CSCI 360
Introduction to Operating Systems

Introduction

Humayun Kabir

Professor, CS, Vancouver Island University, BC, Canada

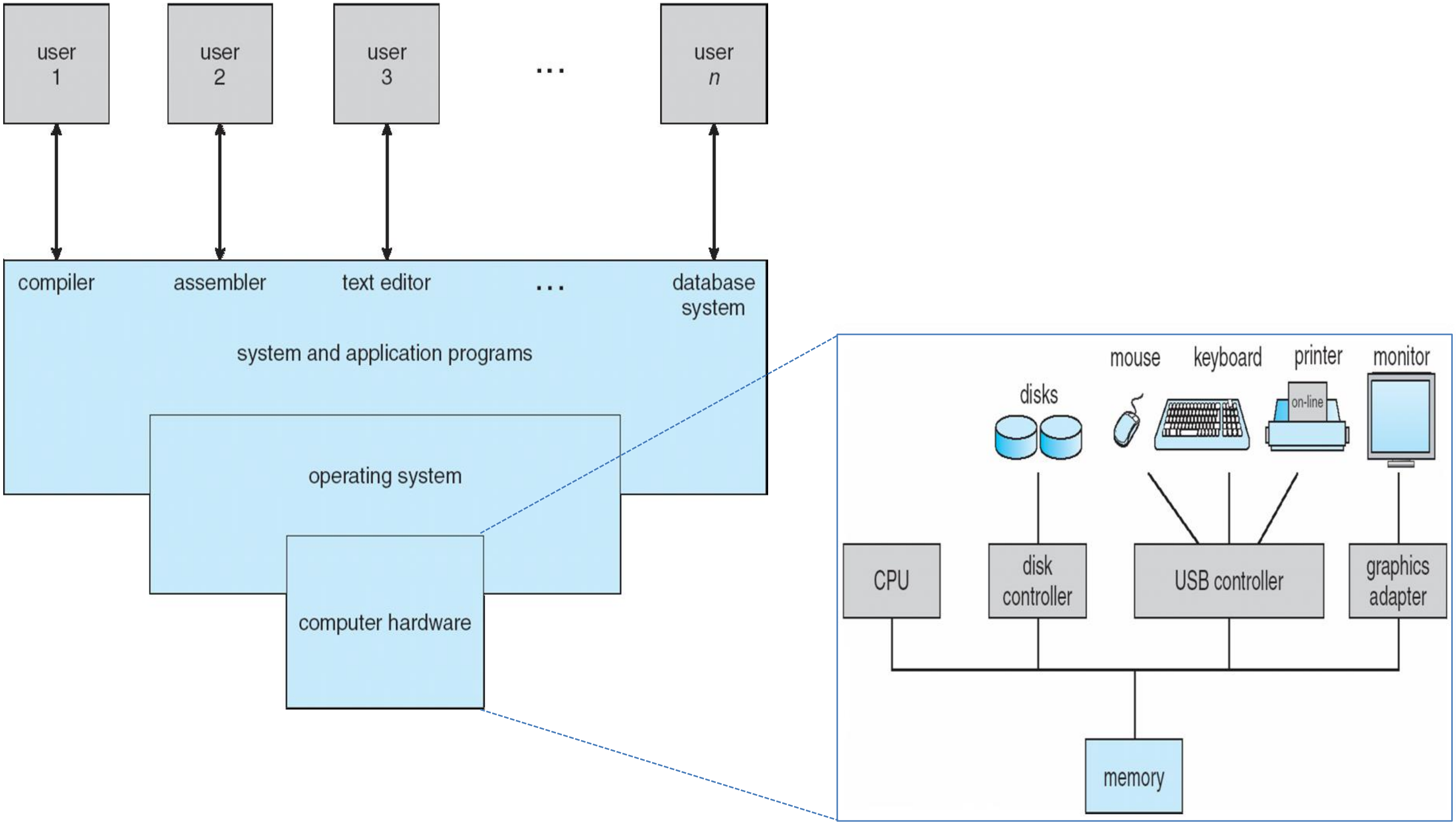
Outline

- Operating System Examples
- Operating System and Computer System
- Operating System Basics
- Operating System Roles
- Operating System Components
- Dual-modes: User and Kernel Modes
- System Calls
- Operating System Architecture

Operating System Examples

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

Operating System and Computer System



Operating System Basics

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

Operating System Roles

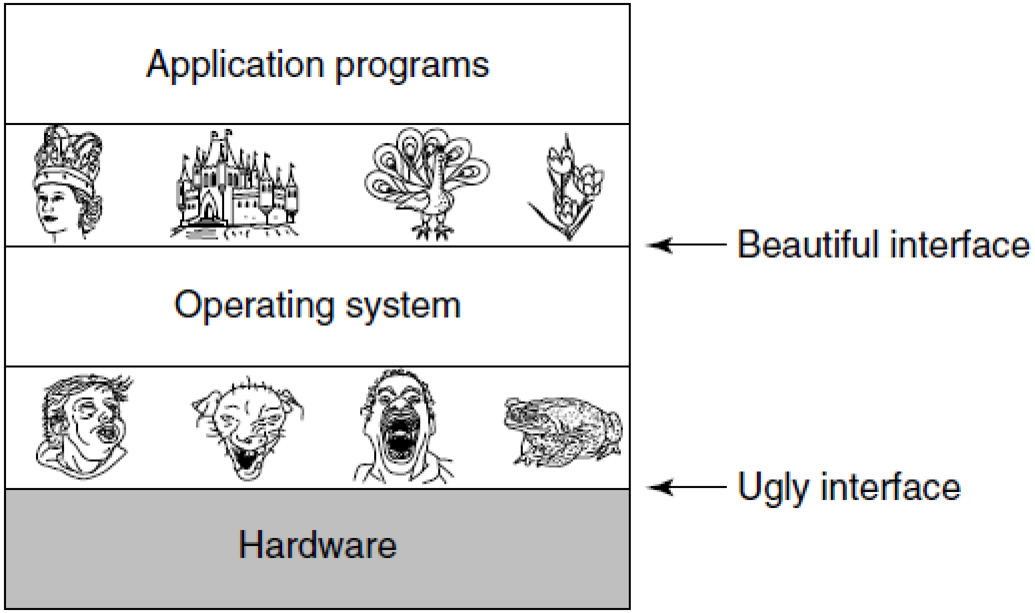
- Two important roles:
  - Provides a **nice abstraction** around the hardware or **extend the machine**.
  - **Manage** the hardware **resources**.

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.

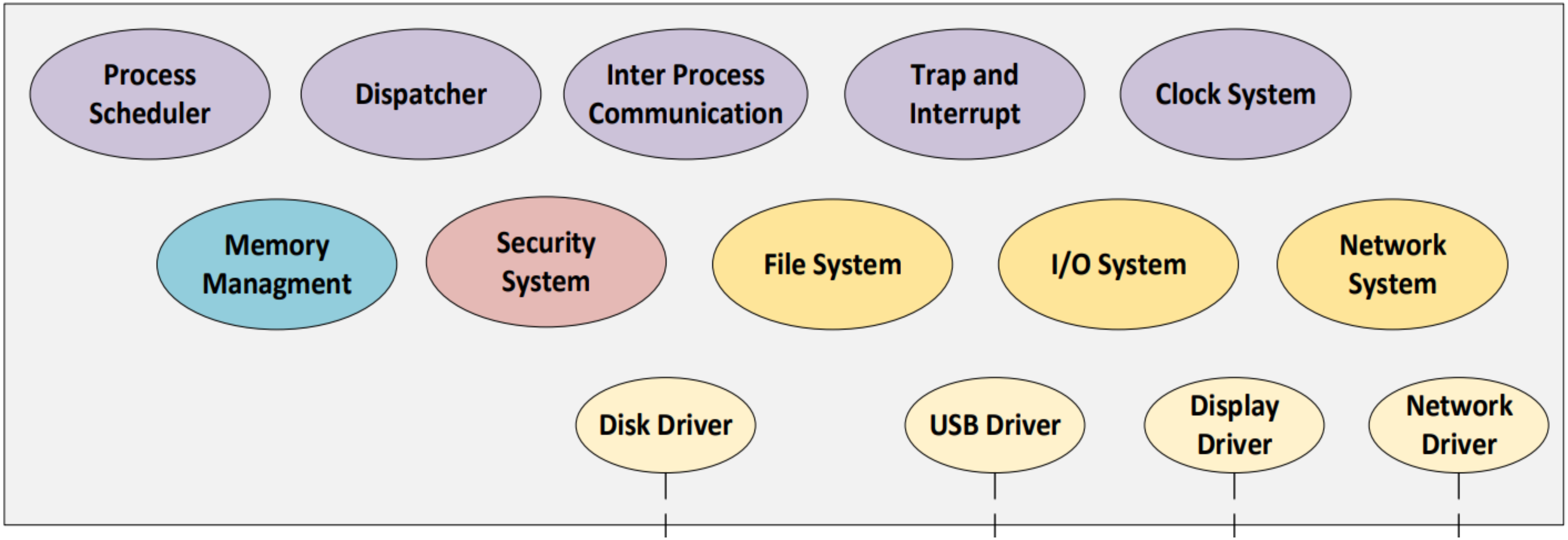


OS Roles: Resource Manager

- Provides **orderly** and **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.



Operating System Components

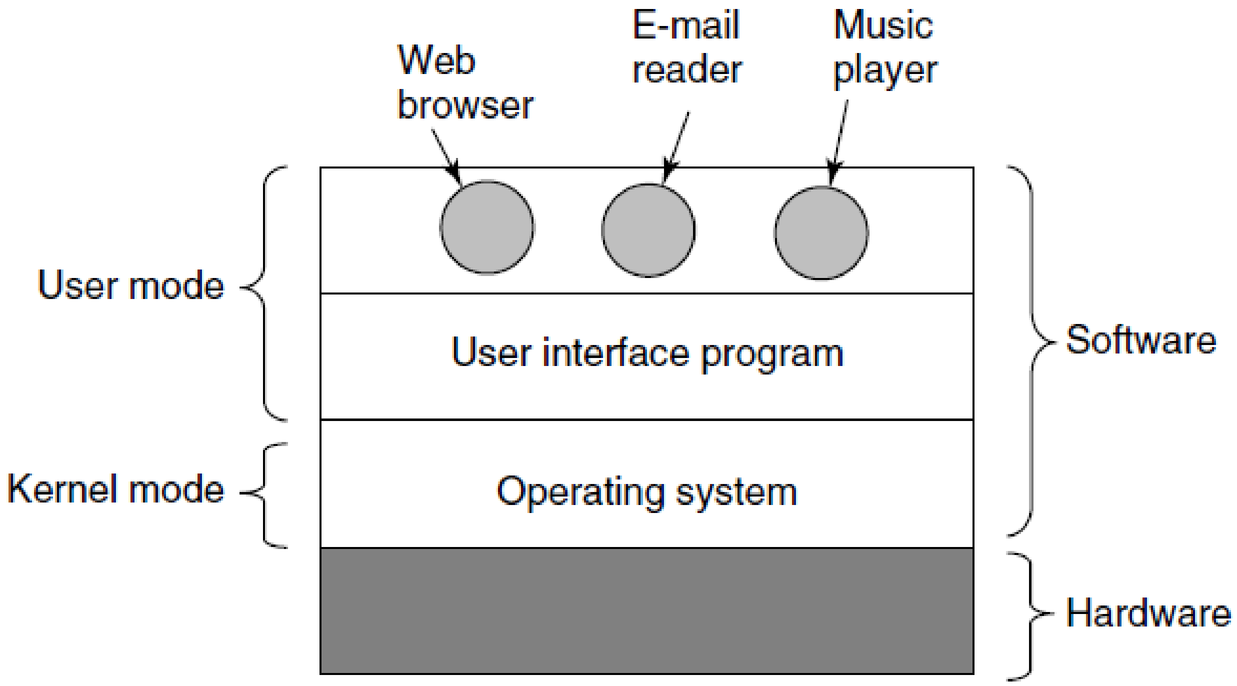
Will learn details about followings components:

- Process Management System
  - Scheduler, Dispatcher, and Inter-process-communication
- Memory Management System
- File System
- I/O System

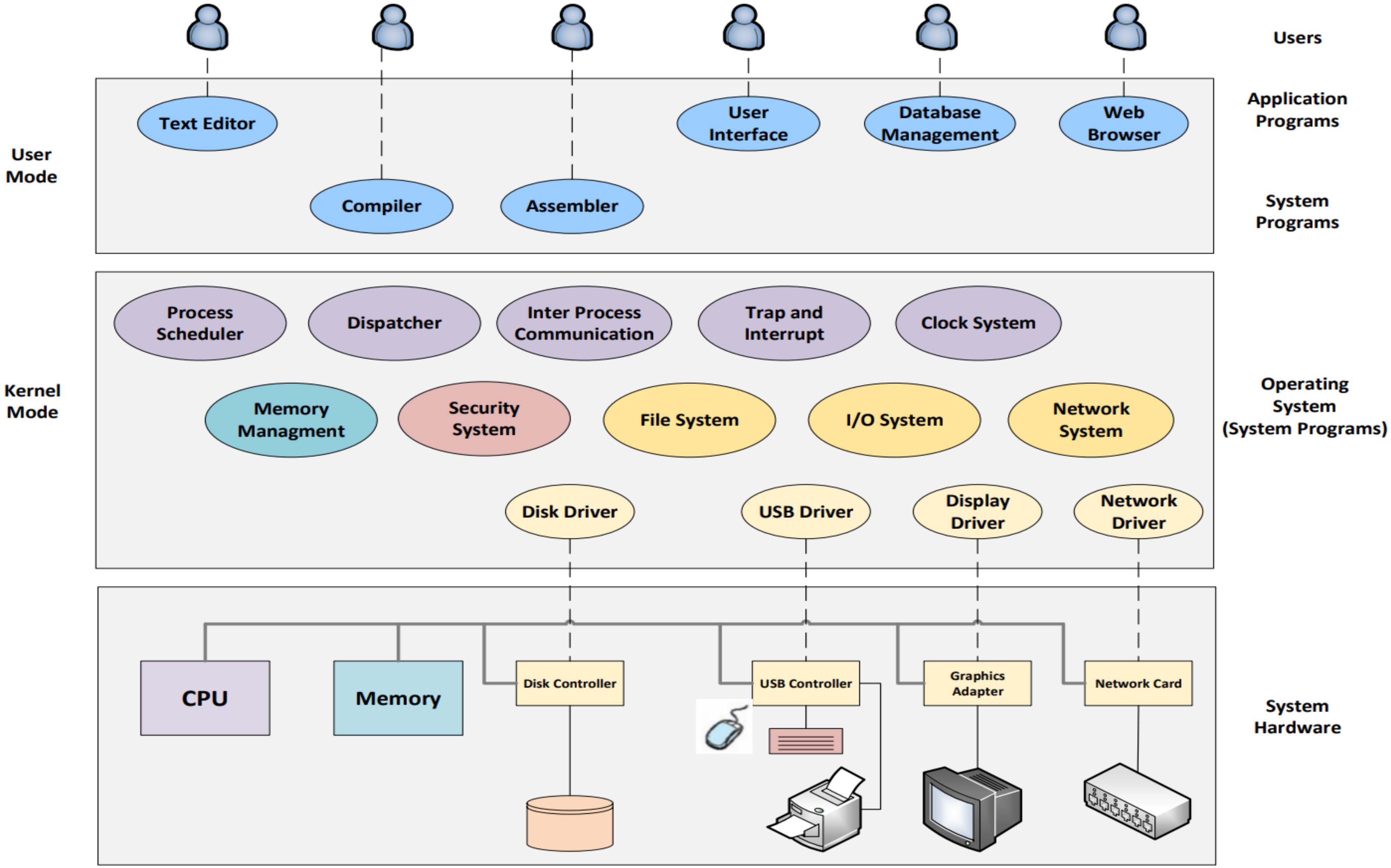
Dual Modes: User and Kernel

- **Dual-modes** (**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.

Dual-modes: User and Kernel



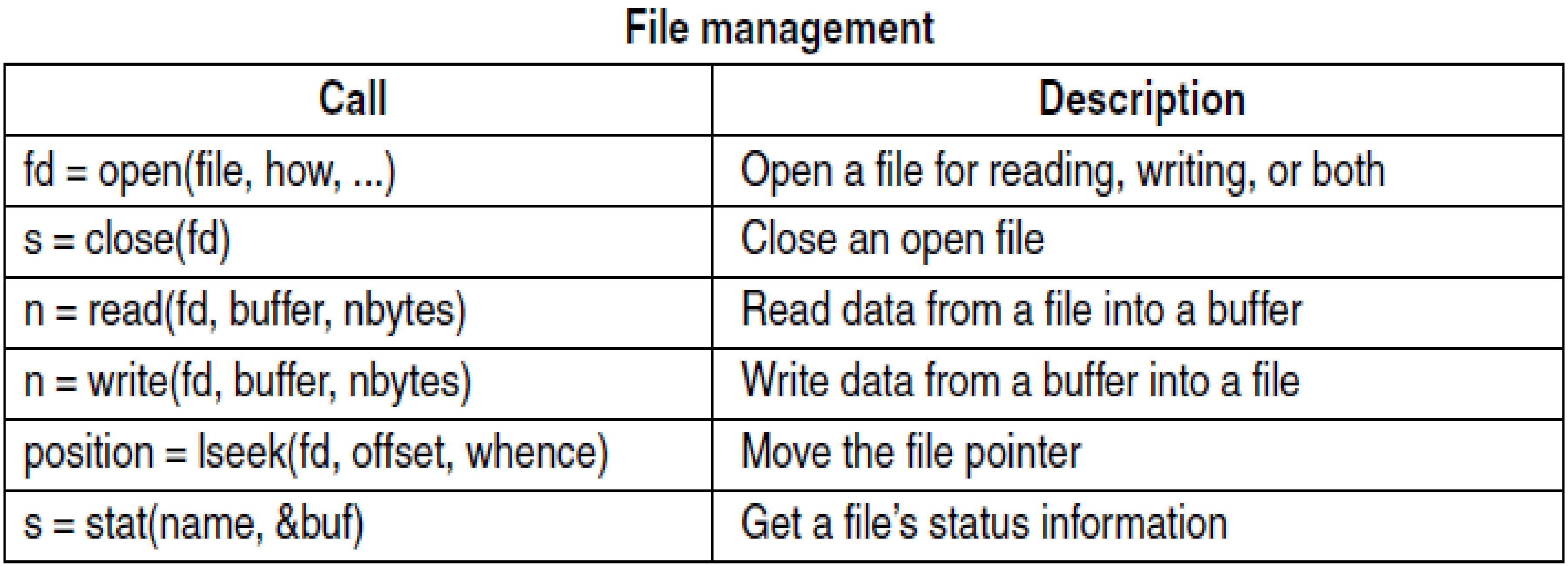
Dual-modes: User and Kernel



System Calls

- **Programs** in **User mode,** can indirectly access hardware resources and execute **privileged** instructions through **system calls.**
- System call changes mode to **kernel**, provides system level service by executing kernel routine, and resets mode to **user** at return.
- **Mode bit** provided by hardware, gives the ability to distinguish when system is running user mode code or kernel mode code.

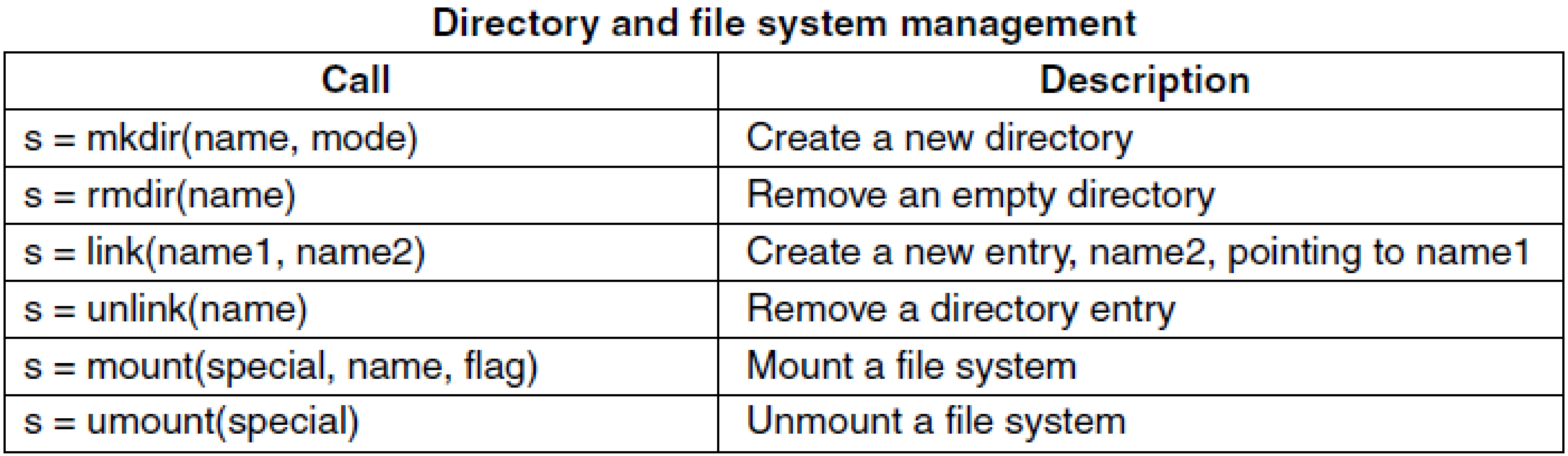
Unix System Call Examples



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

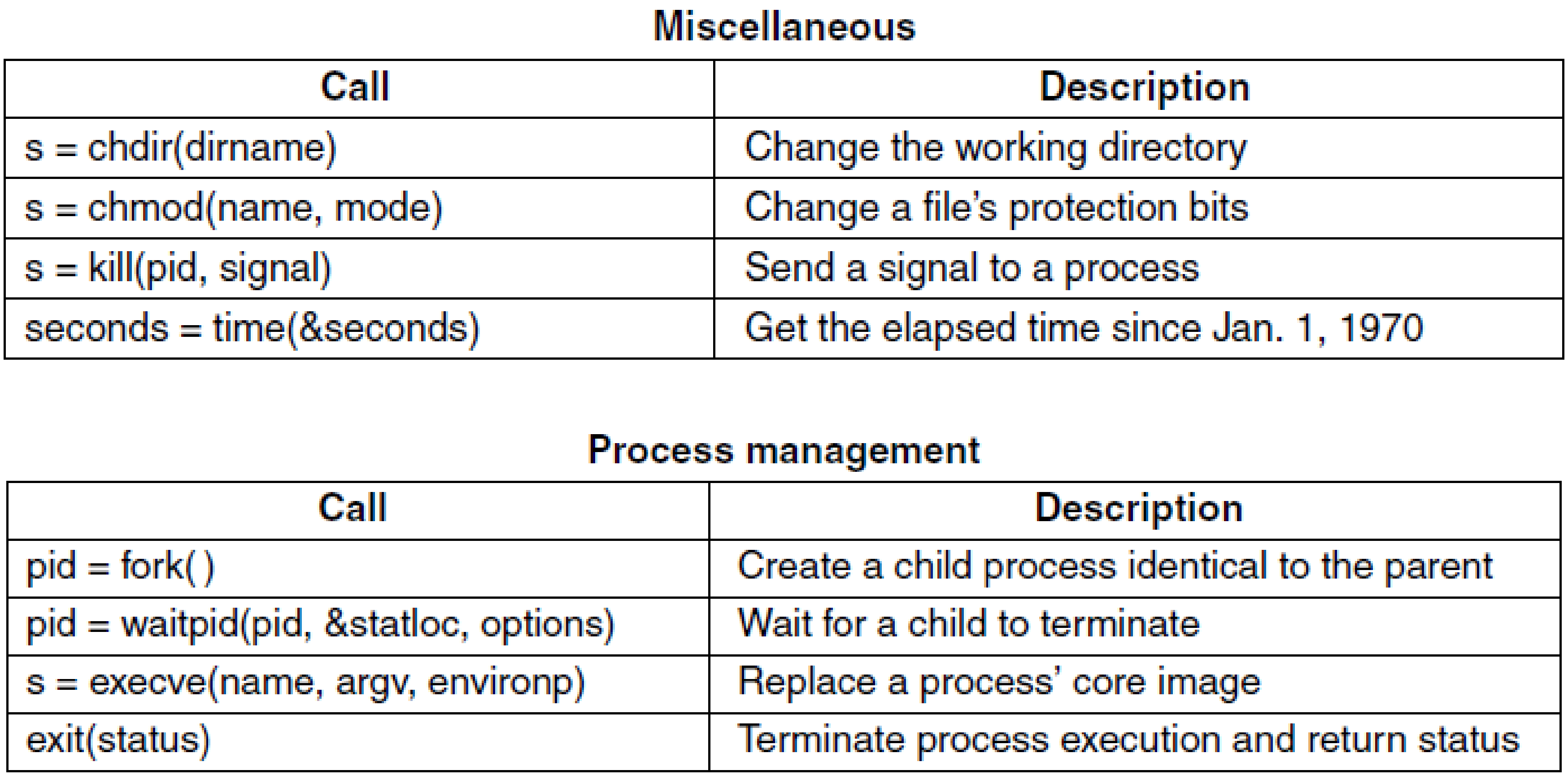
Unix System Call Examples



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

Unix System Call Examples



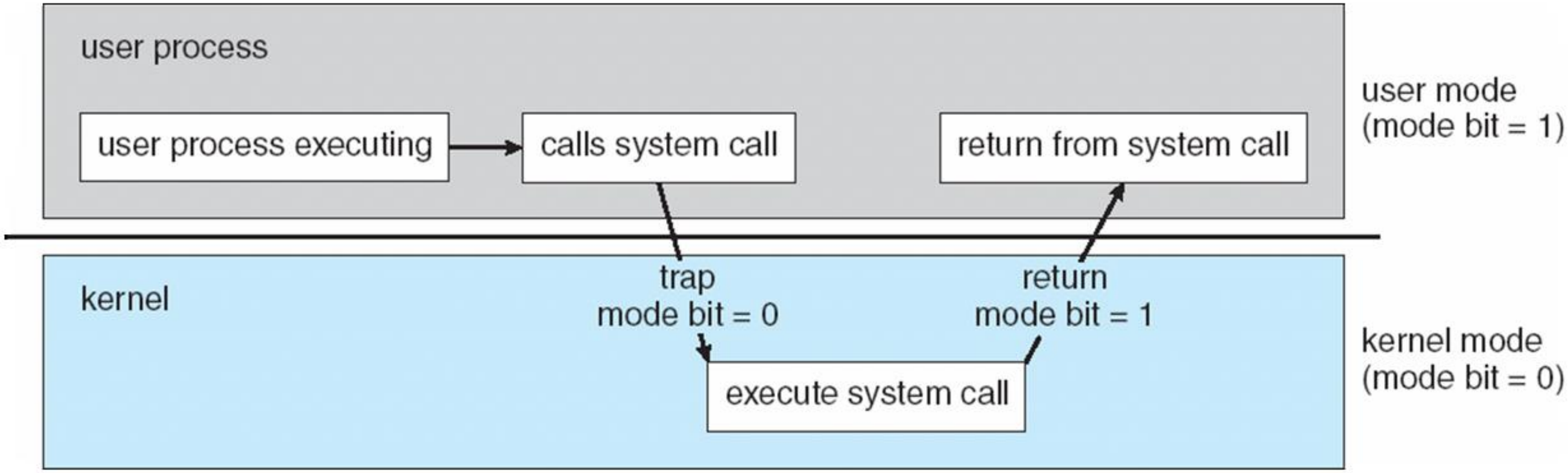
Miscellaneous

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

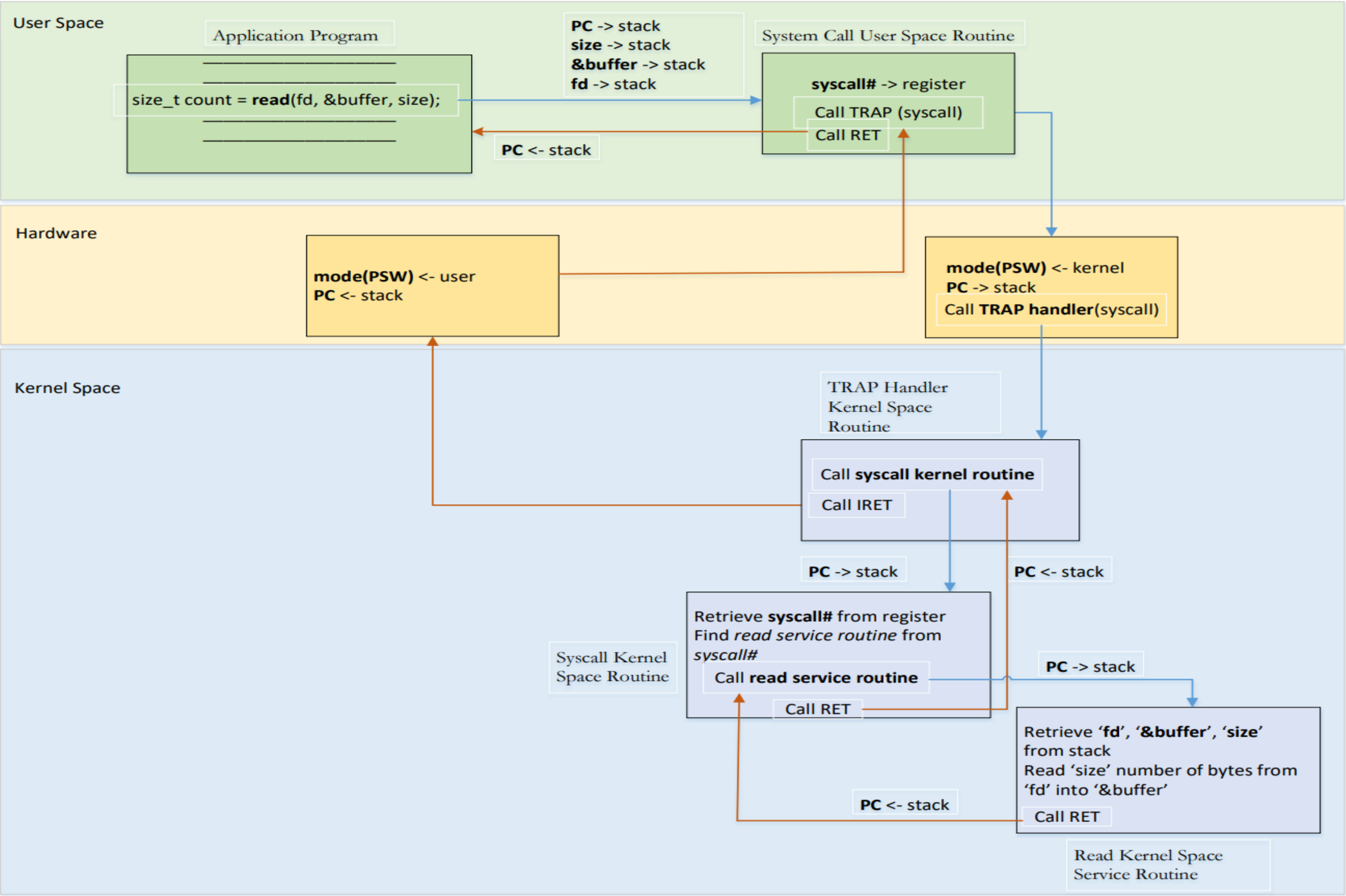
Process management

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

System Call Simplified View



System Call Elaborated



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

Syscall Kernel
Space Routine

Retrieve **syscall#** from register
Find *read service routine* from *syscall#*
Call **read service routine**
Call RET

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

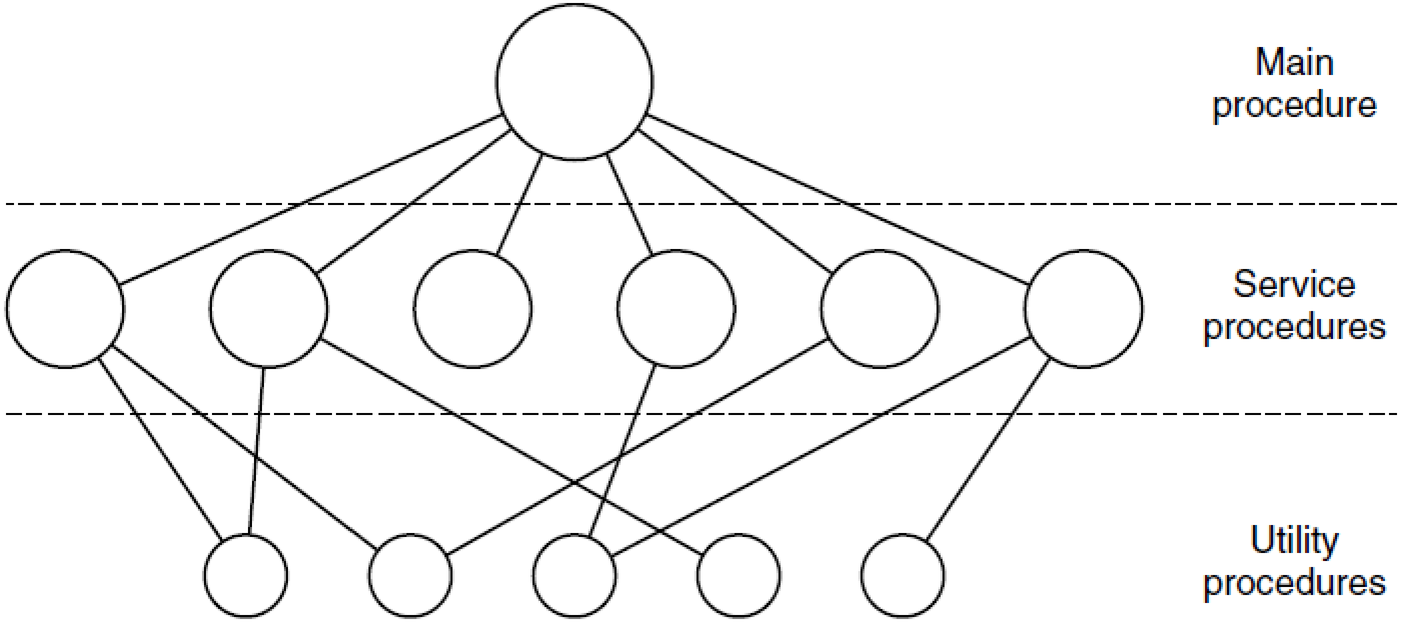
Operating System Architecture

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

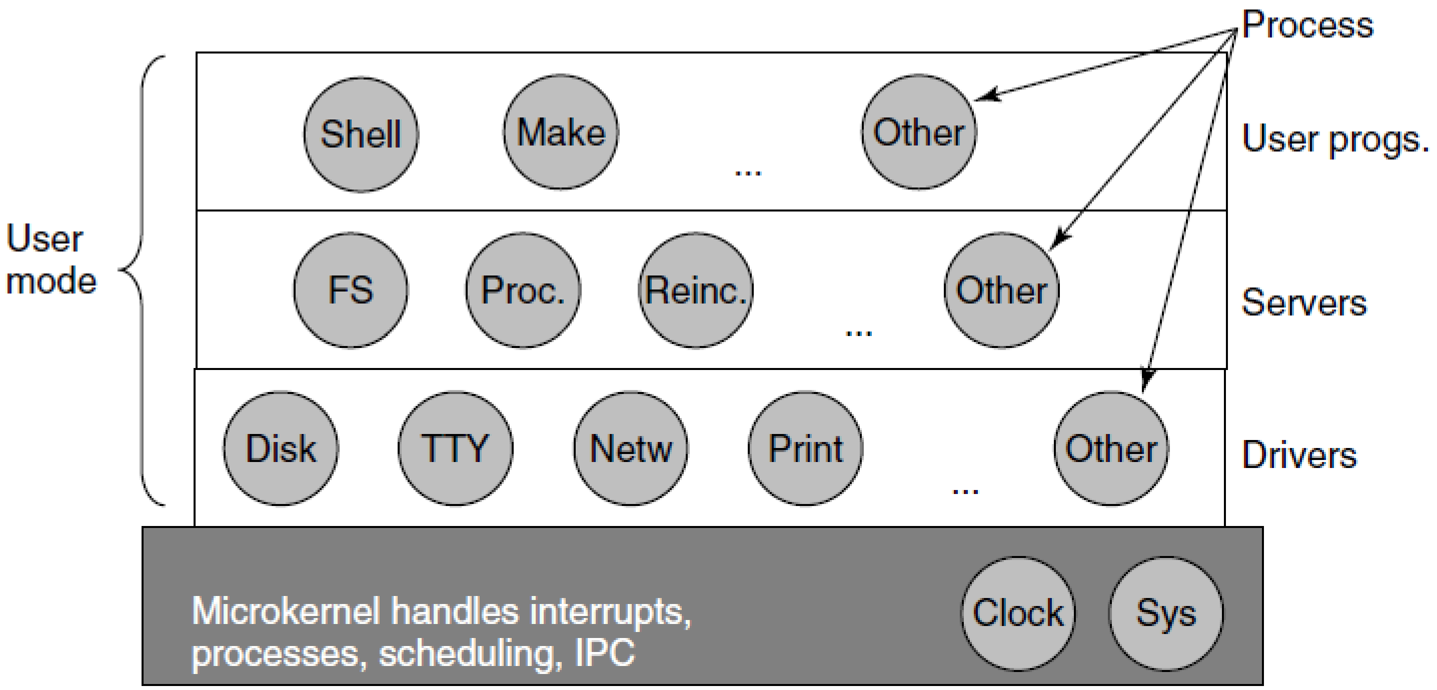
OS Architecture: Monolithic

- Consists of a main procedure, a set of service procedures, and a set of utility procedures.
- Starts with the main procedure that invokes the requested service procedure.
- Service procedures carry out the system calls.
- Utility procedures help service procedures and being called by service procedures.

OS Architecture: Monolithic



OS Architecture: Microkernels



Summary

- Operating System Examples
- Operating System in Computer System
- Operating System Basics
- Operating System Roles
- Operating System Components
- Dual-modes: User and Kernel Modes
- System Calls
- Operating System Architecture

Next

Process Management

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