Operating System Services

- Operating systems provide an environment for execution of programs and services to programs and users
- 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

Operating System Services (Cont.)

- One set of operating-system services provides functions that are helpful to the user (Cont.):
  - **File-system manipulation** - The file system is of particular interest. Programs need to read and write files and directories, create and delete them, search them, list file Information, permission management.
  - **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

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 - CPU cycles, main memory, file storage, I/O devices.

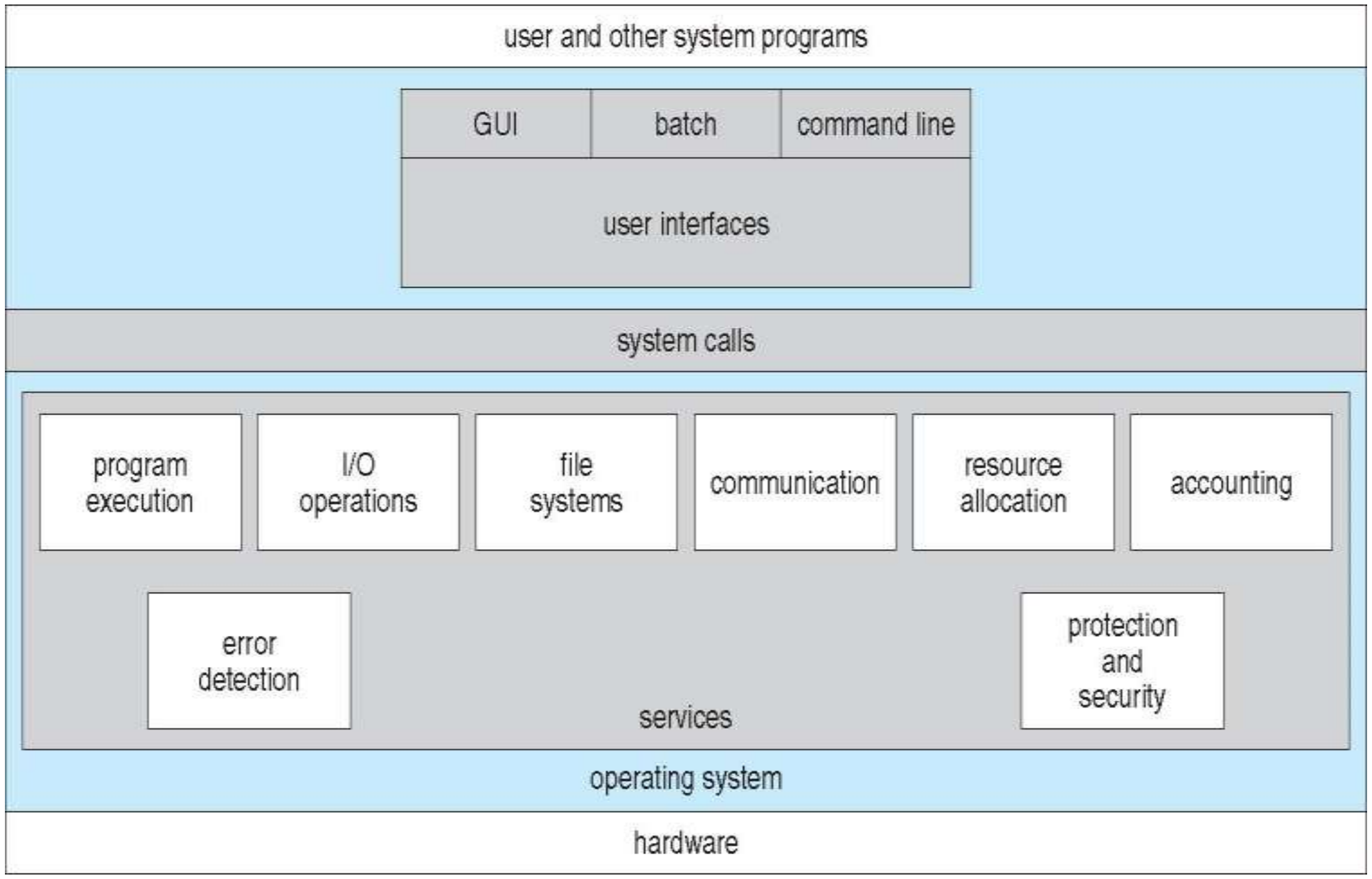
Operating System Services (Cont.)

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

A View of Operating System Services



System calls

- In computing, a **system call** is the programmatic way in which a computer program requests a service from the kernel of the operating system it is executed on.
- A system call is a way for programs to **interact with the operating system**. A computer program makes a system call when it makes a request to the operating system's kernel.
- System call **provides** the services of the operating system to the user programs via Application Program Interface(API).
- It provides an interface between a process and operating system to allow user-level processes to request services of the operating system. System calls are the only entry points into the kernel system. All programs needing resources must use system calls.

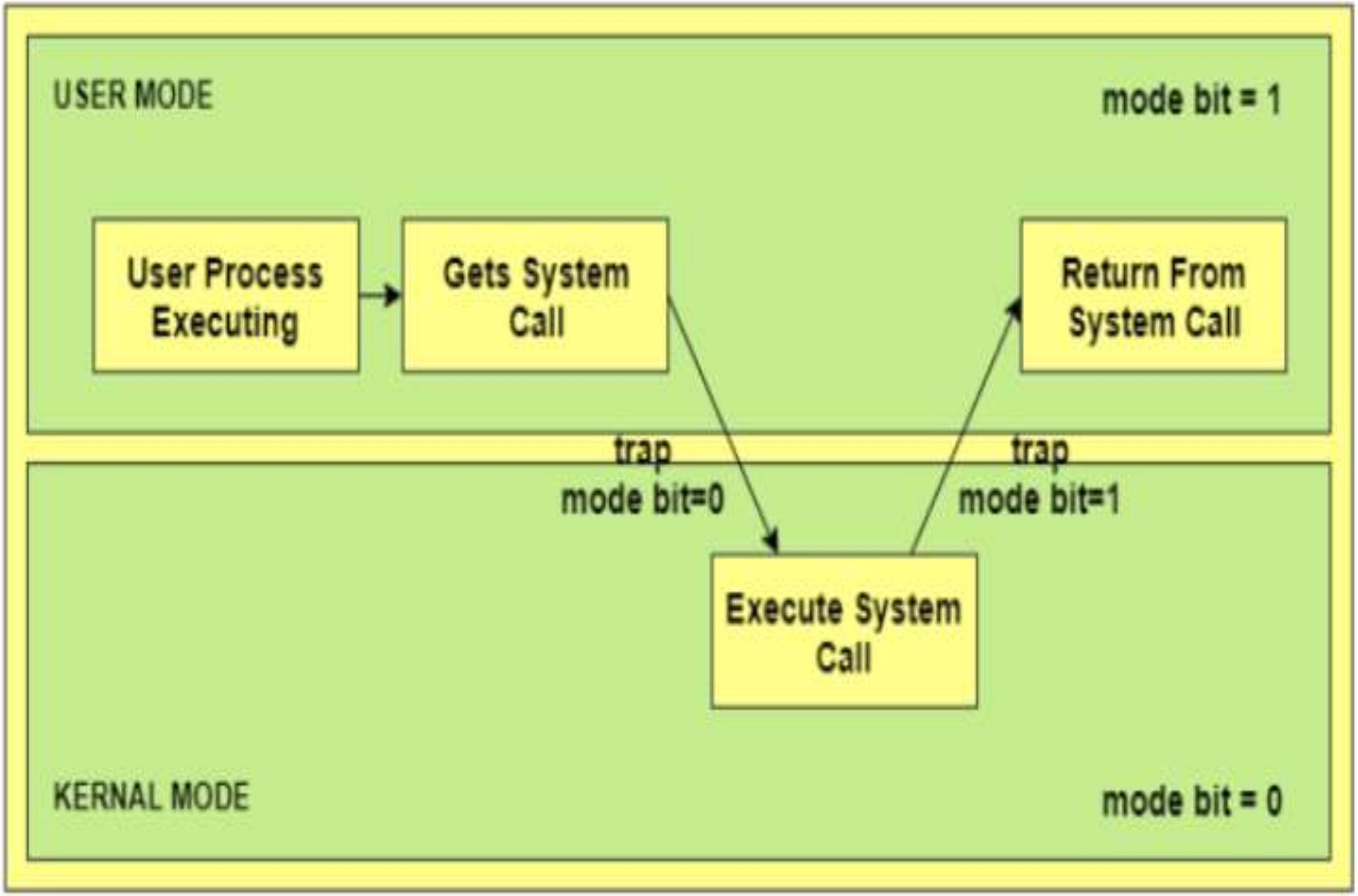
System Call Implementation

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

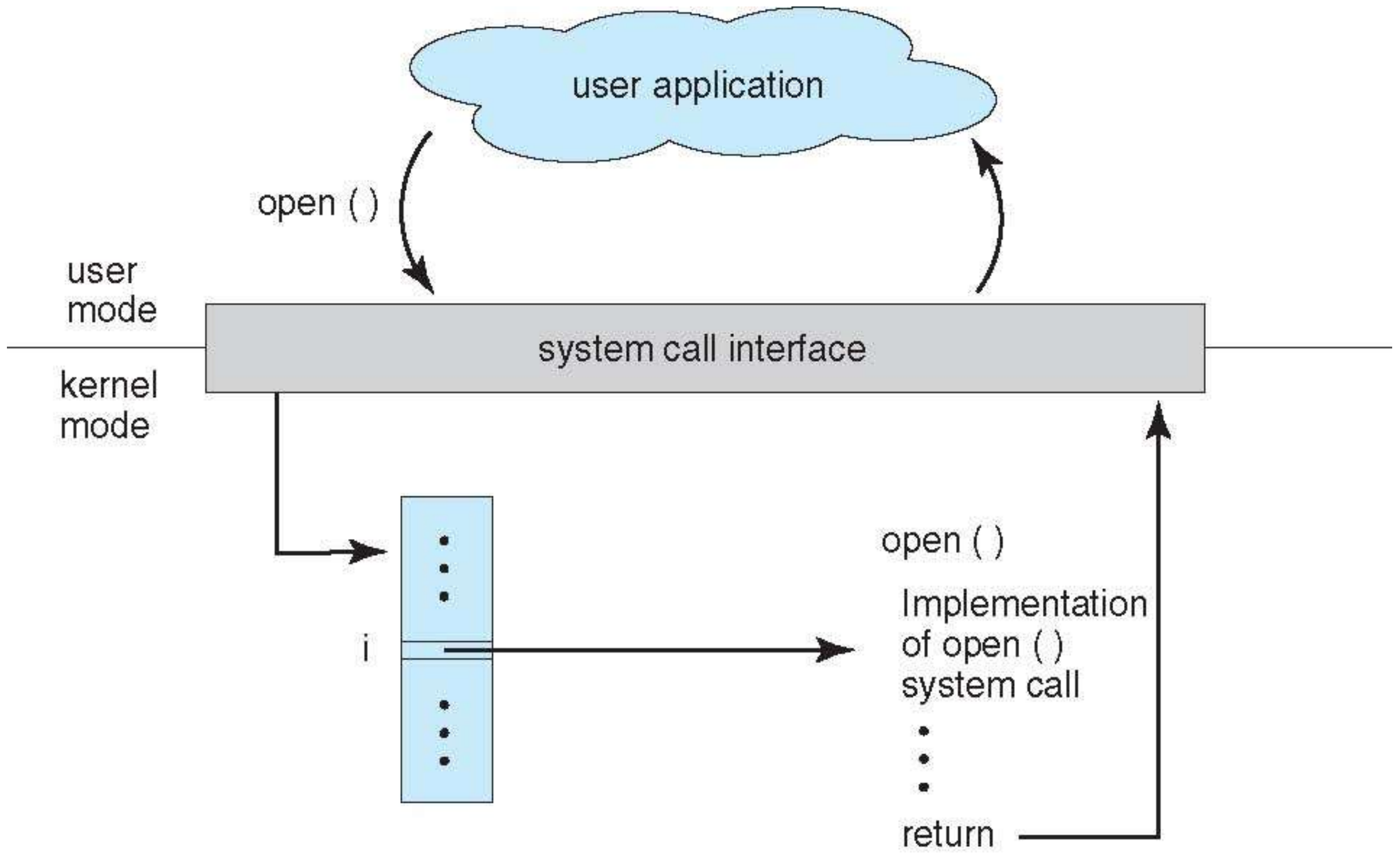
User Mode vs Kernel Mode

- **User Mode**
- The system is in user mode when the operating system is running a user application such as handling a text editor. The transition from user mode to kernel mode occurs when the application requests the help of operating system or an interrupt or a system call occurs.
- The mode bit is set to 1 in the user mode. It is changed from 1 to 0 when switching from user mode to kernel mode.

- **Kernel Mode**
- The system starts in kernel mode when it boots and after the operating system is loaded, it executes applications in user mode. There are some privileged instructions that can only be executed in kernel mode.
- These are interrupt instructions, input output management etc. If the privileged instructions are executed in user mode, it is illegal and a trap is generated.
- The mode bit is set to 0 in the kernel mode. It is changed from 0 to 1 when switching from kernel mode to user mode.


USER MODE
mode bit = 1
User Process Executing
Gets System Call
Return From System Call
trap
mode bit=0
trap
mode bit=1
Execute System Call
KERNAL MODE
mode bit = 0

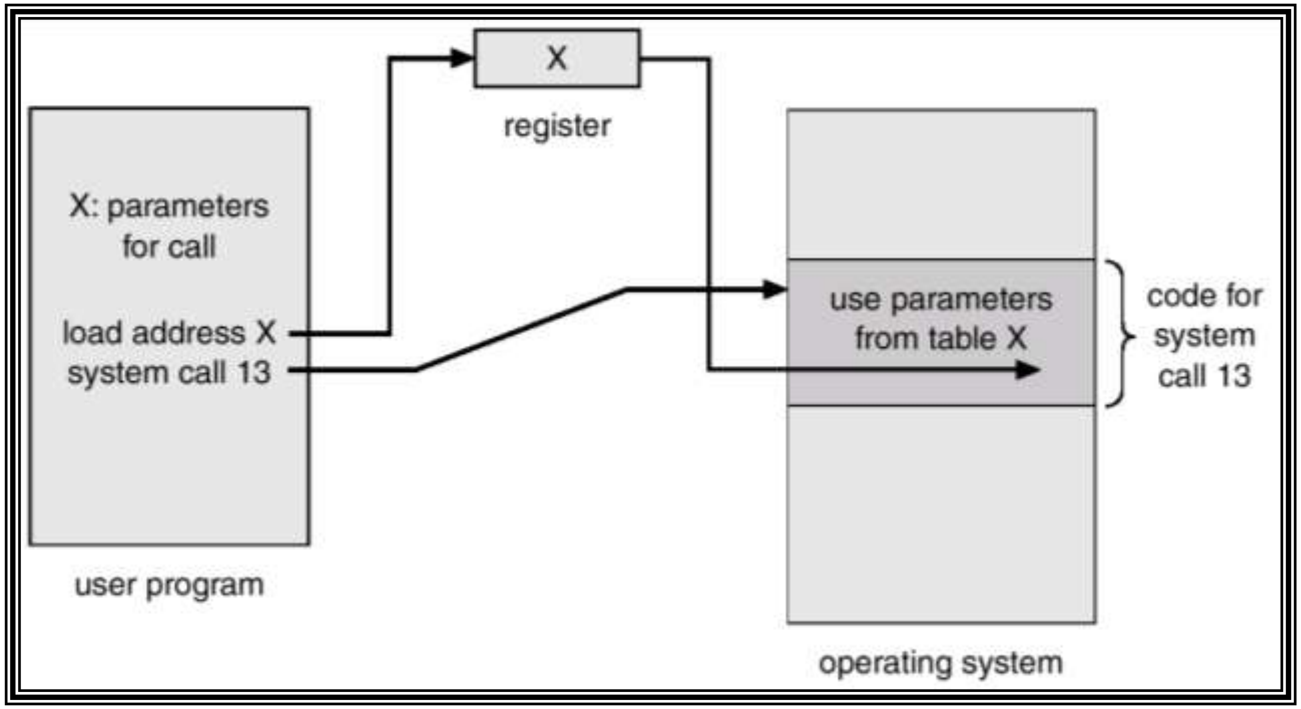
API – System Call – OS Relationship



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
  - Block and stack methods do not limit the number or length of parameters being passed

Passing of Parameters As A Table



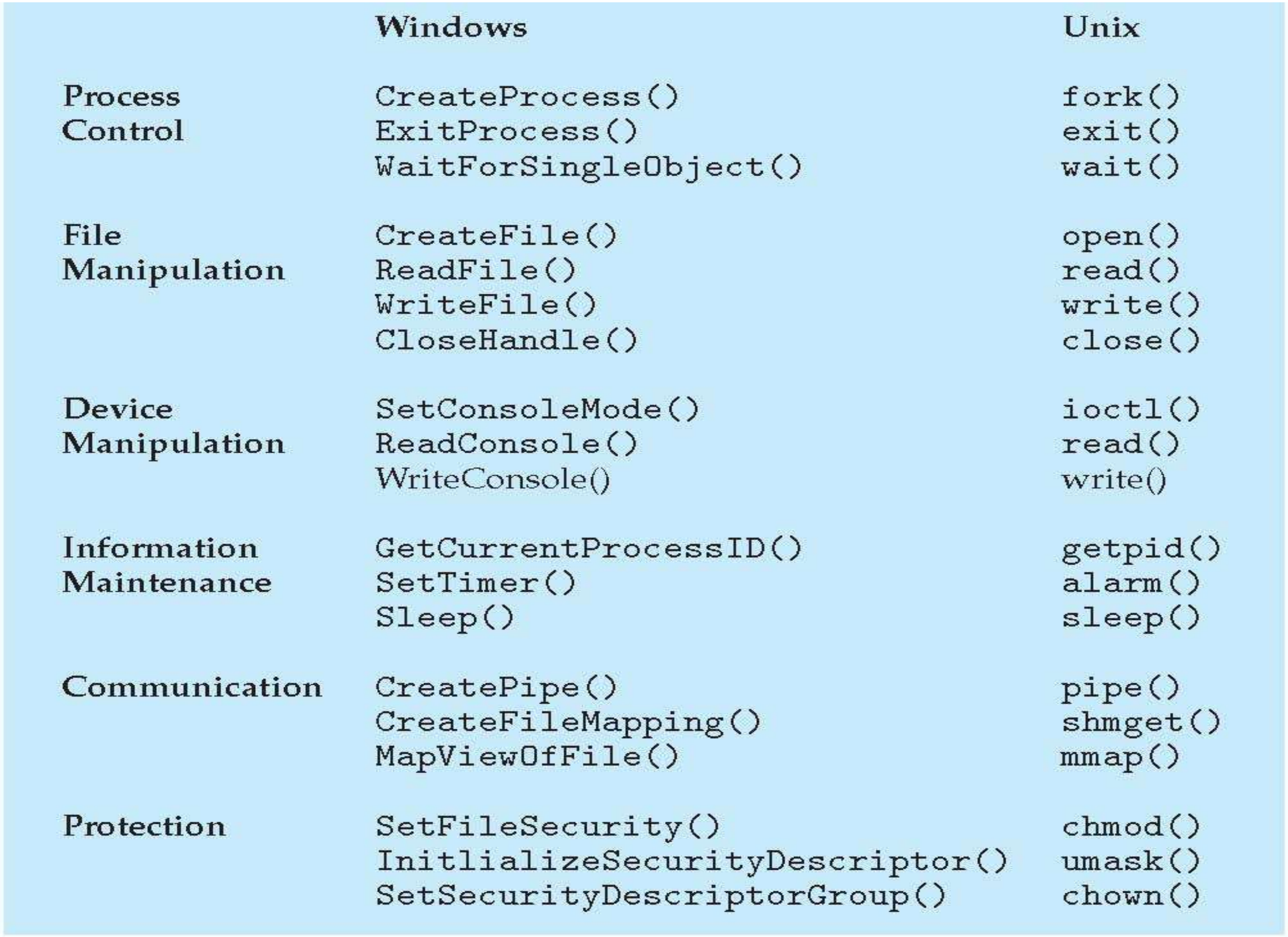
Types of System Calls

- Process control
  - End, abort, load, create, terminate, wait event, signal event, allocate and free memory
- File management
  - Create file, delete, open, close, read, write, get file attributes, set file attributes
- Device management
  - Request device, release device
- Information maintenance
  - Get time or date, set time, get process attributes, Set process attributes , get and set system data
- Communications
  - Send , receive message, transfer status information, create and delete communication connection

Services Provided by System Calls :

- Process creation and management
- Main memory management
- File Access, Directory and File system management
- Device handling(I/O)
- Protection
- Networking, etc

Examples of Windows and Unix System Calls



	Windows	Unix
Process Control	CreateProcess()	fork()
	ExitProcess()	exit()
	WaitForSingleObject()	wait()
File Manipulation	CreateFile()	open()
	ReadFile()	read()
	WriteFile()	write()
	CloseHandle()	close()
Device Manipulation	SetConsoleMode()	ioctl()
	ReadConsole()	read()
	WriteConsole()	write()
Information Maintenance	GetCurrentProcessID()	getpid()
	SetTimer()	alarm()
	Sleep()	sleep()
Communication	CreatePipe()	pipe()
	CreateFileMapping()	shmget()
	MapViewOfFile()	mmap()
Protection	SetFileSecurity()	chmod()
	InitlializeSecurityDescriptor()	umask()
	SetSecurityDescriptorGroup()	chown()