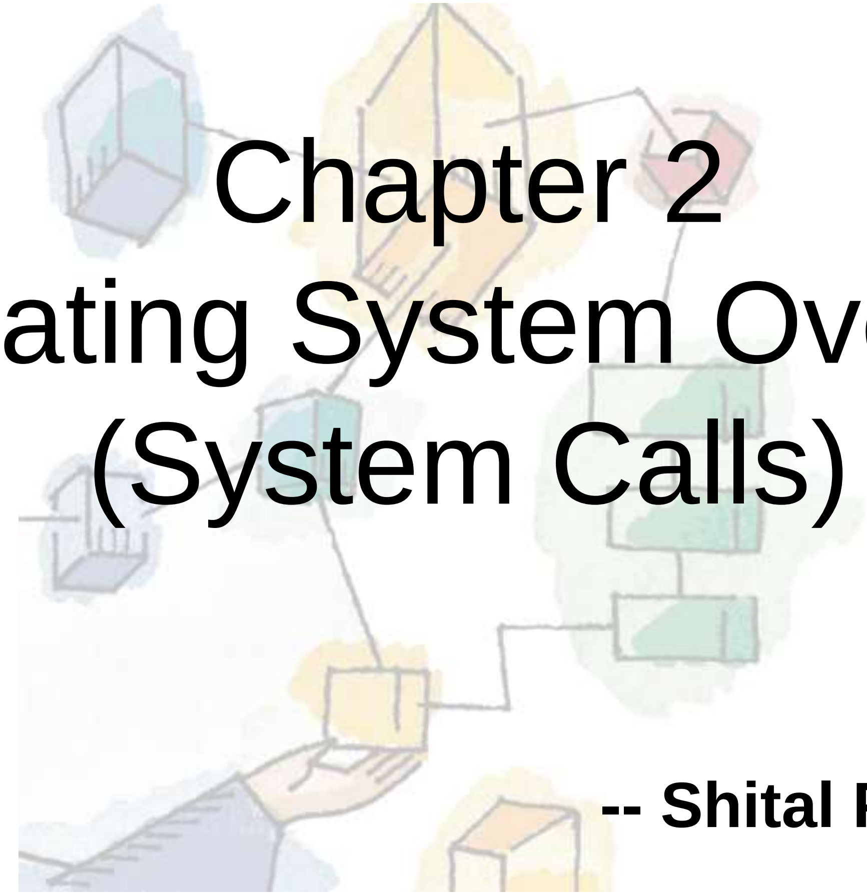
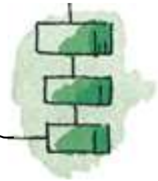


# Chapter 2
# Operating System Overview
# (System Calls)

**-- Shital Pathar**

# System Calls

- **Definition:**
  - *Programming interface* to the services provided by the OS
- Typically written in a *high-level language* (C or C++)
- **Use:**
- Mostly accessed by programs via a high-level **Application Programming Interface (API)** rather than direct system call use

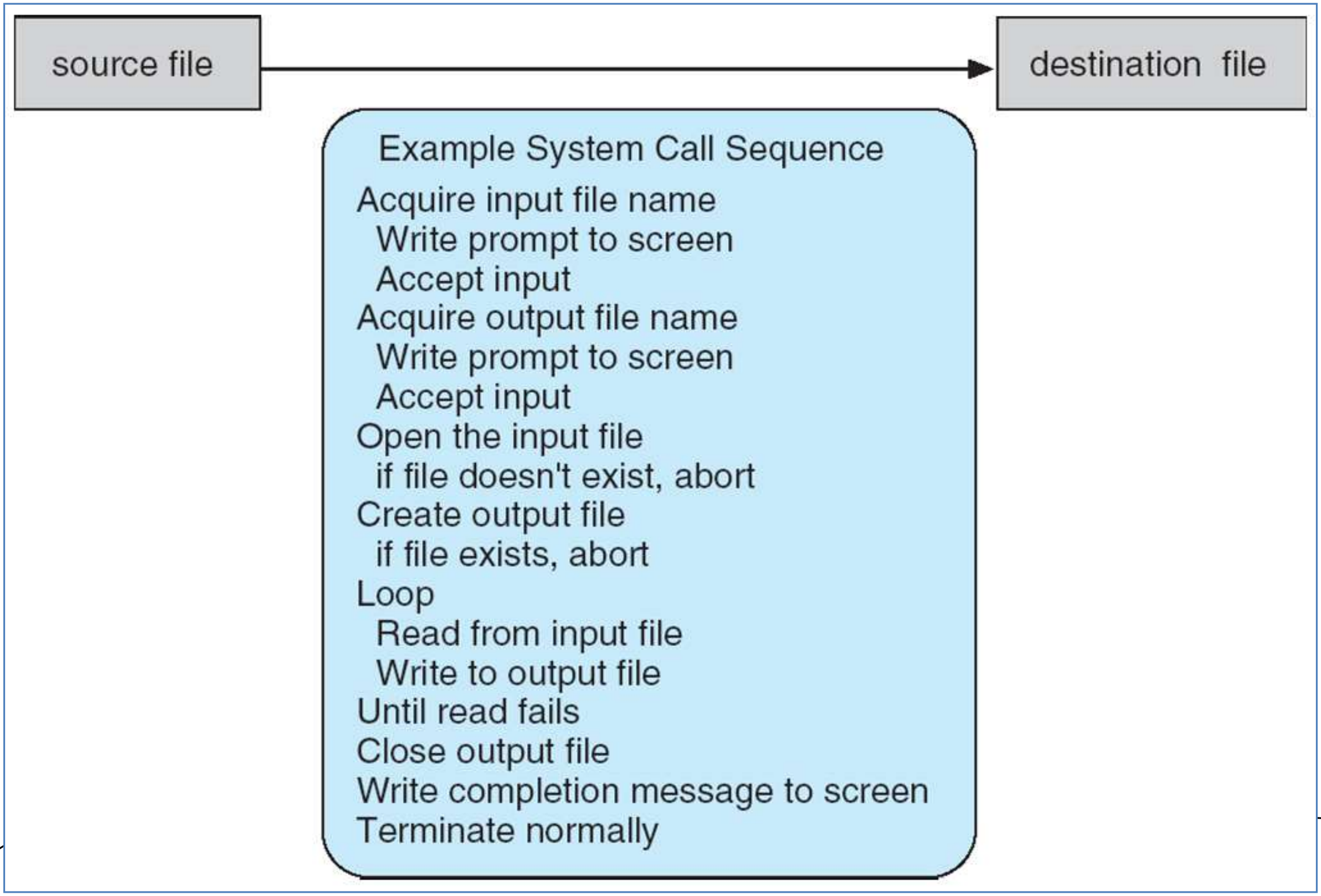


# System Calls

- **Three most common APIs are**
  - Win32 API for Windows,
  - POSIX API for POSIX-based systems (including virtually all versions of UNIX, Linux, and Mac OS X),
  - Java API for the Java virtual machine (JVM)

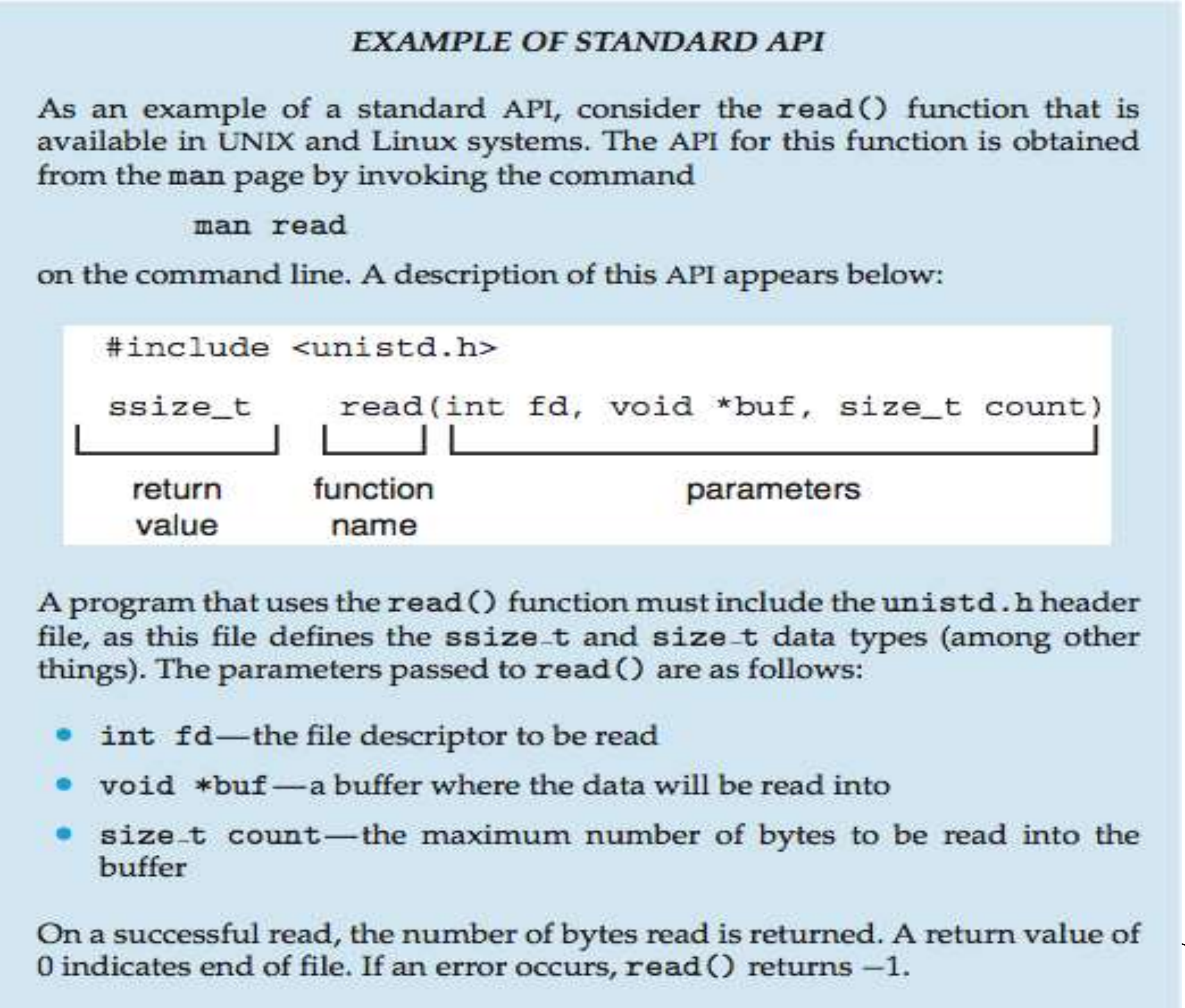
# Example of System Calls

**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
```



### *EXAMPLE OF STANDARD API*

As an example of a standard API, consider the `read()` function that is available in UNIX and Linux systems. The API for this function is obtained from the `man` page by invoking the command

```
man read
```

on the command line. A description of this API appears below:

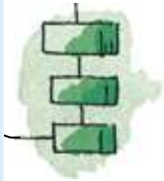
```
#include <unistd.h>

ssize_t    read(int fd, void *buf, size_t count)
```

return value
function name
parameters

A program that uses the `read()` function must include the `unistd.h` header file, as this file defines the `ssize_t` and `size_t` data types (among other things). The parameters passed to `read()` are as follows:

- `int fd`—the file descriptor to be read
- `void *buf`—a buffer where the data will be read into
- `size_t count`—the maximum number of bytes to be read into the buffer

On a successful read, the number of bytes read is returned. A return value of 0 indicates end of file. If an error occurs, `read()` returns −1.



# System Call Implementation

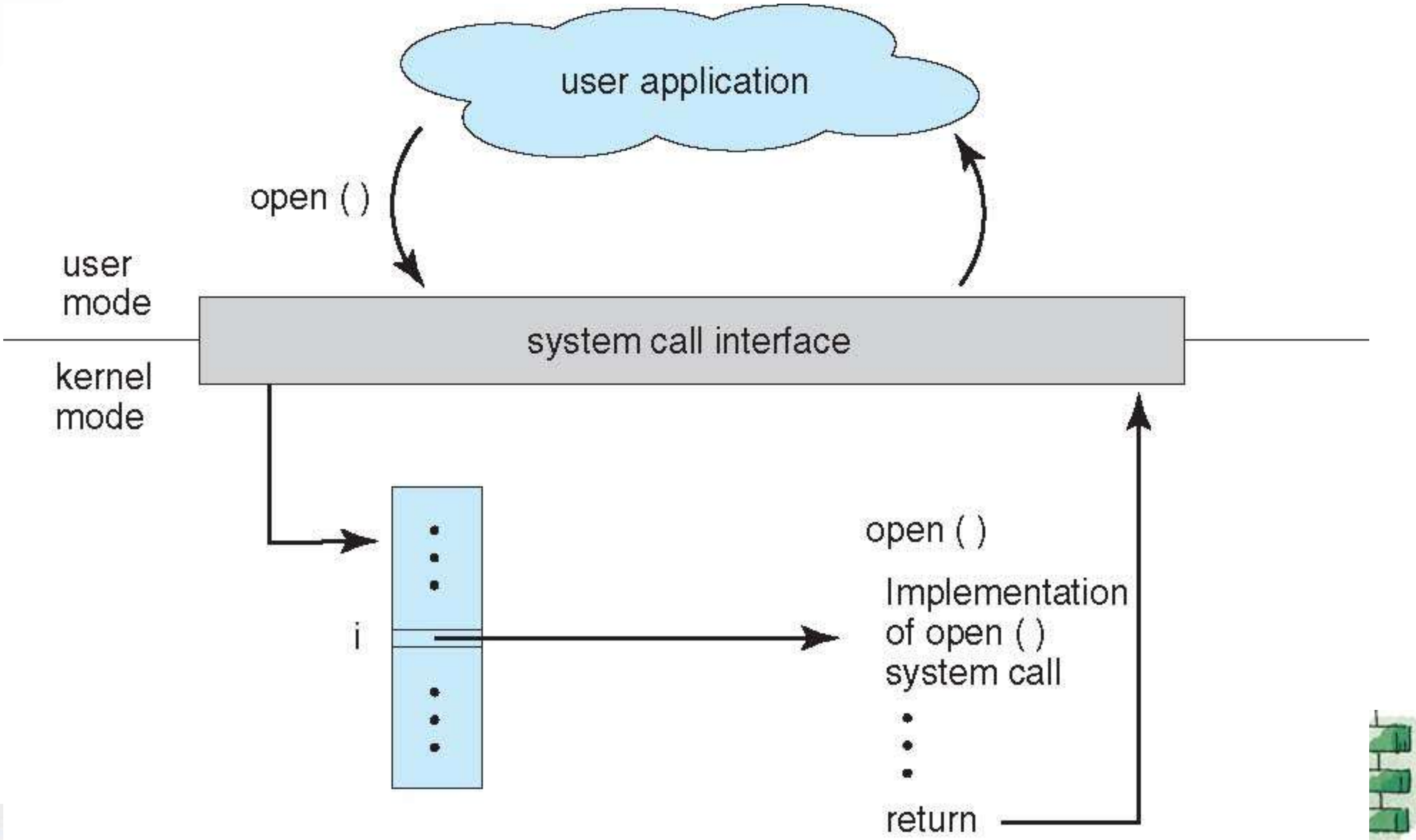
- Typically, a number associated with each system call
  - **System-call interface** maintains a table indexed according to these numbers

- The system call interface invokes  the intended system call in OS kernel and returns status of the system call and any return values

# System Call Implementation

- *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)

# API – System Call – OS Relationship



# Types of System Calls

- **File management**
  - create file, delete file
  - open, close file
  - read, write, reposition
  - get and set file attributes
- **Device management**
  - request device, release device
  - read, write, reposition
  - get device attributes, set device attributes
  - logically attach or detach devices

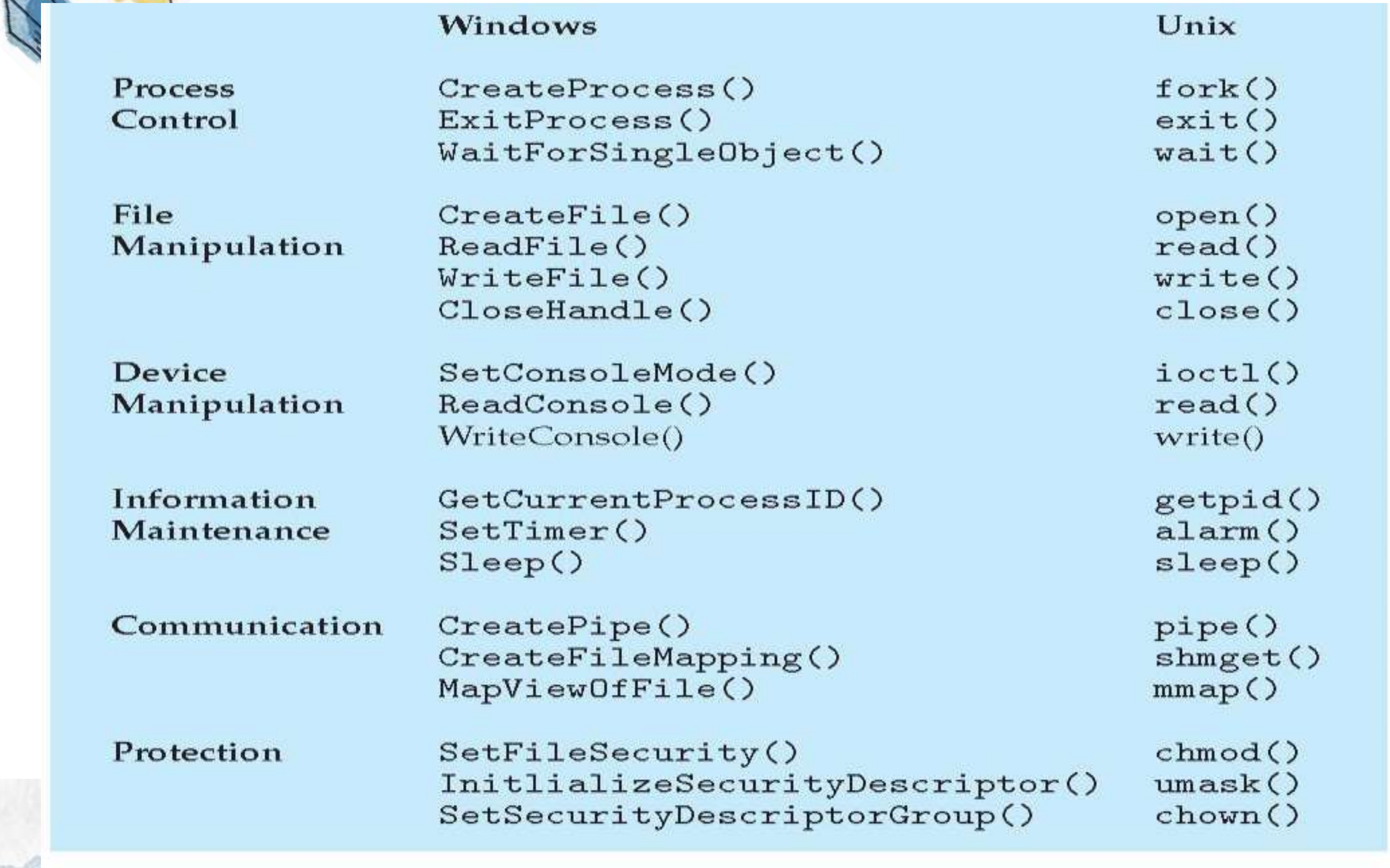
# Types of System Calls

- **Information maintenance**
  - get time or date, set time or date
  - get system data, set system data
  - get and set process, file, or device attributes
- **Communications**
  - create, delete communication connection
  - send, receive messages
  - transfer status information
  - attach and detach remote devices

# Types of System Calls

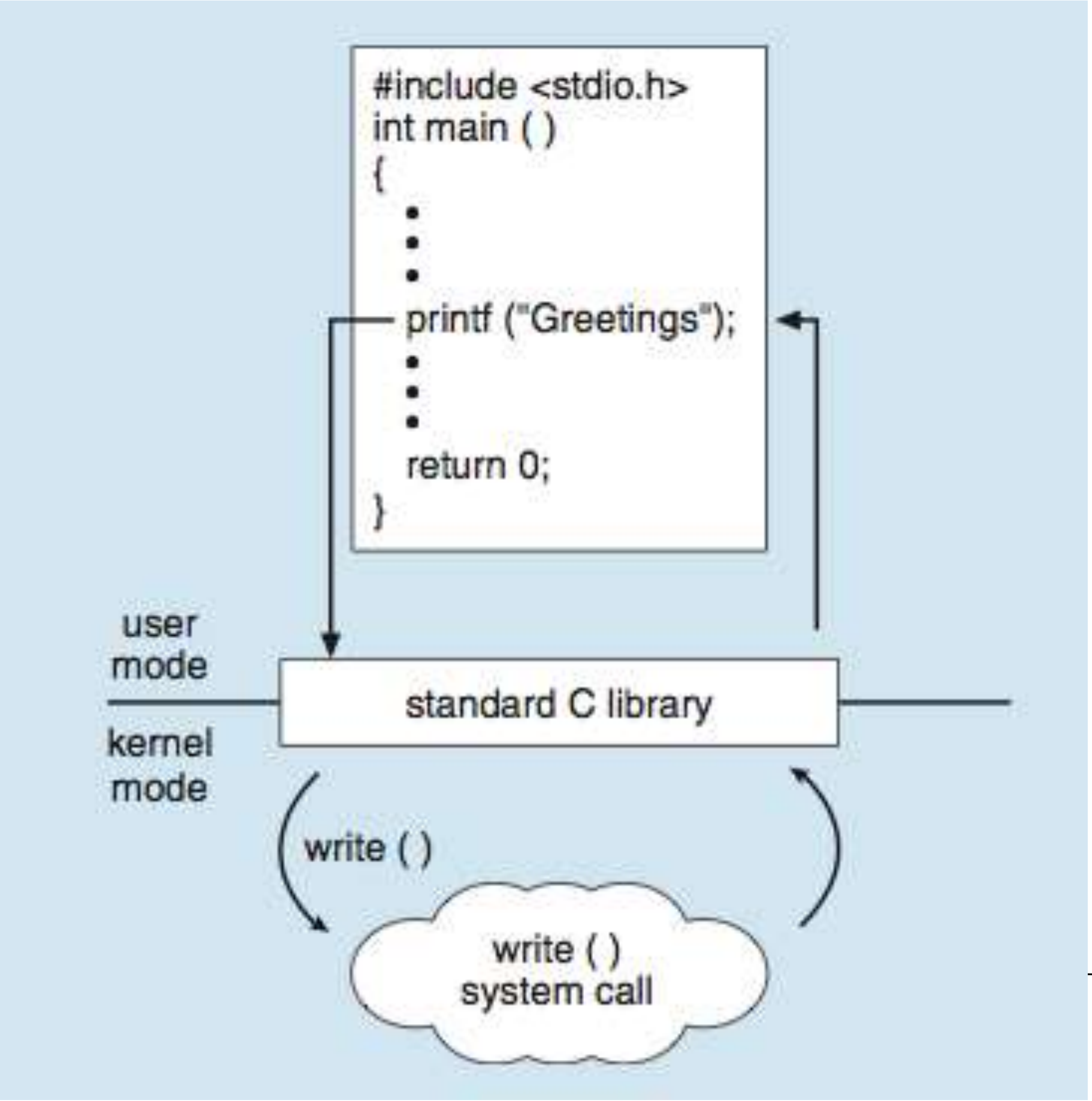
- **Protection**
  - Control access to resources
  - Get and set permissions
  - Allow and deny user access

# 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() |

# Standard C Library Example



```
#include <stdio.h>
int main ( )
{
   •
   •
   •
   printf ("Greetings");
   •
   •
   •
   return 0;
}
```

# References

- Operating Systems: Internals and Design Principles by William Stallings (6th Edition)
- Operating System Concepts by Silberschatz, Galvin and Gagne (9th Edition)