CSCI 360
Introduction to Operating Systems

Introduction

Humayun Kabir

Professor, CS, Vancouver Island University, BC, Canada

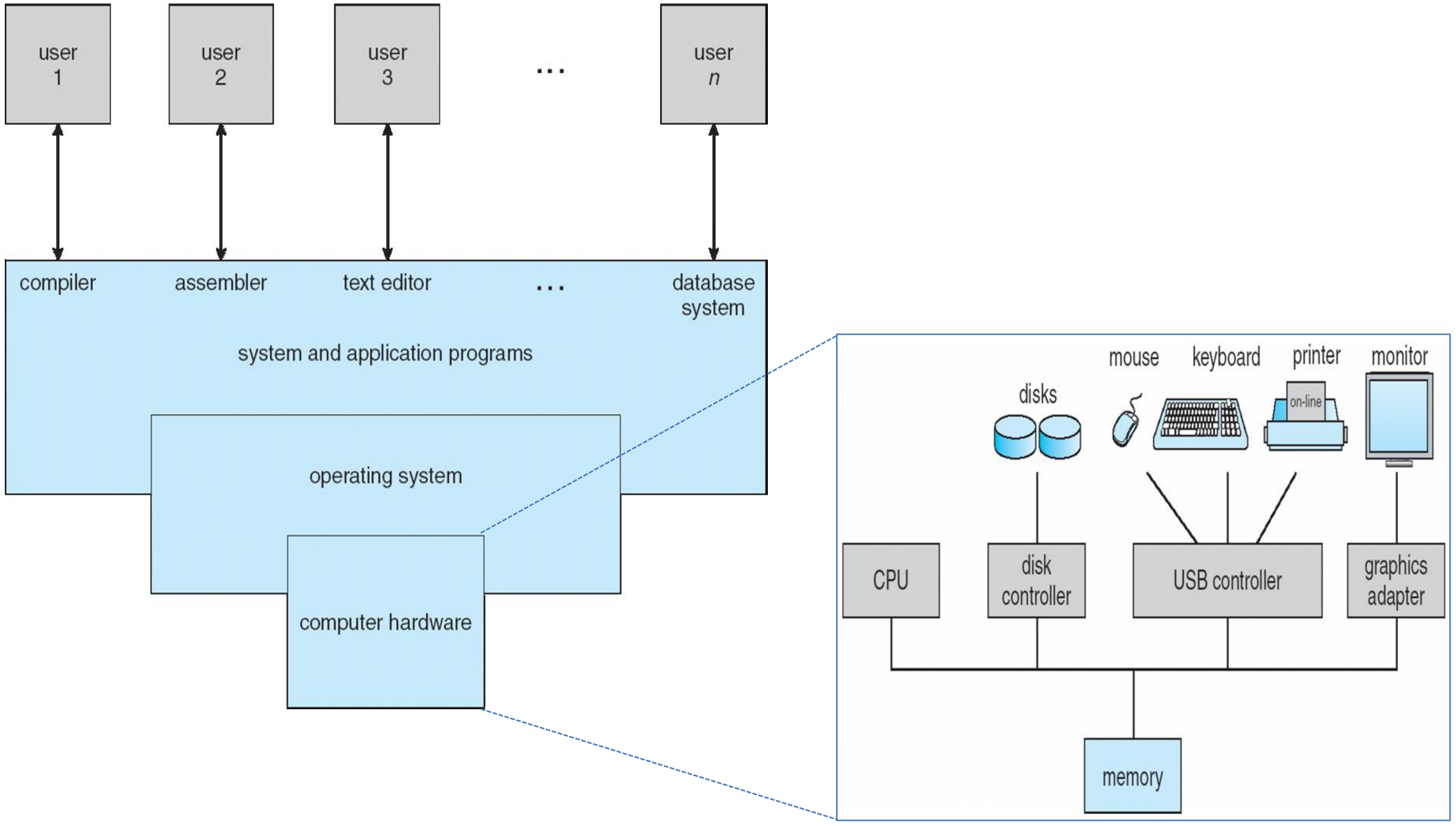
Outline

- What is Operating System?
- Operating System Roles
- Operating System Components
- Operating System Modes
- System Calls
- Operating System Architecture

What is an Operating System?

- Unix, FreeBSD (UC Berkeley Unix)
- Minix, Mach, L4
- Linux
- Mac OS X
- Windows
- Android
- iOS

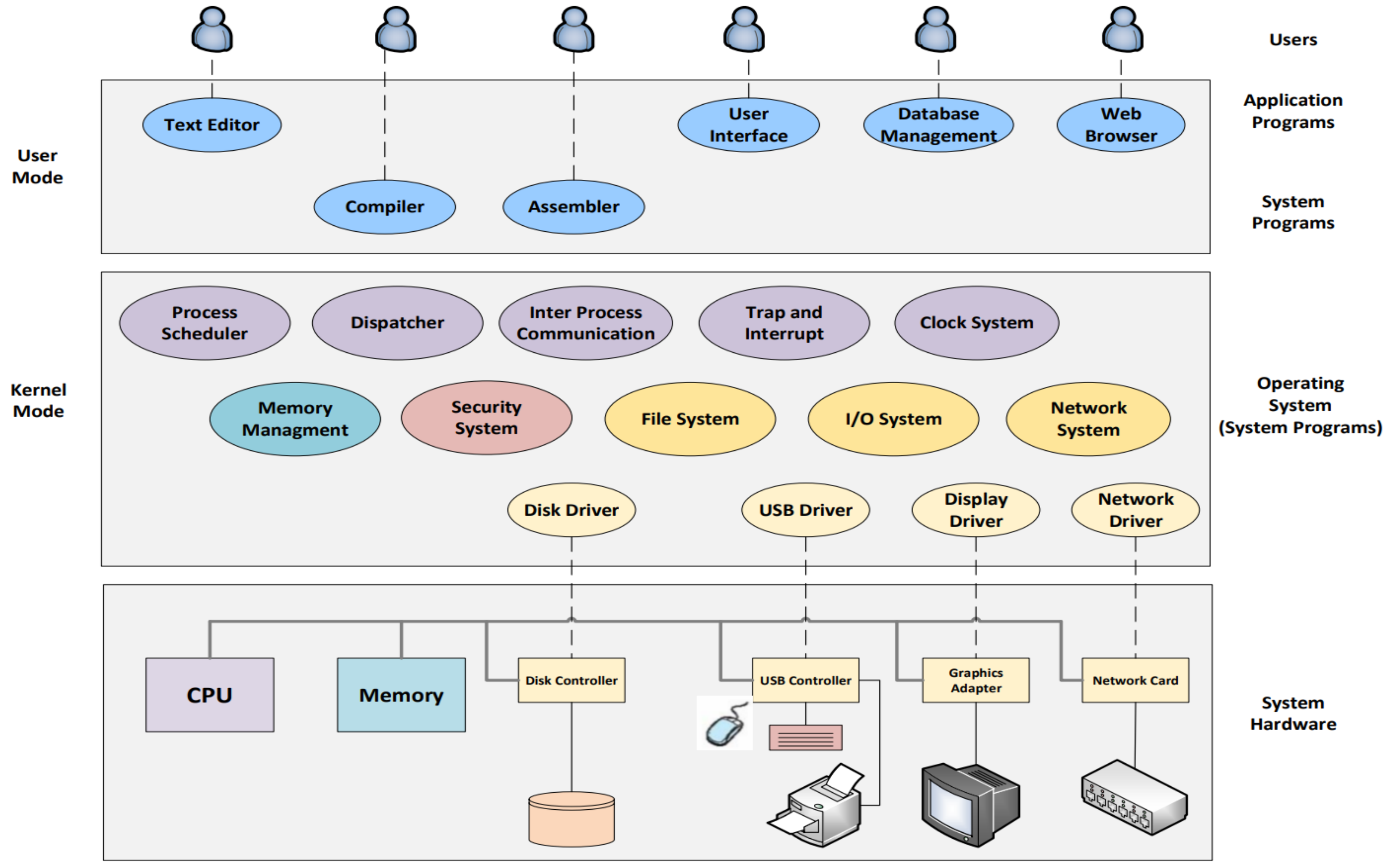
What is an Operating System?



What is an Operating System?

- A system software that acts as an intermediary between the application software and computer hardware
- Operating System:
  - Executes application software.
  - Provides system level services to the application software.
  - Provides convenience to the application software developers.
  - Controls and enables efficient usage of system hardware.

What is an Operating System?



Operating System Roles

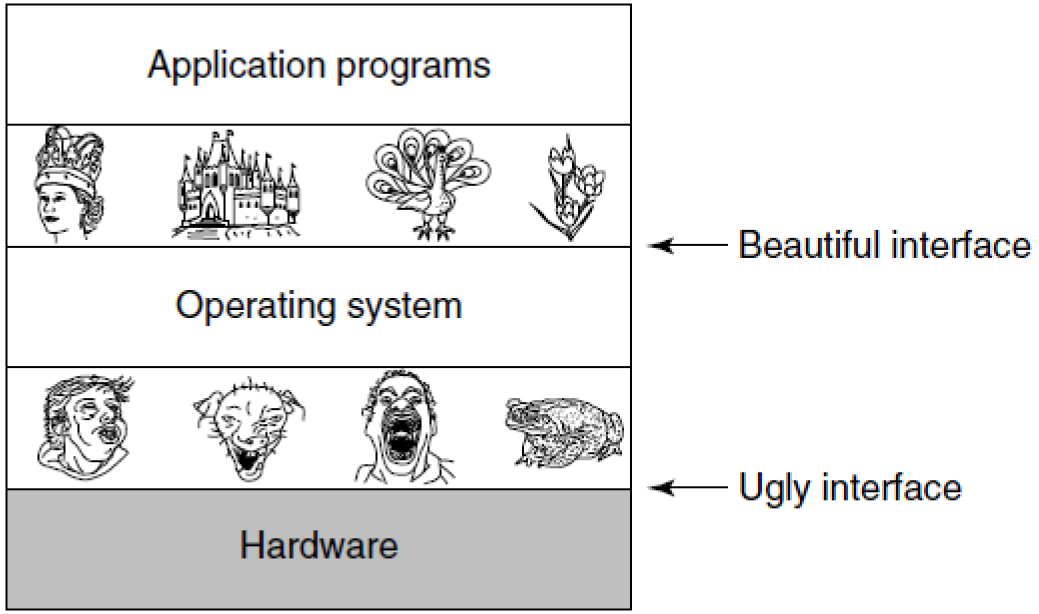
- Two important roles:
  - Provides a **nice abstraction** around the hardware or **extend the machine.**
  - **Manage** the hardware **resources**.

OS Roles: Extended Machine

- Computer architecture at low level is primitive and awkward to program.
- Application programmers do not want to get too intimately involved at low level.
- Application programmers want simple and high-level abstraction of the architecture to deal with.

OS Roles: Extended Machine

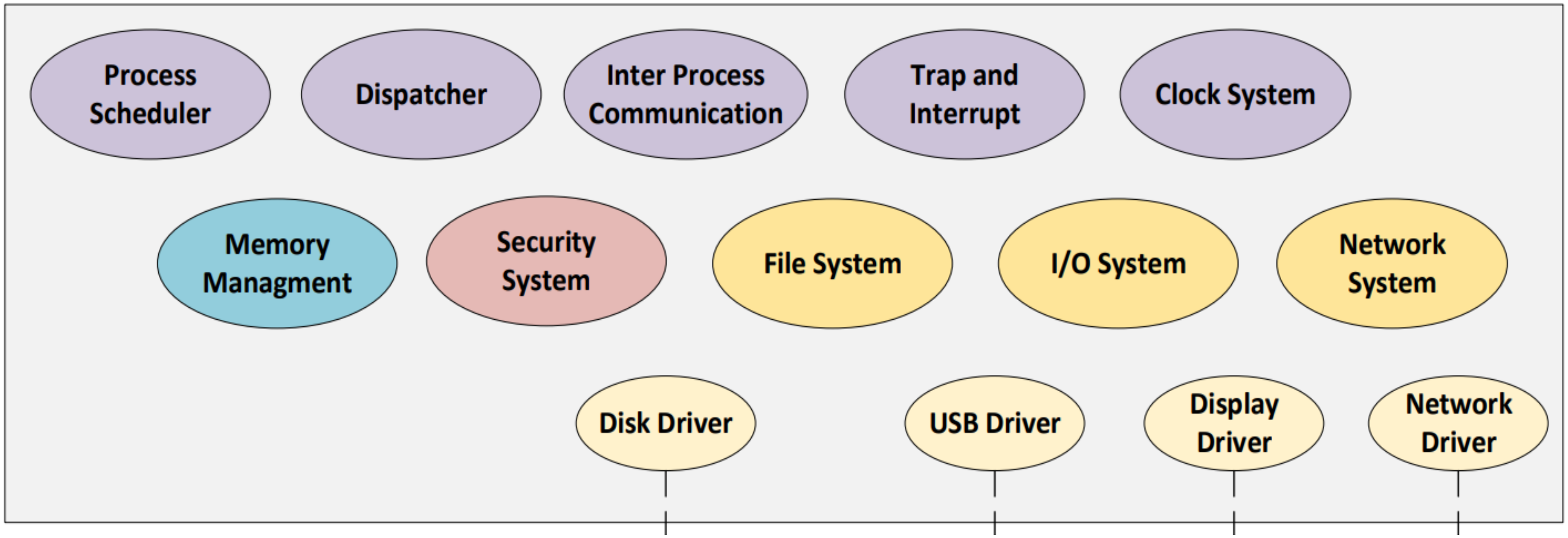
- Operating system hides the complex hardware and presents nice, clean, elegant, consistent abstractions to work with.



OS Roles: Resource Manager

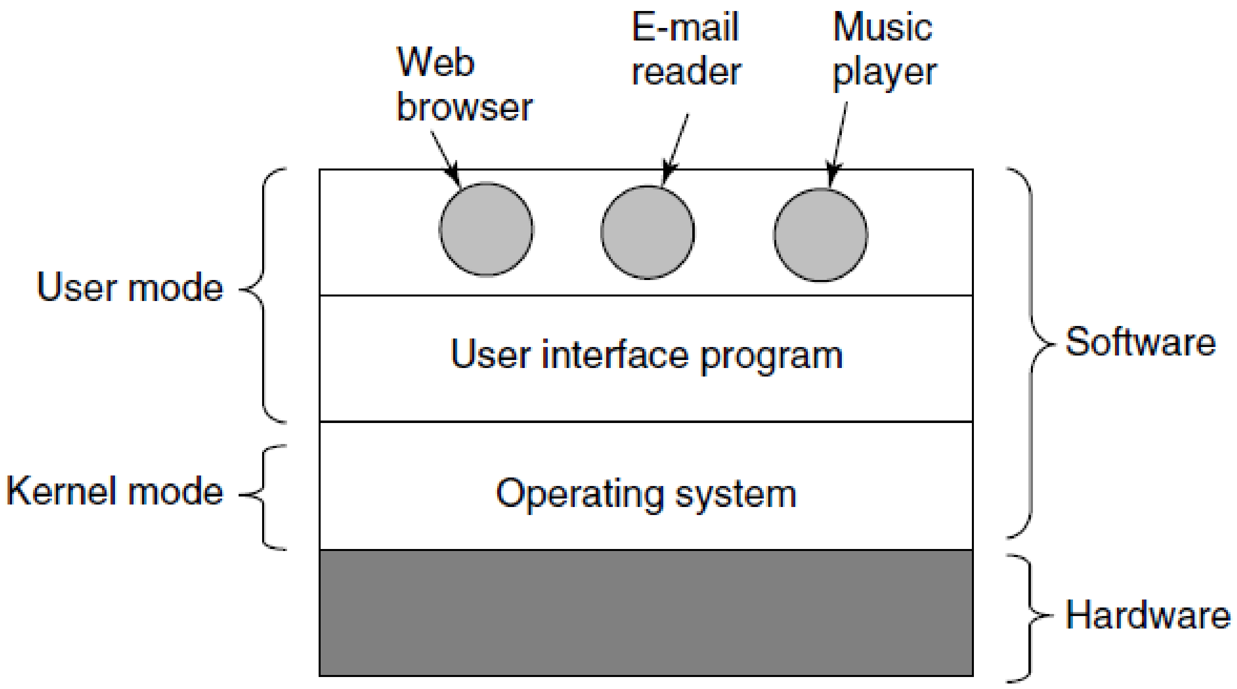
- Provides orderly, controlled allocation of resources
  - Keeps track which programs are using which resources.
  - Grants resource requests and accounts resource usage.
  - Mediate conflicting resource requests.
- Shares resources
  - Time and space multiplexing

Operating System Components



- Consists of many components and each component performs specific tasks.
- Will learn details about followings components:
  - Process Management System
  - Memory Management System
  - File System
  - I/O System

Operating System Modes

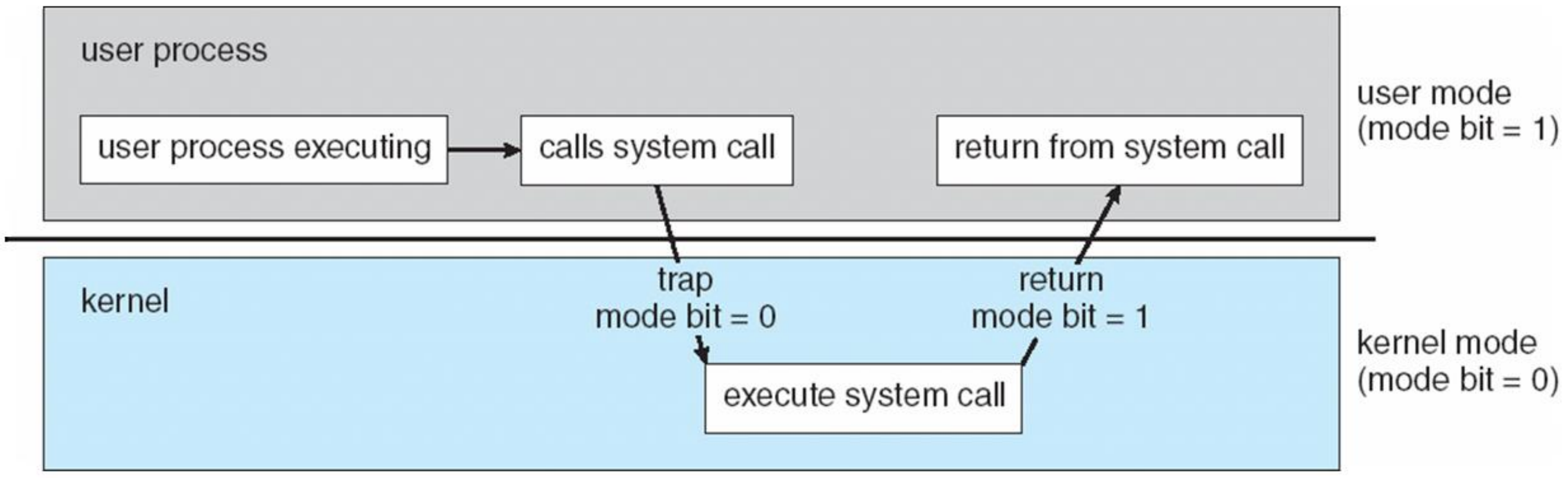


Operating System Modes

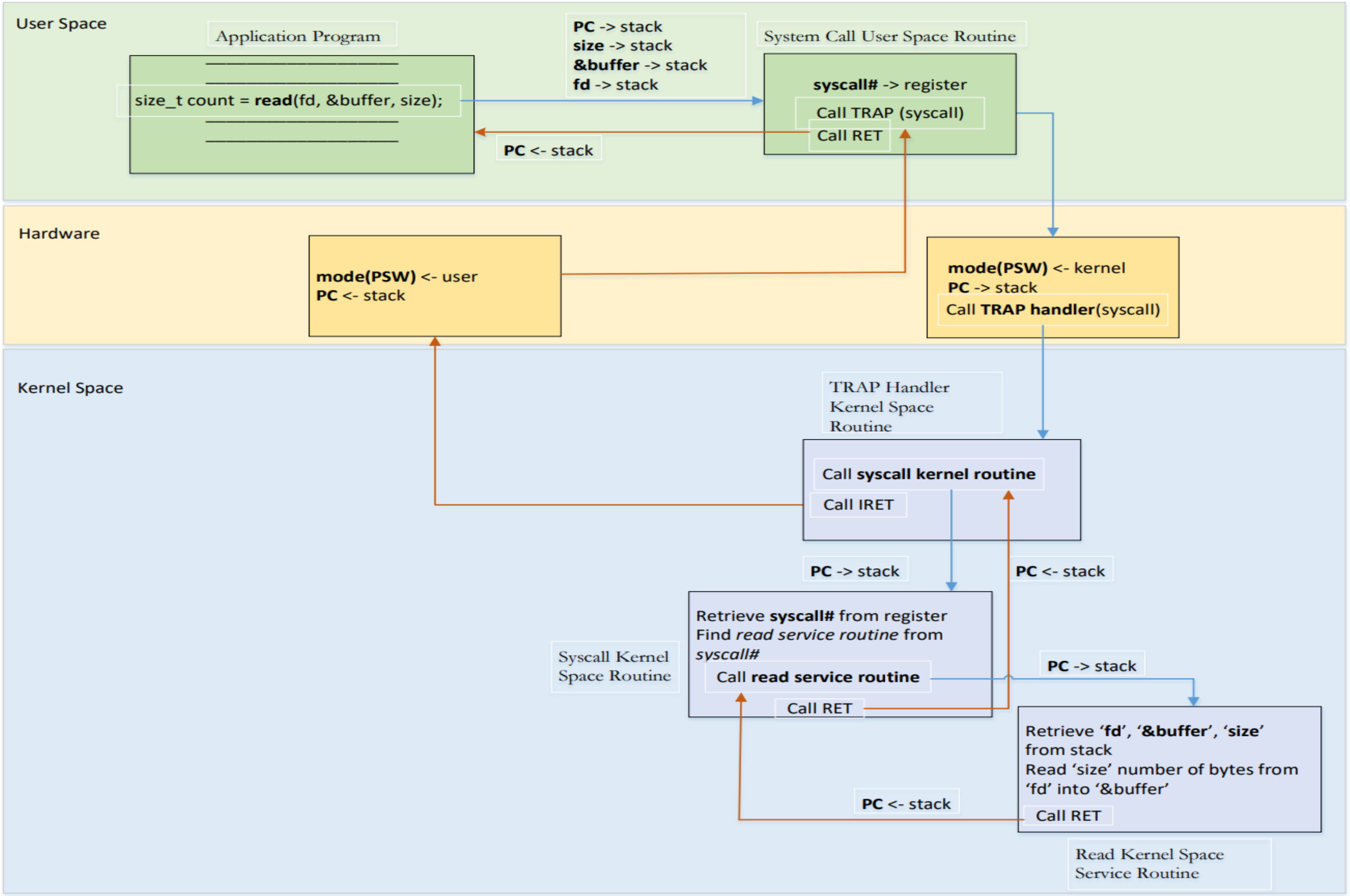
- **Dual-mode** (**user mode** and **kernel mode**) operation allows OS to protect itself and other system components
  - **OS** runs in **kernel mode**, has complete access to all the hardware and can execute all instructions, including **privileged** ones.
  - **Other Programs** run in **User mode,** cannot access the hardware directly and cannot run **privileged** instructions.
  - **Programs** in **User mode,** can indirectly access hardware and execute **privileged** instructions through **system calls.**

Operating System Modes

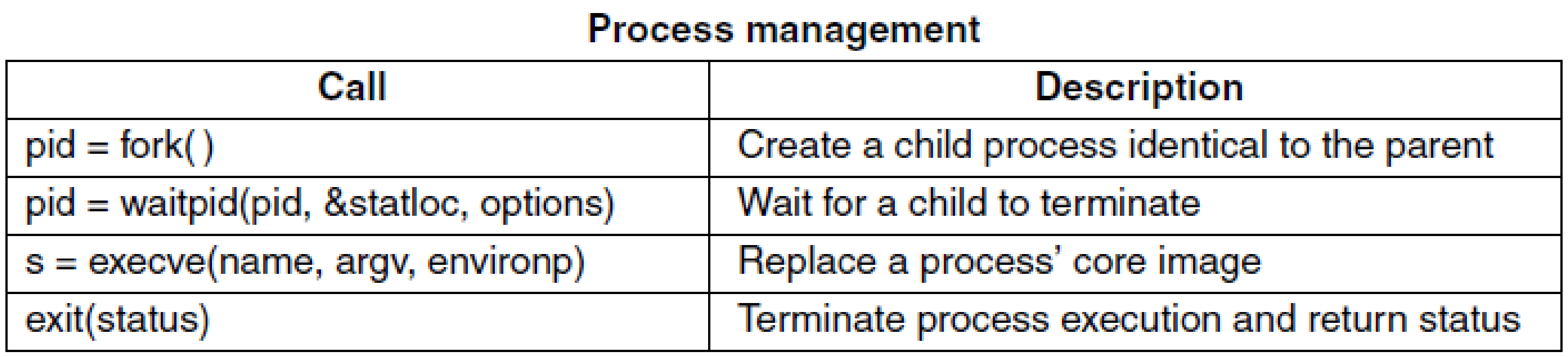
- **System call** changes mode to kernel, return from call resets it to user
- **Mode bit** provided by hardware, gives the ability to distinguish when system is running user code or kernel code



System Calls

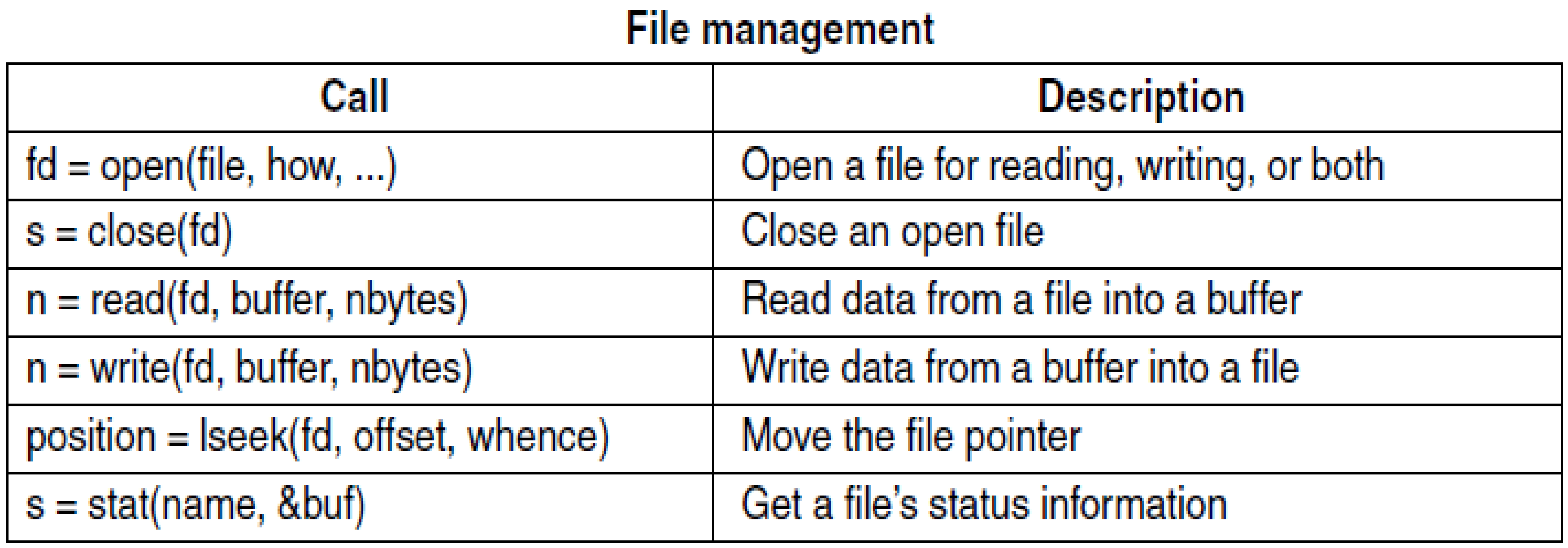

User Space
Application Program
size_t count = read(fd, &buffer, size);
PC -> stack
size -> stack
&buffer -> stack
fd -> stack
System Call User Space Routine
syscall# -> register
Call TRAP (syscall)
Call RET
PC <- stack
Hardware
mode(PSW) <- user
PC <- stack
mode(PSW) <- kernel
PC -> stack
Call TRAP handler(syscall)
Kernel Space
TRAP Handler Kernel Space Routine
Call syscall kernel routine
Call IRET
PC -> stack
PC <- stack
Retrieve syscall# from register
Find read service routine from syscall#
Call read service routine
Call RET
Syscall Kernel Space Routine
PC -> stack
Retrieve 'fd', '&buffer', 'size' from stack
Read 'size' number of bytes from 'fd' into '&buffer'
Call RET
PC <- stack
Read Kernel Space Service Routine

Unix System Calls



Process management

Call	Description
pid = fork( )	Create a child process identical to the parent
pid = waitpid(pid, &statloc, options)	Wait for a child to terminate
s = execve(name, argv, environp)	Replace a process' core image
exit(status)	Terminate process execution and return status



File management

Call	Description
fd = open(file, how, ...)	Open a file for reading, writing, or both
s = close(fd)	Close an open file
n = read(fd, buffer, nbytes)	Read data from a file into a buffer
n = write(fd, buffer, nbytes)	Write data from a buffer into a file
position = lseek(fd, offset, whence)	Move the file pointer
s = stat(name, &buf)	Get a file's status information

Unix System Calls

Directory and file system management

Call	Description
s = mkdir(name, mode)	Create a new directory
s = rmdir(name)	Remove an empty directory
s = link(name1, name2)	Create a new entry, name2, pointing to name1
s = unlink(name)	Remove a directory entry
s = mount(special, name, flag)	Mount a file system
s = umount(special)	Unmount a file system

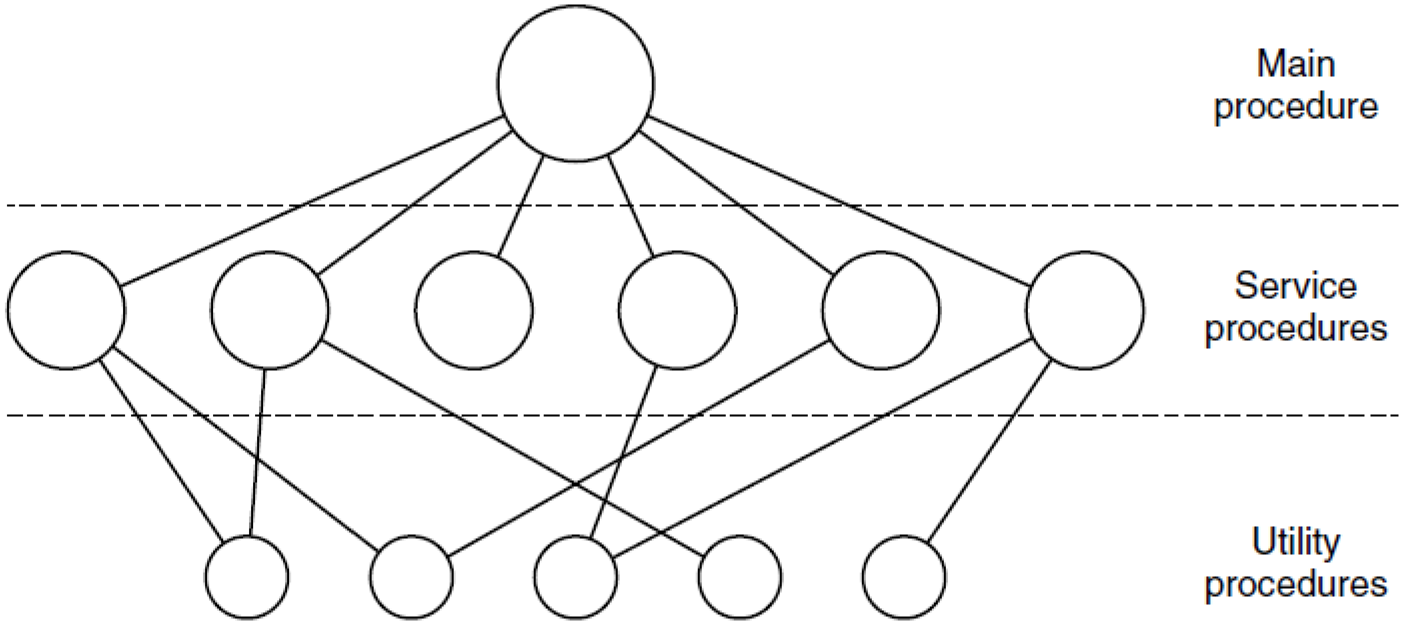
Miscellaneous

Call	Description
s = chdir(dirname)	Change the working directory
s = chmod(name, mode)	Change a file's protection bits
s = kill(pid, signal)	Send a signal to a process
seconds = time(&seconds)	Get the elapsed time since Jan. 1, 1970

Operating System Architecture

- Way to organize operating system components.
- Two dominating architectures:
  - Monolithic
  - Microkernels

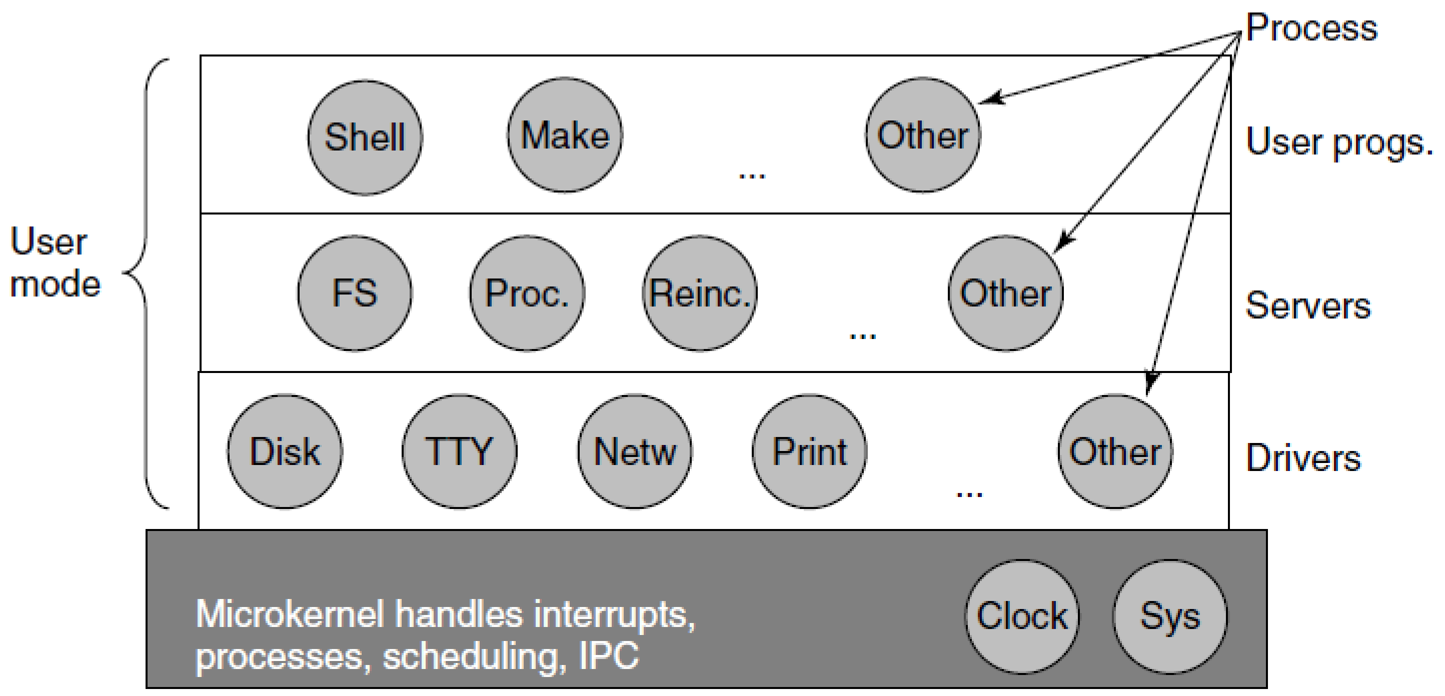
OS Architecture: Monolithic



OS Architecture: Monolithic

- A main program that invokes the requested service procedure.
- A set of service procedures that carry out the system calls.
- A set of utility procedures that help the service procedures.

OS Architecture: Microkernels



Summary

- What is Operating System?
  - ✓ System software
- Operating System Roles
  - ✓ Extended Machine
  - ✓ Resource Manager
- Operating System Components
- Operating System Modes
  - ✓ User Mode
  - ✓ Kernel Mode
- System Calls
- Operating System Architecture
  - ✓ Monolithic
  - ✓ Microkernels

Next

Process Management

- Process Abstraction
- Process Operations
- Process States
- Process Scheduling
- Context Switching
- Inter Process Communications (IPC)
- Process Synchronization