Lecture 2
OS Basics

Prof. Yinqian Zhang

Fall 2025

Outline

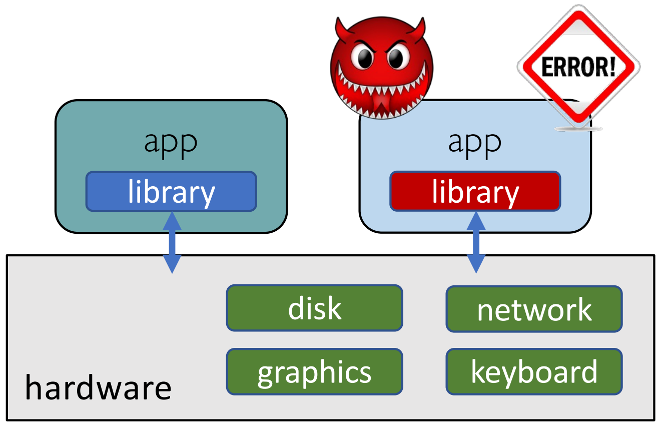
- Dual-mode operations
- Kernel structure
- Operating system services

Dual-mode Operations

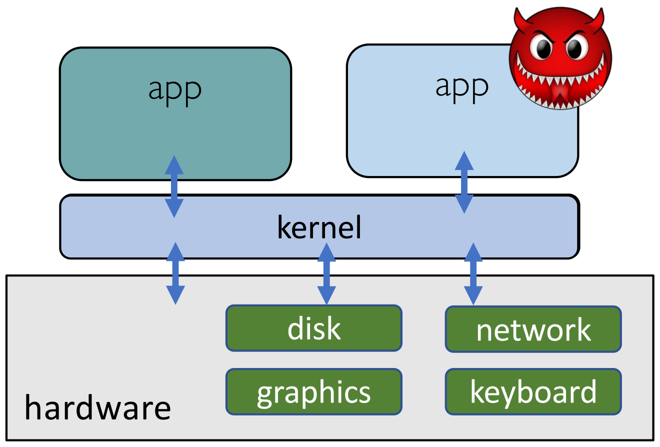
Evolution of Operating Systems

- A library to handle low-level I/O
  - Issue: Fault and security isolation
- Kernel: A bigger "library" to handle low-level I/O
  - Kernel needs to be protected from faulty/malicious apps

早期的程序通过 库 直接操作硬件（磁盘、显卡、键盘、网络等）



如果有恶意程序，它能直接操作硬件，毫无安全隔离。



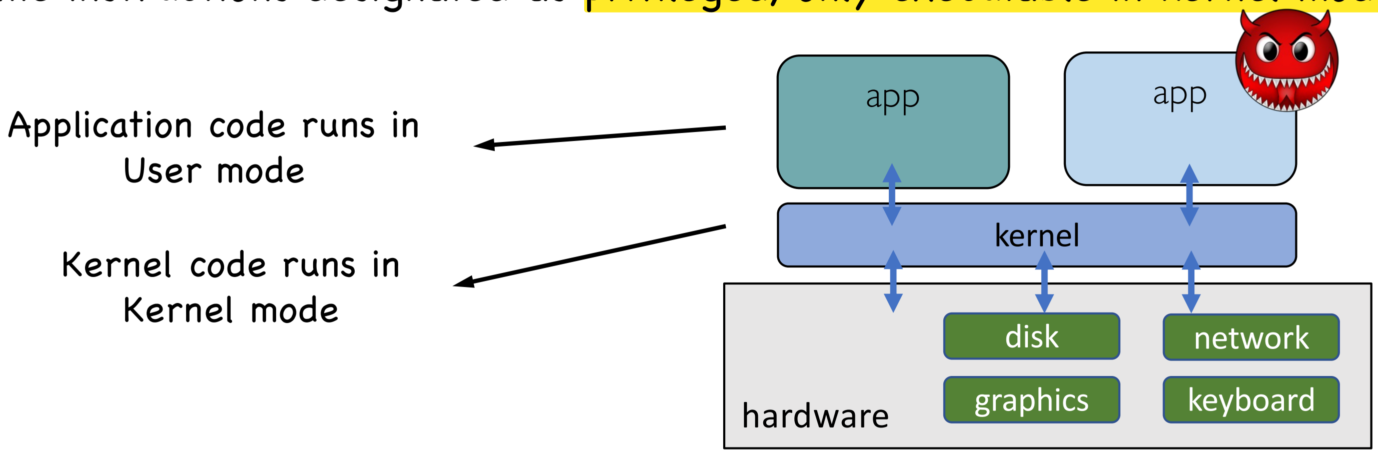
内核作为中间层，提供 保护和隔离。

即便有恶意或错误的程序，硬件不会被轻易破坏

Kernel Mode vs. User Mode

操作系统需要保护自己和硬件资源，防止普通程序随意访问、破坏系统

- Dual-mode operation allows OS to protect itself and other system components
  - Mode bits provided by CPU hardware
    - Provides ability to distinguish when system is running user code or kernel code
    - Some instructions designated as privileged, only executable in kernel mode


Application code runs in User mode
Kernel code runs in Kernel mode
app
app
kernel
disk
network
graphics
keyboard
hardware

CPU 提供了 双模式 (dual-mode)：
・用户态 (User Mode)：应用程序运行的模式，权限受限。
・内核态 (Kernel Mode)：操作系统内核运行的模式，权限最高，可以直接操作硬件。

普通应用程序（app）运行在 用户态，只能通过调用系统调用 (system call) 请求操作系统服务。
系统调用会触发 从用户态切换到内核态，操作系统帮它执行需要高权限的操作（如访问磁盘、网络、显卡、键盘）。
执行完后再切回用户态，返回应用程序。

Dual-mode Operation

- Hardware provides at least two modes:
  - "Kernel" mode: Run kernel code
  - "User" mode: Normal programs executed
- What is needed in the hardware to support "dual mode" operation?
  - A bit for representing current mode (user/kernel mode bit)
  - Certain operations / actions only permitted in kernel mode
    - In user mode they fail or trap

某些操作（比如 I/O、修改页表、关/开中断）只能在内核态执行。

  - User → Kernel transition *sets* kernel mode AND saves the user PC
    - Operating system code carefully puts aside user state then performs the necessary operations
  - Kernel → User transition *clears* kernel mode AND restores appropriate user PC

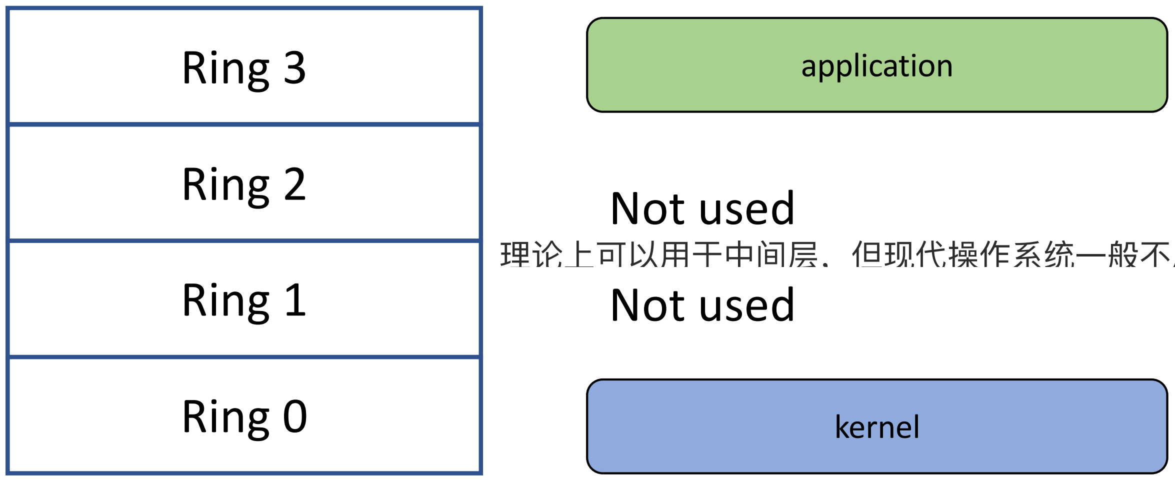
用户态 → 内核态 的切换

- 当用户程序需要执行特权操作时，会通过 系统调用 (system call) 触发切换。
- CPU 将模式位切换为内核态，并且 保存用户态的程序计数器 (PC)。
- 操作系统接管，执行相应的服务（比如访问磁盘）

内核态 → 用户态 的切换

- 内核完成操作后，CPU 清除模式位（回到用户态），恢复保存的用户 PC。
- 程序继续从切换前的位置运行

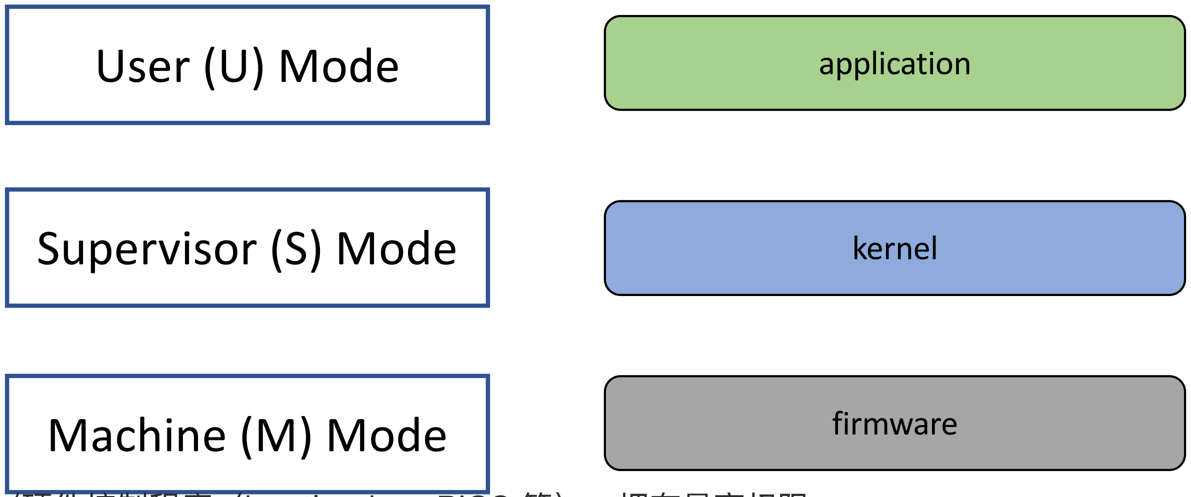
Mode Bits in CPUs



理论上可以用于中间层，但现代操作系统一般不用，只保留 Ring 0 和 Ring 3，

x86
(Intel & AMD)

Mode Bits in CPUs (Cont’d)

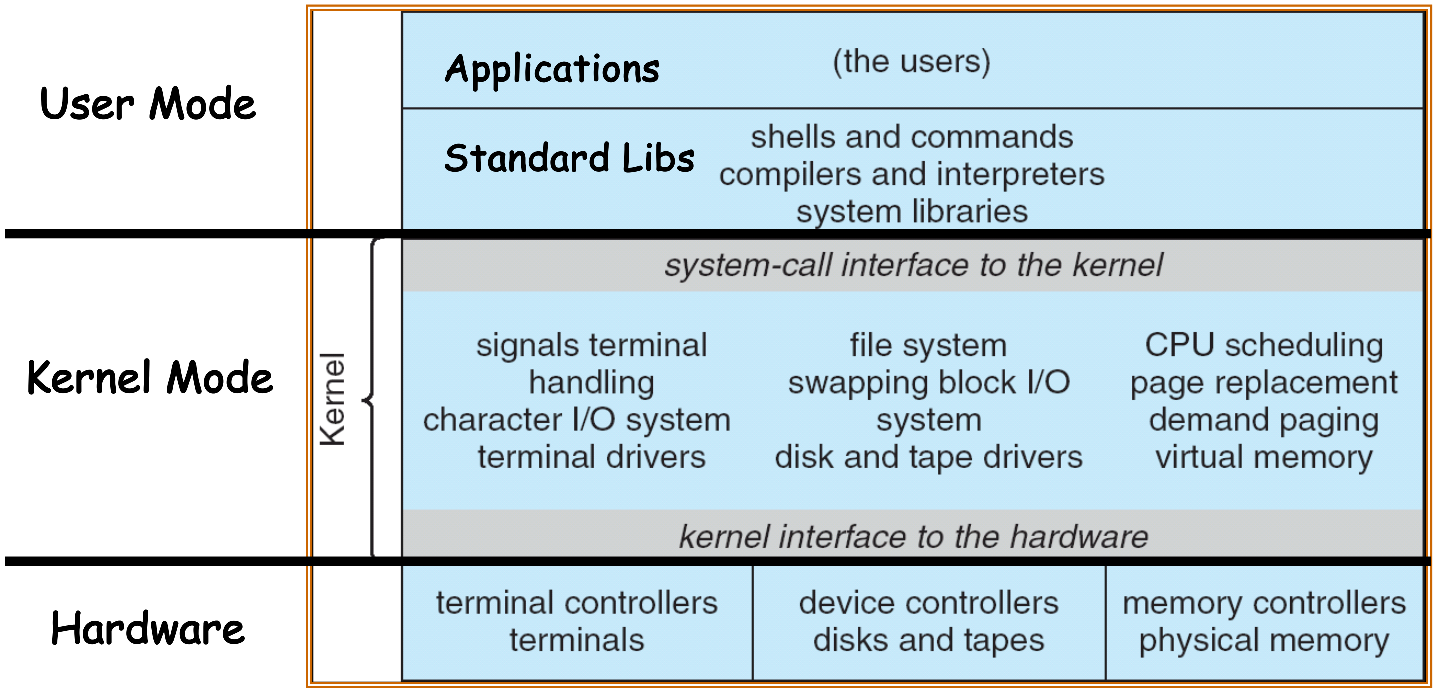


运行固件/硬件控制程序（bootloader、BIOS 等）、拥有最高权限

RISC-V

Unix System Structure

用户态程序不能直接访问硬件，而是通过 库函数 + 系统调用 间接使用内核提供的功能。



内核通过驱动程序屏蔽细节，为上层提供统一接口

3 types of Mode Transitions

- System call
  - Process requests a system service, e.g., exit
  - Like a function call, but “outside” the process
  - Does not have the address of the system function to call
  - Marshall the syscall id and args in registers and exec syscall

- 概念：进程主动请求操作系统提供的服务（例如 exit()、read()、write()）。
- 特点：
  - 类似函数调用，但目标函数不在用户进程内部，而在 内核。
  - 用户进程并不知道系统函数的具体地址，只能通过系统调用号 (syscall id) 和参数 (args) 传给内核。
  - 内核执行对应服务后再返回用户态。

典型例子：打开文件 **open()**，打印 **printf()**（底层是 **write()** 系统调用）。

- Interrupt
  - External asynchronous event triggers context switch
  - e. g., Timer, I/O device
  - Independent of user process

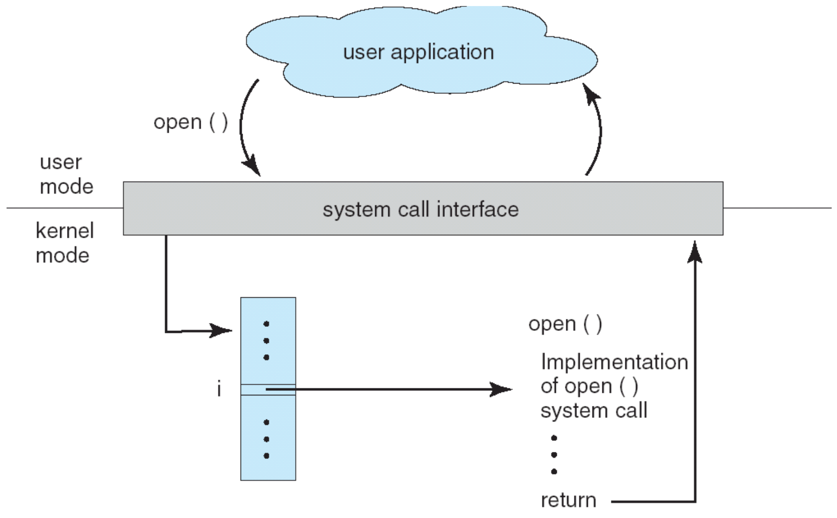
CPU 会保存当前状态，跳转到内核对应的中断处理程序。

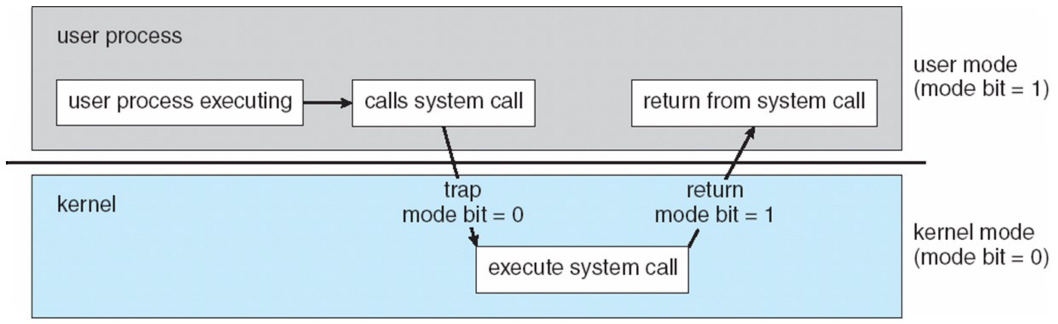
- Trap or Exception
  - Internal synchronous event in process triggers context switch
  - e.g., Protection violation (segmentation fault), Divide by zero, …

属于进程执行过程中检测到的错误或特殊情况。
CPU 会立即切换到内核态，执行异常处理程序。

System Calls

- Programming interface to the services provided by the OS
- Typically written in a high-level language (C or C++)



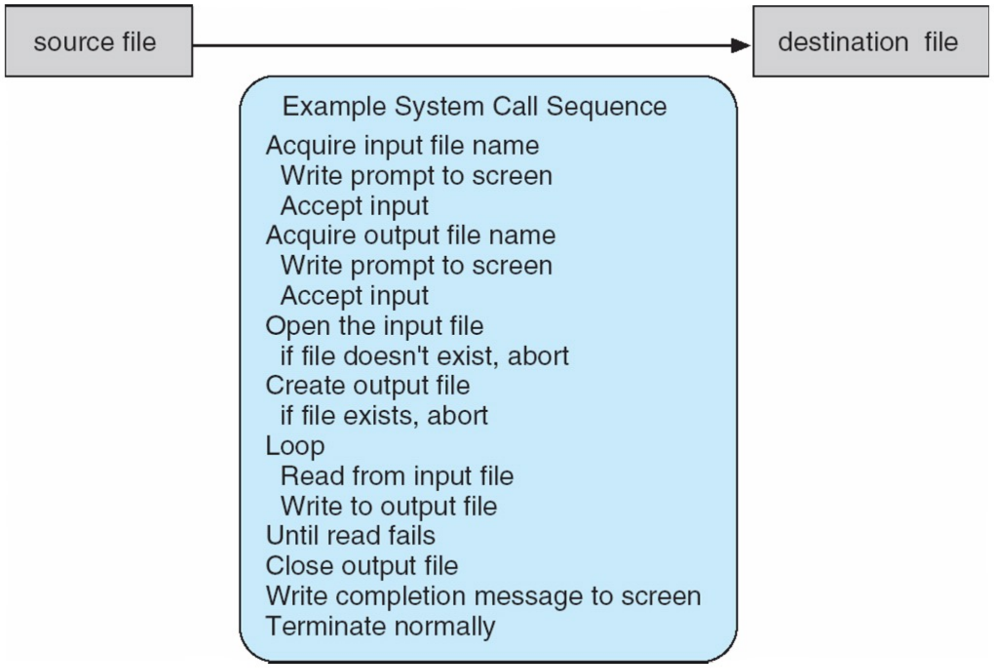


System Call Implementation

- Typically, a number associated with each system call
  - System-call interface maintains a table indexed according to these numbers
- The system call interface invokes intended system call in OS kernel and returns status of the system call and any return values
- The caller needs to know nothing about how the system call is implemented
  - Just needs to obey calling convention and understand what OS will do
  - Most details of OS interface hidden from programmer by library API
    - Managed by run-time support library (set of functions built into libraries included with compiler)

Example of System Calls

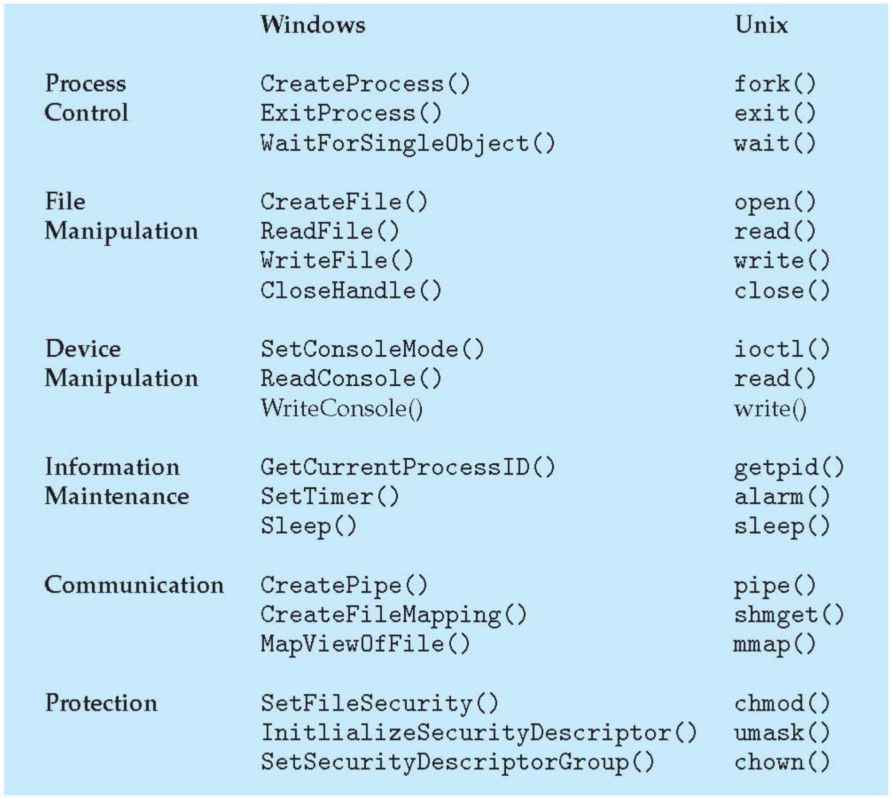
- System call sequence to copy the contents of one file to another file



```
Example System Call Sequence
Acquire input file name
  Write prompt to screen
  Accept input
Acquire output file name
  Write prompt to screen
  Accept input
Open the input file
  if file doesn't exist, abort
Create output file
  if file exists, abort
Loop
  Read from input file
  Write to output file
Until read fails
Close output file
Write completion message to screen
Terminate normally
```

Types of System Calls

- Process control
- File management
- Device management
- Information maintenance
- Communications
- Protection



	Windows	Unix

Process Control	CreateProcess()
	ExitProcess()
	WaitForSingleObject()
		fork()
		exit()
		wait()

File Manipulation	CreateFile()
	ReadFile()
	WriteFile()
	CloseHandle()
		open()
		read()
		write()
		close()

Device Manipulation	SetConsoleMode()
	ReadConsole()
	WriteConsole()
		ioctl()
		read()
		write()

Information Maintenance	GetCurrentProcessID()
	SetTimer()
	Sleep()
		getpid()
		alarm()
		sleep()

Communication	CreatePipe()
	CreateFileMapping()
	MapViewOfFile()
		pipe()
		shmget()
		mmap()

Protection	SetFileSecurity()
	InitlializeSecurityDescriptor()
	SetSecurityDescriptorGroup()
		chmod()
		umask()
		chown()

Exception and Interrupt

和代码有关，每次都会在同样的地方exception

- Exceptions (synchronous) react to an abnormal condition
  - E.g., Map the swapped-out page back to memory
  - Divide by zero
  - Illegal memory accesses

• 程序本身“犯错”或触发了某种需要操作系统介入的情况。

interrupt和代码没关系，而是和什么时候接收到cpu的指令决定的

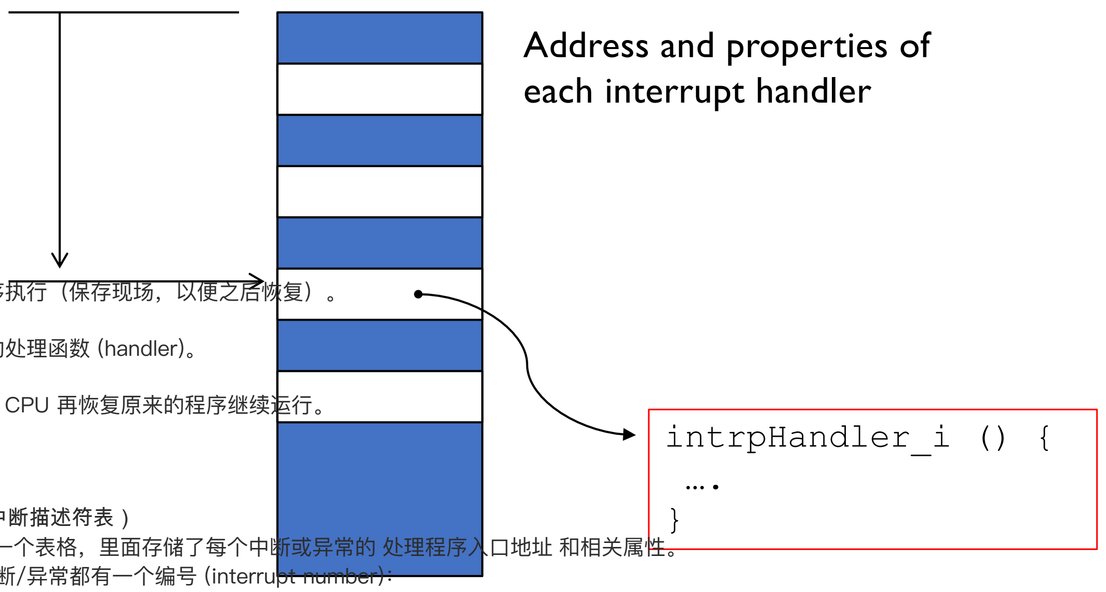
- Interrupts (asynchronous) preempt normal execution
  - Notification from device (e.g., new packets, disk I/O completed)
  - Preemptive scheduling (e.g., timer ticks)
  - Notification from another CPU (i.e., Inter-processor Interrupts)

产生原因：

- 设备完成任务（比如硬盘读写完成、收到网络数据包）
- 定时器中断（用于实现抢占式调度，CPU 切换到别的任务）
- 其他 CPU 发来的跨处理器中断 (IPI, Inter-Processor Interrupt)

Exception and Interrupt (cont'd)

- Same procedure
  - Stop execution of the current program
  - Start execution of a handler
  - Processor accesses the handler through an entry in the Interrupt Descriptor Table (IDT)
  - Each interrupt is defined by a number



```
intrpHandler_i () {
 ….
}
```

停止当前程序执行（保存现场，以便之后恢复）。

跳转到对应的处理函数 (handler)。

处理完成后，CPU 再恢复原来的程序继续运行。

IDT（中断描述符表）
IDT 是一个表格，里面存储了每个中断或异常的 处理程序入口地址 和相关属性。
每个中断/异常都有一个编号 (interrupt number):
例如 0：除零错误异常
例如 14：缺页异常 (Page Fault)
例如 32：定时器中断
CPU 收到中断/异常后，会根据编号去 IDT 查表，找到对应的处理函数地址，然后执行该 handler。

Kernel Structures

宏内核

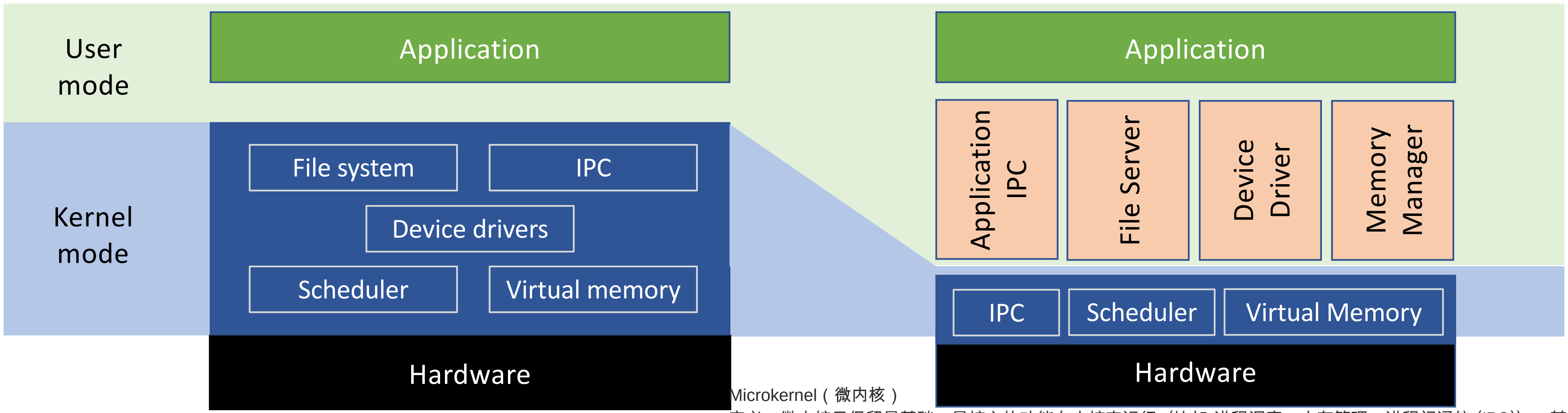
Monolithic Kernel

- A monolithic kernel is an operating system software framework that holds all privileges to access I/O devices, memory, hardware interrupts and the CPU stack.
- Monolithic kernels contain many components, such as memory subsystems and I/O subsystems, and are usually very large
  - Including filesystems, device drivers, etc.
- Monolithic kernel is the basis for Linux, Unix, MS-DOS.

Monolithic Kernel (单体内核)
定义：单体内核是一种将操作系统几乎所有核心功能都放在内核空间（kernel mode）里运行的设计。
特点：
内核有 最高权限（能直接访问 I/O 设备、内存、中断、CPU 等）。
包含很多子系统：内存管理、I/O 子系统、文件系统、设备驱动等。
规模庞大，代码量大。
优点：
运行效率高：因为所有功能都在内核态，调用开销小。
缺点：
安全性差：一个驱动或子系统出错，可能导致整个系统崩溃。
扩展性差：要修改或更新某个模块，往往需要改动整个内核。
例子：Linux、Unix、MS-DOS 都是单体内核。

Micro Kernel

- Microkernels outsource the traditional operating system functionality to ordinary user processes for better flexibility, security, and fault tolerance.



Microkernel（微内核）
定义：微内核只保留最基础、最核心的功能在内核态运行（比如 进程调度、内存管理、进程间通信 (IPC)），其他功能（文件系统、设备驱动等）移到用户态作为独立服务运行。
特点：
内核更小、更简洁。
通过 消息传递 (IPC) 在用户态和内核态之间通信。
优点：
安全性高：用户态的服务崩溃不会影响整个系统。
容错性好：出问题的模块可以单独重启。
灵活性强：容易扩展和维护。
缺点：
性能开销大：因为很多功能需要频繁在用户态和内核态之间切换。
例子：Minix、QNX、部分现代操作系统的实验版本。

Micro Kernel (Cont'd)

- OS functionalities are pushed to user-level servers (e.g., user-level memory manager)
- User-level servers are trusted by the kernel (often run as root)

内核只负责最底层的 保护机制（确保一个进程不能随意访问另一个进程的内存）。
而 资源管理策略（如怎么分配多少内存给进程）交给用户态服务去做。

- Protection mechanisms stay in kernel while resource management policies go to the user-level servers
- Representative micro-kernel OS
  - Mach, 1980s at CMU
  - seL4, the first formally verified micro-kernel, http://sel4.systems/

Micro Kernel (Cont'd)

- Pros

  因为大部分功能在用户态，内核只做最基本的事，出错影响小。

  - Kernel is more responsive (kernel functions in preemptible user-level processes)
  - Better stability and security (less code in kernel)
  - Better support of concurrency and distributed OS (later.....)
- Cons
  - More IPC needed and thus more context switches (slower)

  因为内核和用户态之间需要频繁通信 (IPC)，会导致更多 上下文切换 (context switches)，速度比单体内核慢。

Hybrid Kernel

- A combination of a monolithic kernel and a micro kernel
  - Example: Windows OS

特点：

把一些传统上放在微内核用户态的功能（如驱动）放回内核态以提升性能。

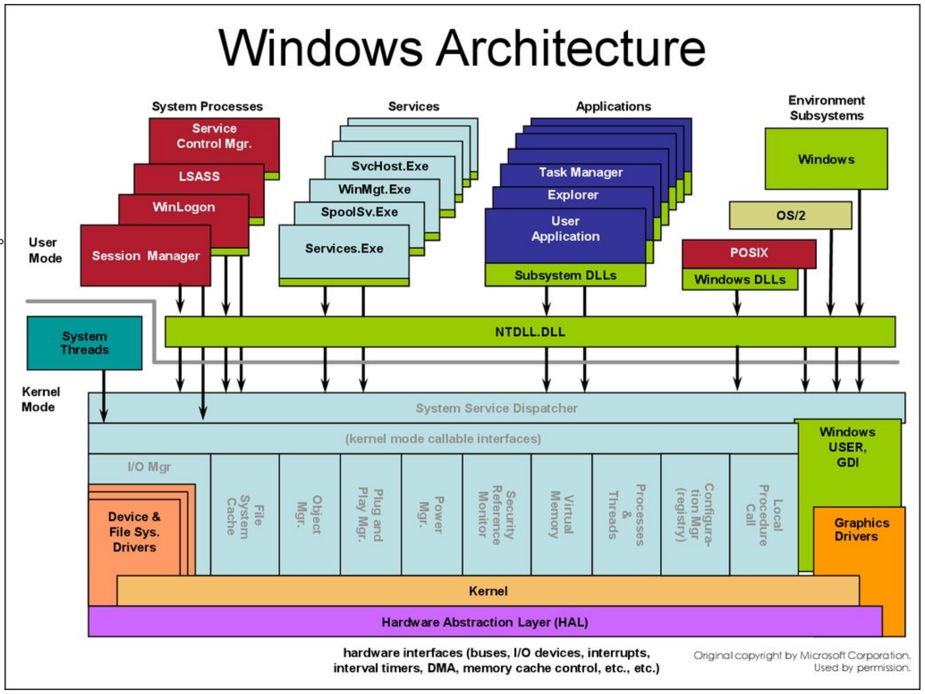
但依然保留部分模块化设计，提升灵活性和稳定性。

实例：

Windows OS 就是典型的 Hybrid Kernel。
图中展示了 Windows 的分层结构：

用户态：应用程序、服务、子系统。

内核态：系统调用接口、内存管理、设备驱动、I/O 管理、调度器等。



FrameKernel

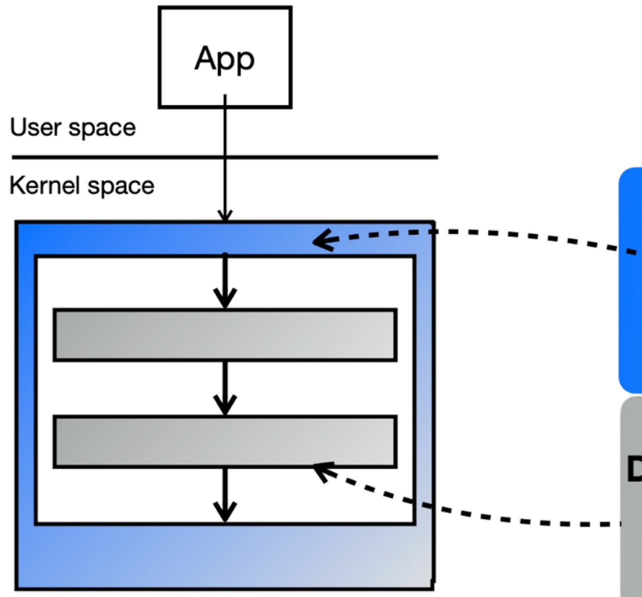
FrameKernel 是一种新型的内核设计理念，结合了：

单一地址空间 (single address space)

安全语言 (safe language, 如 Rust)

安全/不安全代码分区 (safe/unsafe halves)

A framekernel = single address space + safe language + safe/unsafe halves



Framekernel

	Allow unsafe?	Responsibilities	Code Sizes	Memory Safety
Privileged OS Framework	Yes	Encapsulate low-level unsafe code into safe abstractions	Small	Examined by programmers
De-privileged OS Services	No	Implement OS functionalities, including device drivers	Large	Guaranteed by Rust compiler

Asterinas: the First FrameKernel Impl

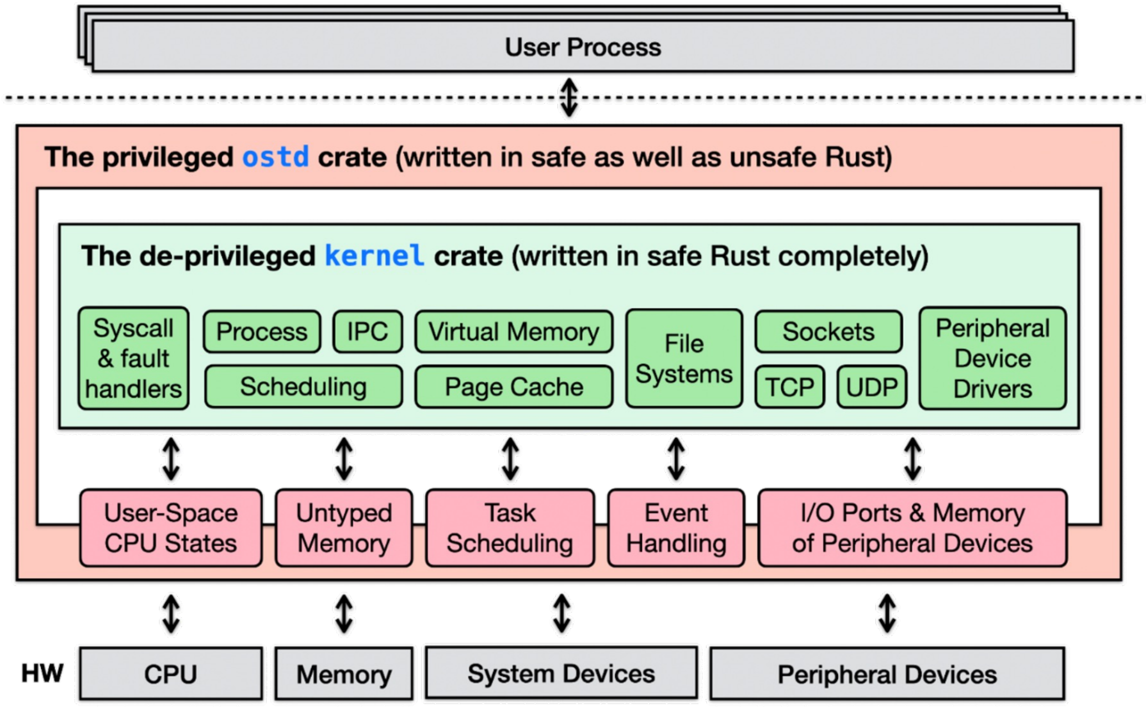


Figure. An overview of Asterinas

- **Feature rich:** >170 Linux system calls implemented with a total of 70K lines of Rust
- **Minimized TCB:** ~20% compared to RedLeaf (~60%) and Theseus (~50%)
- **Open sourced:** https://github.com/asterinas/asterinas

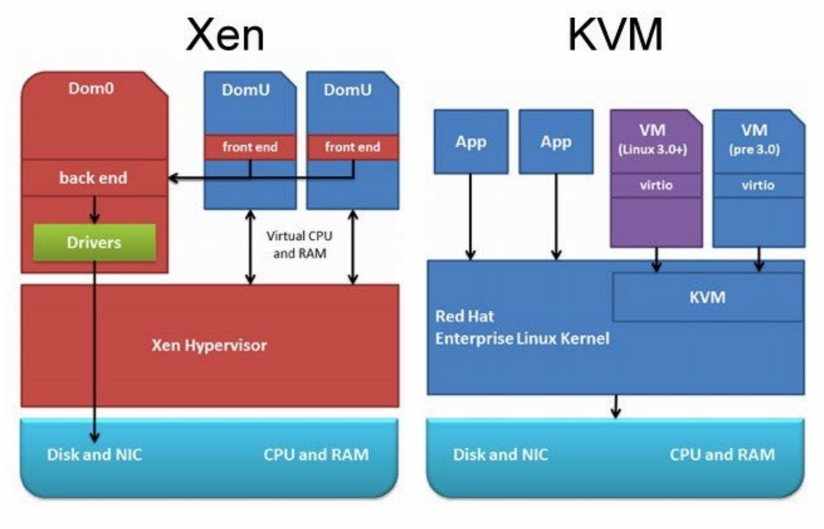
极小的 TCB（Trusted Computing Base）：

・TCB 表示操作系统中必须信任的最小核心代码。

・Asterinas 的 TCB 只有 ~20%，相比 RedLeaf (~60%) 和 Theseus (~50%) 小得多，意味着更安全。

Virtualization and Hypervisors

- Hypervisor (or virtual machine manager/monitor, or VMM) emphasizes on **virtualization and isolation**
    - OS can run on hypervisor (almost) without modification
    - Resource partition among VMs
    - Micro kernel can sometimes be used to implement hypervisors



类型与例子：

Xen（典型的 Type-1 Hypervisor）：

Xen 直接运行在硬件上，虚拟机通过 Xen 提供的接口访问资源。

图中显示 Dom0（管理虚拟机）和 DomU（用户虚拟机）。

KVM（Kernel-based Virtual Machine）（Type-2 Hypervisor）：

基于 Linux 内核。

Linux 作为宿主机，KVM 作为内核模块提供虚拟化支持。

每个虚拟机（VM guest）作为用户进程运行。

OS Design Principles

- Internal structure of different Operating Systems can vary widely
  - Start by defining goals and specifications
  - Affected by choice of hardware, type of system
- User goals and System goals
  - User goals – operating system should be convenient to use, easy to learn, reliable, safe, and fast
  - System goals – operating system should be easy to design, implement, and maintain, as well as flexible, reliable, error-free, and efficient

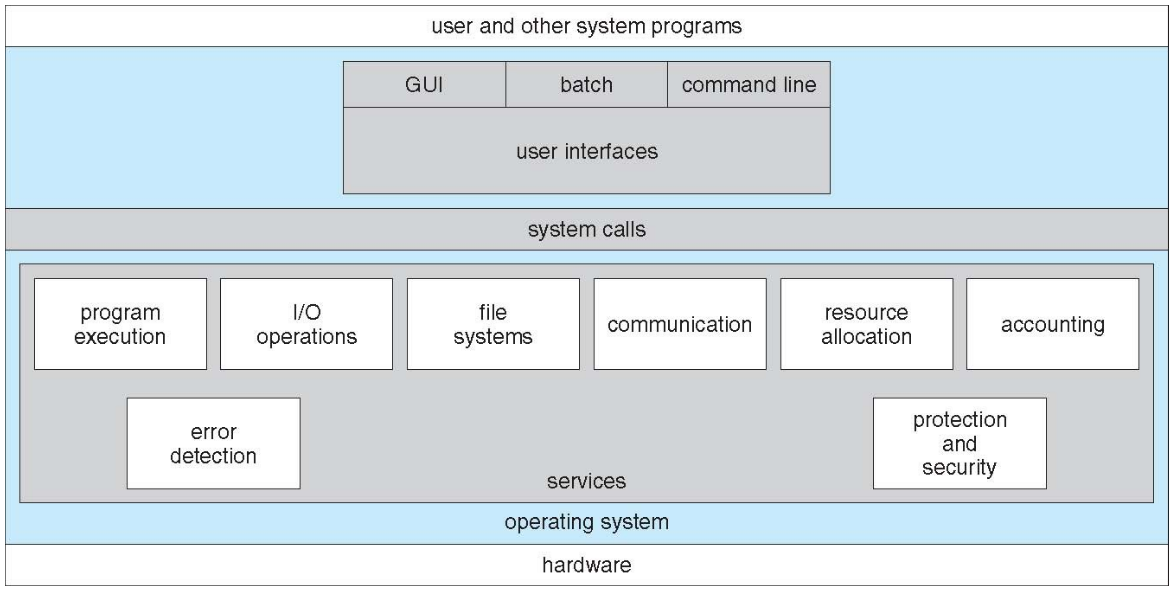
OS Design Principles

- OS separates policies and mechanisms
  - Policy: which software could access which resource at what time
    - E.g., if two processes access the same device at the same time, which one goes first
    - E.g., if a process hopes to read from keyboard
  - Mechanism: How is the policy enforced
    - E.g., request queues for devices, running queues for CPUs
    - E.g., access control list for files, devices, etc.
  - The separation of policy from mechanism is a very important principle, it allows maximum flexibility if policy decisions are to be changed later

Operating System Services

Operating System Services

- Operating system provides a set of services to application programs



Operating System Services

- One set of operating-system 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
  - **File-system manipulation** - The file system is of particular interest. Obviously, programs need to read and write files and directories, create and delete them, search them, list file Information, permission management

Operating System Services (Cont)

- One set of operating-system services provides functions that are helpful to the user (Cont):
  - **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

Operating System Services (Cont)

- 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 - Some (such as CPU cycles, main memory, and file storage) may have special allocation code, others (such as I/O devices) may have general request and release code
  - **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
    - If a system is to be protected and secure, precautions must be instituted throughout it. A chain is only as strong as its weakest link

系统安全性取决于最薄弱的一环（链条效应）

User Operating System Interface - CLI

- Command Line Interface (CLI) or command interpreter allows direct command entry
  - Sometimes implemented in kernel, sometimes by systems program
  - Shells: Bourne shell, C Shell, Bourne-Again Shell, Korn Shell
  - Primarily fetches a command from user and executes it
    - Sometimes commands built-in, sometimes just names of programs
    - If the latter, adding new features doesn't require shell modification

Shell 负责从用户那里获取命令，然后执行该命令。

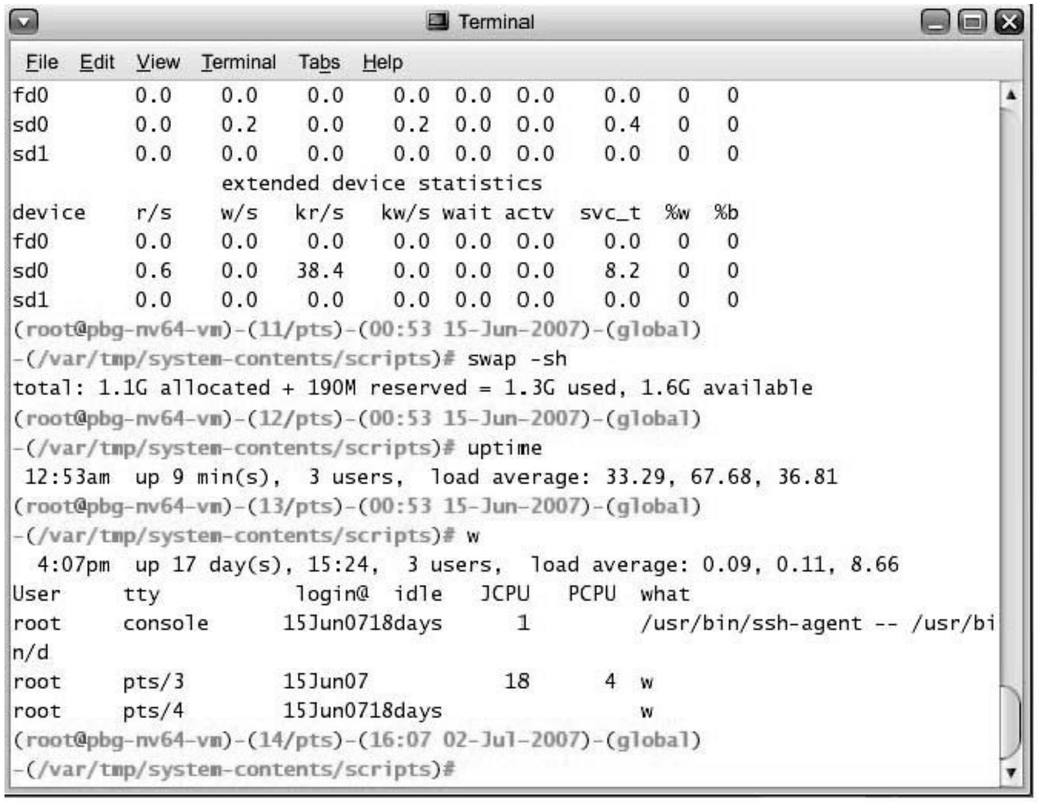
命令可能是内置命令，也可能是外部程序的名字。

外部程序的好处是：添加新功能时无需修改 Shell。

User Operating System Interface - GUI

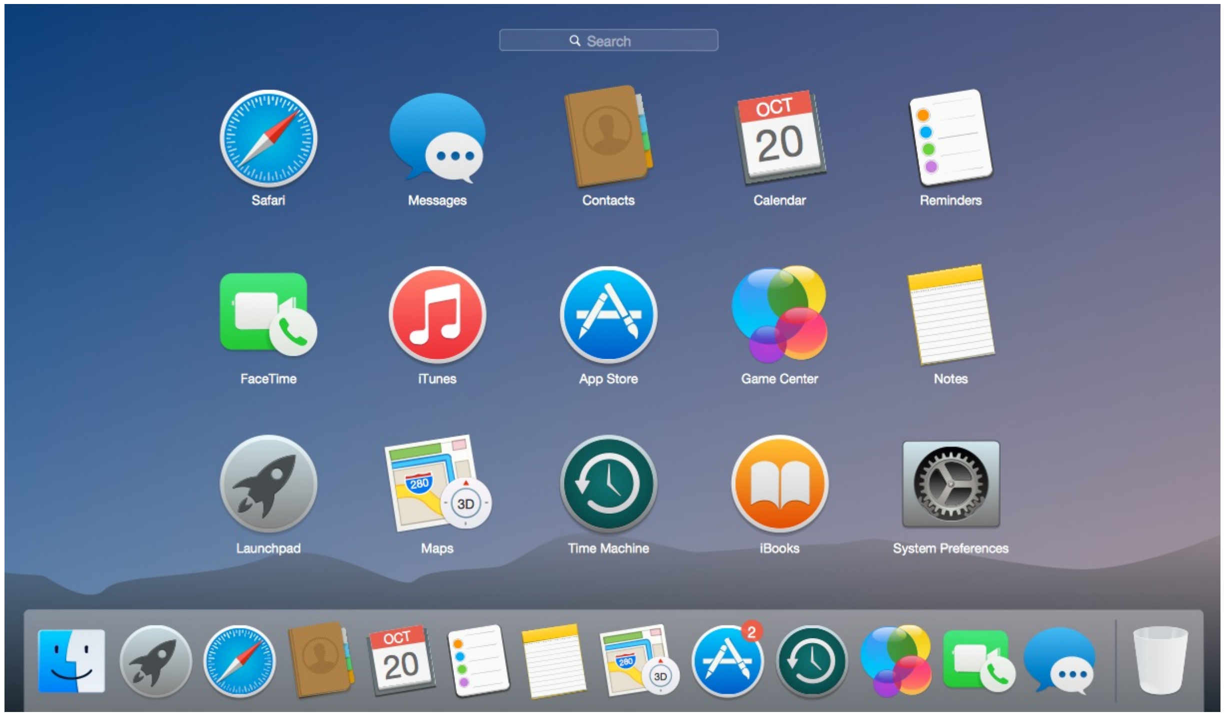
- User-friendly desktop metaphor interface
  - Usually mouse, keyboard, and monitor
  - Icons represent files, programs, actions, etc
  - Various mouse buttons over objects in the interface cause various actions: provide information, options, execute function, open directory (known as a folder)
  - Invented at Xerox PARC
- Many systems now include both CLI and GUI interfaces
  - Microsoft Windows is GUI with CLI "command" shell
  - Apple Mac OS X as "Aqua" GUI interface with UNIX kernel underneath and shells available
  - Solaris is CLI with optional GUI interfaces (Java Desktop, KDE)

Bourne Shell Command Interpreter



```
fd0        0.0    0.0    0.0    0.0  0.0  0.0    0.0   0   0
sd0        0.0    0.2    0.0    0.2  0.0  0.0    0.4   0   0
sd1        0.0    0.0    0.0    0.0  0.0  0.0    0.0   0   0
                  extended device statistics
device     r/s    w/s   kr/s   kw/s wait actv  svc_t  %w  %b
fd0        0.0    0.0    0.0    0.0  0.0  0.0    0.0   0   0
sd0        0.6    0.0   38.4    0.0  0.0  0.0    8.2   0   0
sd1        0.0    0.0    0.0    0.0  0.0  0.0    0.0   0   0
(root@pbg-nv64-vm)-(11/pts)-(00:53 15-Jun-2007)-(global)
-(/var/tmp/system-contents/scripts)# swap -sh
total: 1.1G allocated + 190M reserved = 1.3G used, 1.6G available
(root@pbg-nv64-vm)-(12/pts)-(00:53 15-Jun-2007)-(global)
-(/var/tmp/system-contents/scripts)# uptime
 12:53am  up 9 min(s),  3 users,  load average: 33.29, 67.68, 36.81
(root@pbg-nv64-vm)-(13/pts)-(00:53 15-Jun-2007)-(global)
-(/var/tmp/system-contents/scripts)# w
  4:07pm  up 17 day(s), 15:24,  3 users,  load average: 0.09, 0.11, 8.66
User     tty           login@  idle   JCPU   PCPU  what
root     console      15Jun0718days      1         /usr/bin/ssh-agent -- /usr/bi
n/d
root     pts/3        15Jun07           18      4  w
root     pts/4        15Jun0718days                w
(root@pbg-nv64-vm)-(14/pts)-(16:07 02-Jul-2007)-(global)
-(/var/tmp/system-contents/scripts)#
```

The Mac OS X GUI



System Programs

- System programs provide a convenient environment for program development and execution. They can be divided into:
  - File manipulation
  - Status information
  - File modification
  - Programming language support
  - Program loading and execution
  - Communications
  - Application programs
- Most users' view of the operation system is defined by system programs

System Programs (cont'd)

- Provide a convenient environment for program development and execution
  - Some of them are simply user interfaces to system calls; others are considerably more complex
- File management - Create, delete, copy, rename, print, dump, list, and generally manipulate files and directories
- Status information
  - Some ask the system for info - date, time, amount of available memory, disk space, number of users
  - Others provide detailed performance, logging, and debugging information
  - Typically, these programs format and print the output to the terminal or other output devices
  - Some systems implement a registry - used to store and retrieve configuration information

System Programs (cont'd)

- File modification
  - Text editors to create and modify files
  - Special commands to search contents of files or perform transformations of the text
- Programming-language support - Compilers, assemblers, debuggers and interpreters sometimes provided
- Program loading and execution
- Communications - Provide the mechanism for creating virtual connections among processes, users, and computer systems
  - Allow users to send messages to one another's screens, browse web pages, send electronic-mail messages, log in remotely, transfer files from one machine to another

Thank you!