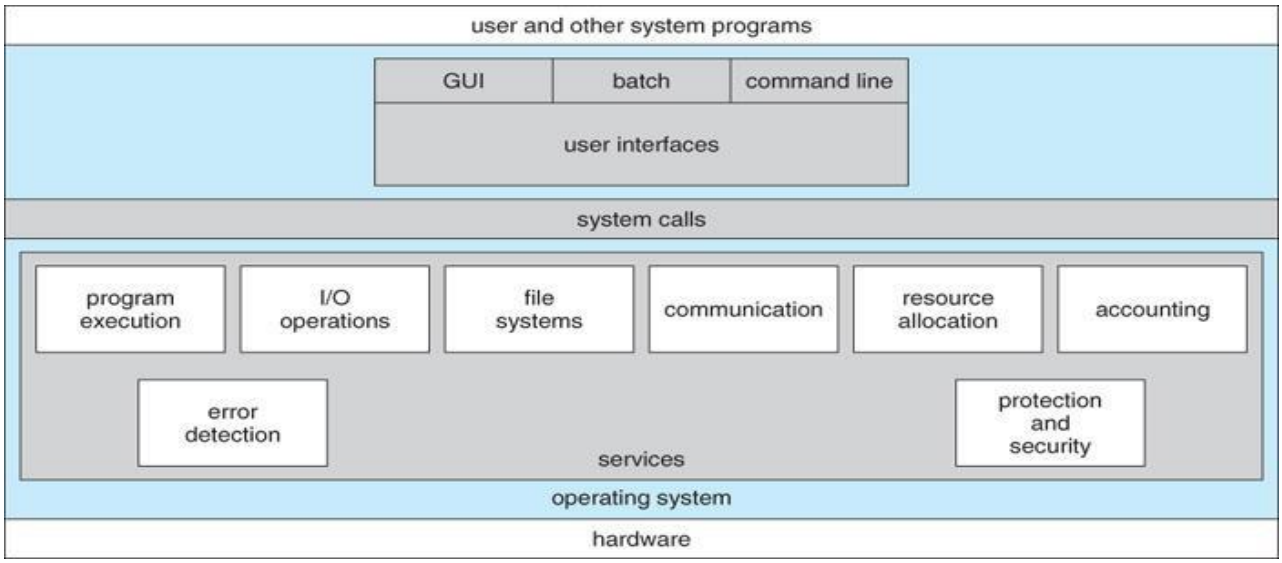
Operating System Services



- One set of operating-system services provides functions that are helpful to the user

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 - 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.

User Operating System Interface - CLI

- Command Line Interface (CLI) or command interpreter allows direct command entry

Sometimes implemented in kernel, sometimes by systems program
sometimes multiple flavors implemented – shells
Primarily fetches a command from user and executes it

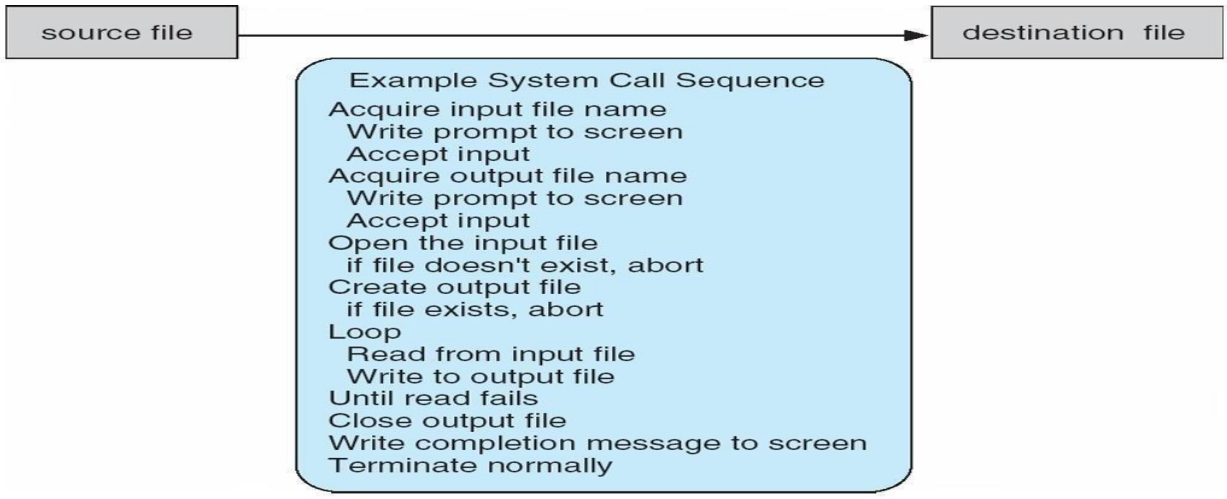
User Operating System Interface - GUI

- 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)

System Calls

- Programming interface to the services provided by the OS
- Typically written in a high-level language (C or C++)
- Mostly accessed by programs via a high-level Application Program Interface (API) rather than direct system call usenThree 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), and Java API for the Java virtual machine (JVM)
- Why use APIs rather than system calls?

Example of System Calls



Example of Standard API

Consider the ReadFile() function in the
Win32 API—a function for reading from a file

A description of the parameters passed to ReadFile() HANDLE file—the file to be read
LPVOID buffer—a buffer where the data will be read into and written from DWORD bytesToRead—the number of bytes to be read into the buffer LPDWORD bytesRead—the number of bytes read during the last read LPOVERLAPPED ovl—indicates if overlapped I/O is being used

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

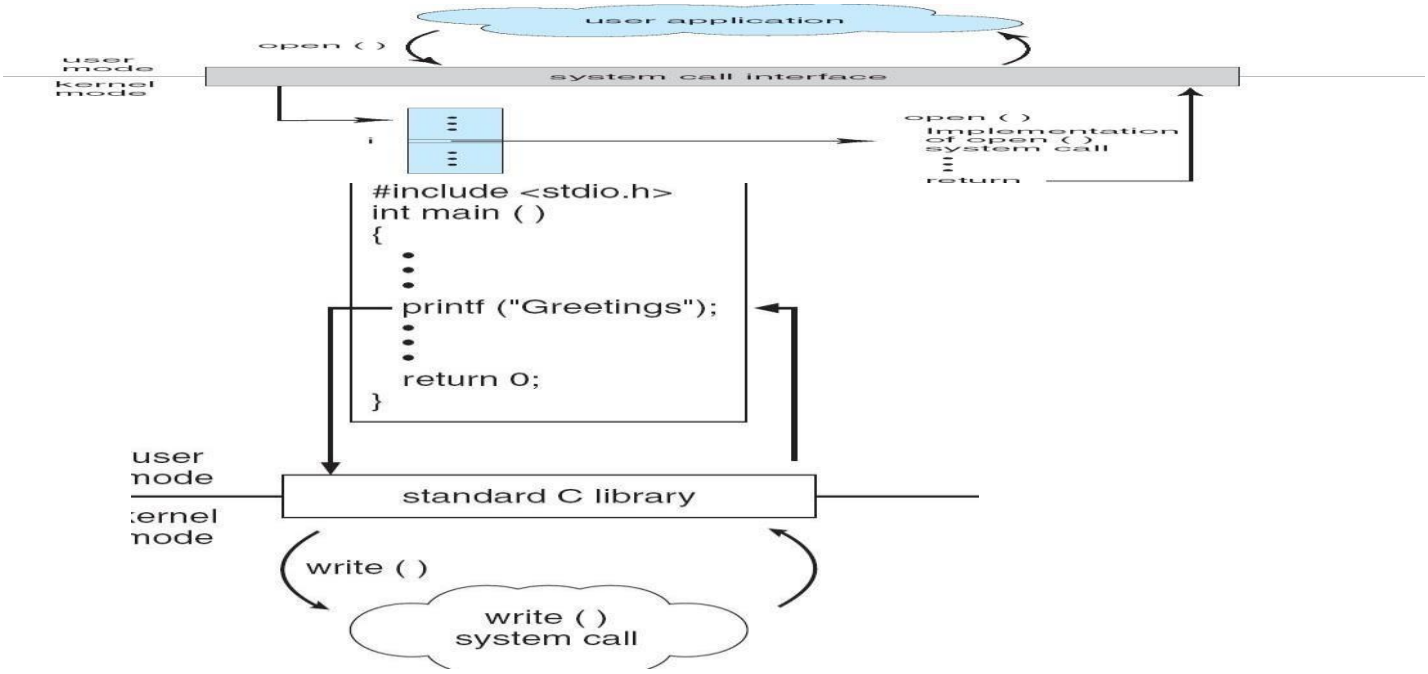
The caller need know nothing about how the system call is implemented Just needs to obey API and understand what OS will

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

Standard C Library Example



System Call Parameter Passing

- Often, more information is required than simply identity of desired system
- call Exact type and amount of information vary according to OS and call
- 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

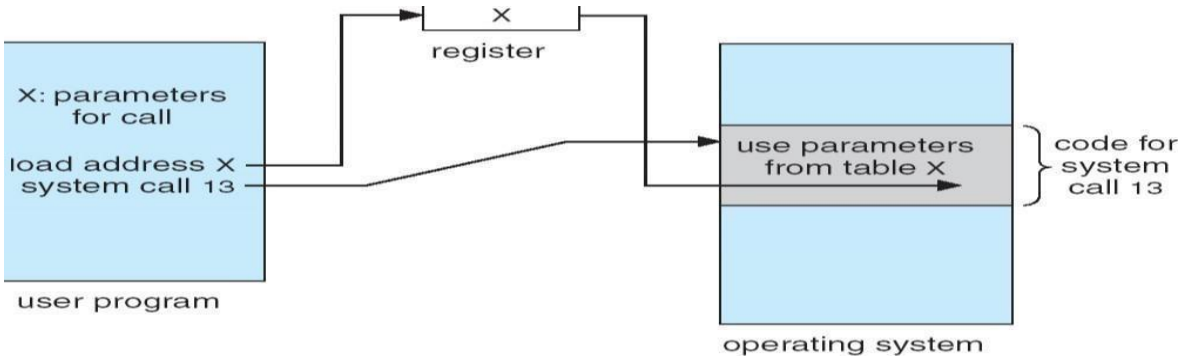
- Parameters stored in a *block,* or table, in memory, and address of block passed as a parameter

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

Parameter Passing via Table



Types of System Calls

1. Process control
2. File management
3. Device management
4. Information maintenance
5. Communications

Process control

A running needs to halt its execution either normally or abnormally.
If a system call is made to terminate the running program, a dump of memory is sometimes taken and an error message generated which can be diagnosed by a debugger

- end, abort

o load, execute
o create process, terminate process
o get process attributes, set process attributes
o wait for time
o wait event, signal event
o allocate and free memory

File management

OS provides an API to make these system calls for managing files
o create file, delete file
o open, close file
o read, write, reposition
o get and set file attributes

Device management

Process requires several resources to execute, if these resources are available, they will be granted and control retuned to user process. Some are physical such as video card and other such as file. User program request the device and release when finished

- o request device, release device
- o read, write, reposition
- o get device attributes, set device attributes
- o logically attach or detach devices

Information maintenance

System calls exist purely for transferring information between the user program and OS. It can return information about the system, such as the number of current users,the version number of the operating system, the amount of free memory or disk space and so on.

- get time or date, set time or date
- get system data, set system data
- get and set process, file, or device attributes

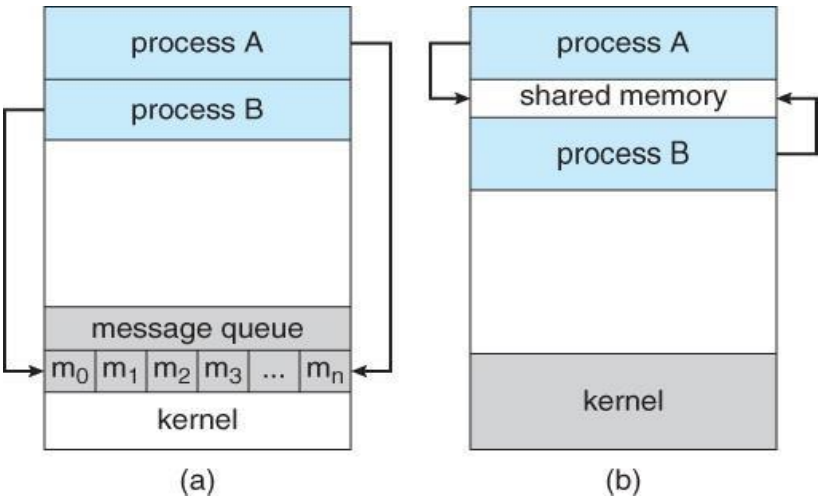
Communications

Two common models of communication

Message-passing model, information is exchanged through an inter process-communication facility provided by the OS.

Shared-memory model, processes use map memory system calls to gain access to regions of memory owned by other processes.

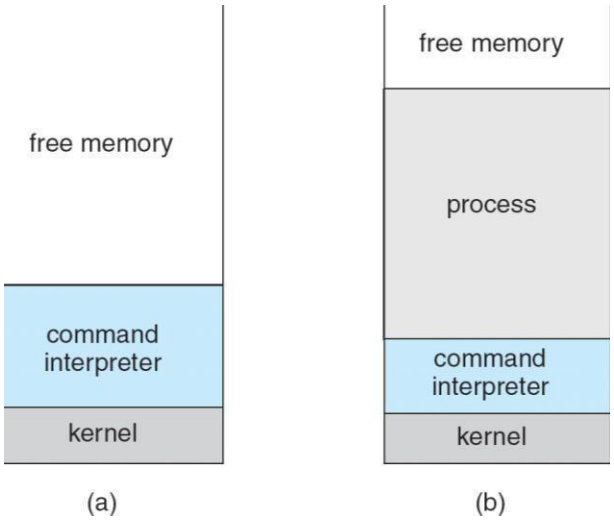
- o create, delete communication connection
- o send, receive messages
- o transfer status information
- o attach and detach remote devices



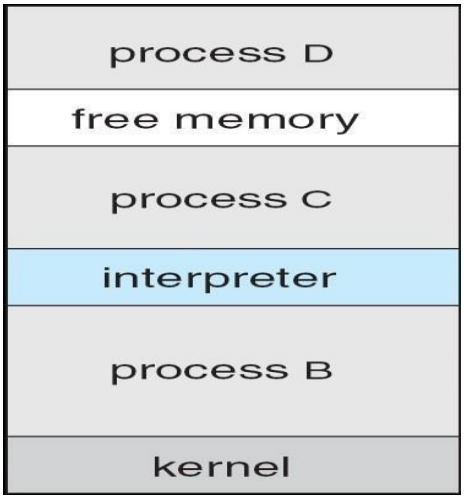
Examples of Windows and Unix System Calls

	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() InitializeSecurityDescriptor() SetSecurityDescriptorGroup()	chmod() umask() chown()

MS-DOS execution



(a) At system startup (b) running a program FreeBSD Running Multiple Programs



System Programs

System programs provide a convenient environment for program development and execution. The 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, not the actual system calls 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

- 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- Absolute loaders, relocatable loaders, linkage editors, and overlay-loaders, debugging systems for higher-level and machine language

- Communications - Provide the mechanism for creating virtual connections among processes, users, andcomputer systems
- Allow users to send messages to one another's screens, browse web pages, send electronic-mailmessages, log in remotely, transfer files from one machine to another