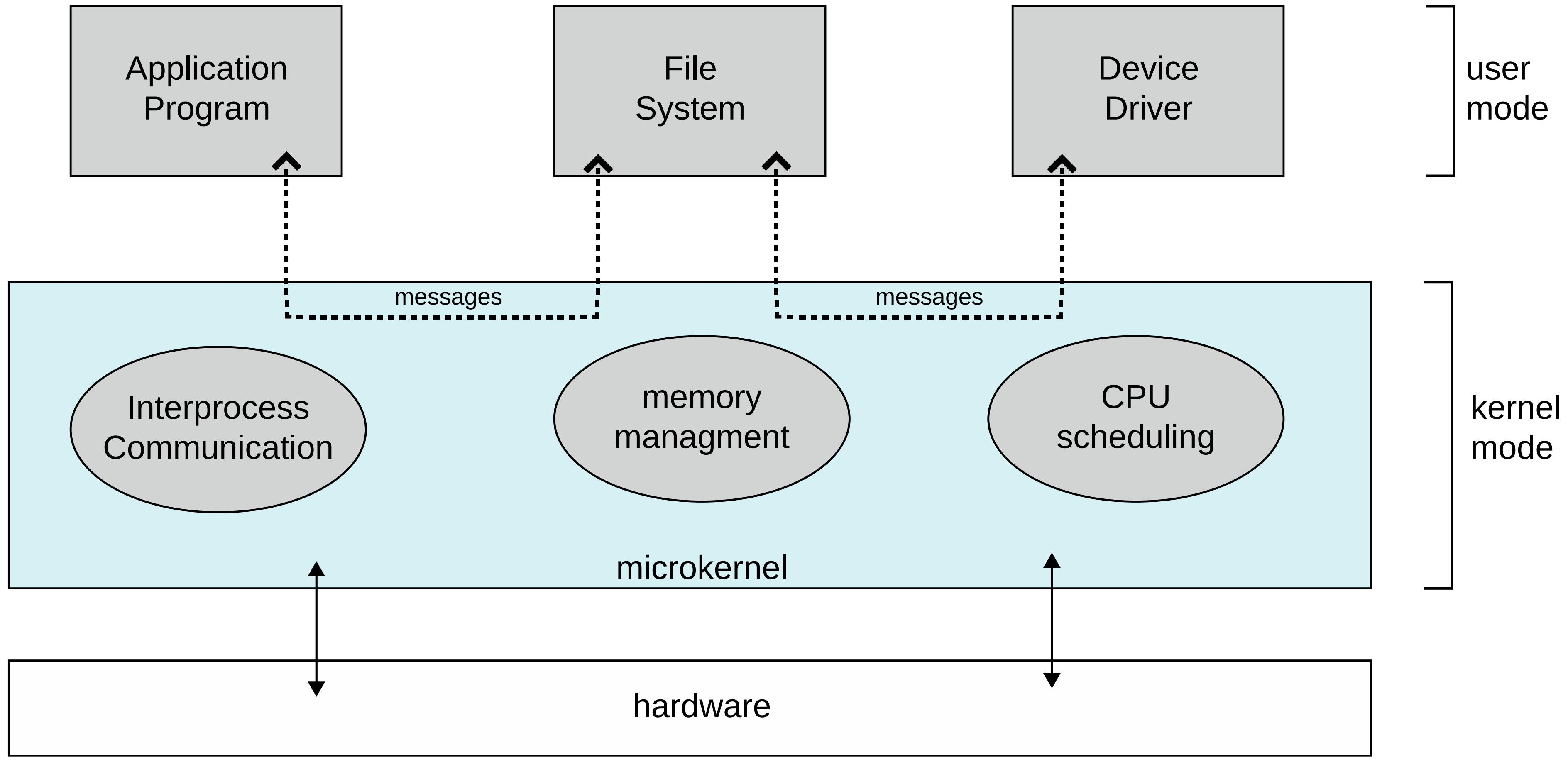
- IDT table initialization
- Syscall table
- How a syscall is invoked
- Syscall implementation
- Parameter passing into a syscall
- In Lab 1 you will add a system call to Xv6

Traditional UNIX System Structure

Beyond simple but not fully layered



Microkernel System Structure


Application Program
File System
Device Driver
user mode
messages
messages
Interprocess Communication
memory managment
CPU scheduling
microkernel
kernel mode
hardware