UNIVERSITY of SAN CARLOS
SCIENTIA • VIRTUS • DEVOTIO

DCISM
DEPARTMENT OF COMPUTER, INFORMATION SCIENCES AND MATHEMATICS

1

CS 3104
OPERATING SYSTEMS

CHAPTER 2

OPERATING-SYSTEM STRUCTURES

OUTLINE

- Operating System Services
- User and Operating-System Interface
- System Calls
- System Services
- Linkers and Loaders

- Outline
- ✓Operating System Services
- User and Operating-System Interface
- System Calls
- System Services
- Linkers and Loaders

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)**
    - **Examples:**
      Command-Line Interface (CLI), Graphical User Interface (GUI), Touchscreen Interface

  - **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**

- Outline
- ✓Operating System Services
- User and Operating-System Interface
- System Calls
- System Services
- Linkers and Loaders

OPERATING SYSTEM SERVICES

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

- Outline
- ✓Operating System Services
- User and Operating-System Interface
- System Calls
- System Services
- Linkers and Loaders

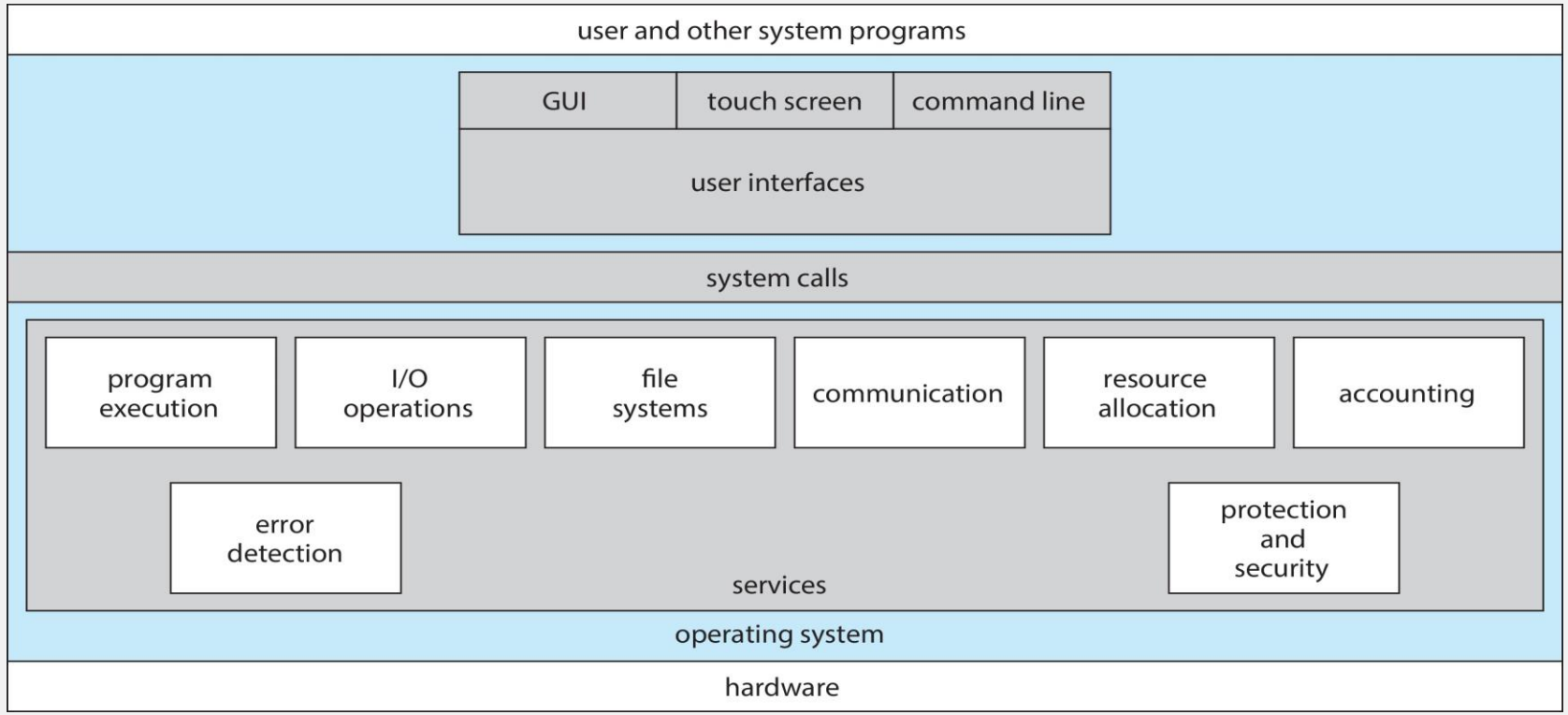
OPERATING SYSTEM SERVICES

- **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.
  - **Logging: 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

- Outline
- ✓Operating System Services
- User and Operating-System Interface
- System Calls
- System Services
- Linkers and Loaders

OPERATING SYSTEM SERVICES

A VIEW OF OPERATING-SYSTEM SERVICES



- Outline
- Operating System Services
- ✓User and Operating-System Interface
- System Calls
- System Services
- Linkers and Loaders

USER AND OPERATING-SYSTEM INTERFACE

Command-Line Interface (CLI)

- It is also known as **Command interpreter**.
- It 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
  - Sometimes commands built-in, sometimes just names of programs
    - If the latter, adding new features doesn't require shell modification

- Outline
- Operating System Services
- ✓User and Operating-System Interface
- System Calls
- System Services
- Linkers and Loaders

USER AND OPERATING-SYSTEM INTERFACE

Bourne-Again (or Bash) Shell Command Interpreter in macOS

```
1. root@r6181-d5-us01:~ (ssh)
root@r6181-d5-u... ⌘1   ssh ⌘2   root@r6181-d5-us01... ⌘3
Last login: Thu Jul 14 08:47:01 on ttys002
iMacPro:~ pbg$ ssh root@r6181-d5-us01
root@r6181-d5-us01's password:
Last login: Thu Jul 14 06:01:11 2016 from 172.16.16.162
[root@r6181-d5-us01 ~]# uptime
 06:57:48 up 16 days, 10:52,  3 users,  load average: 129.52, 80.33, 56.55
[root@r6181-d5-us01 ~]# df -kh
Filesystem            Size  Used Avail Use% Mounted on
/dev/mapper/vg_ks-lv_root
                       50G   19G   28G  41% /
tmpfs                 127G  520K  127G   1% /dev/shm
/dev/sda1             477M   71M  381M  16% /boot
/dev/dssd0000         1.0T  480G  545G  47% /dssd_xfs
tcp://192.168.150.1:3334/orangefs
                       12T  5.7T  6.4T  47% /mnt/orangefs
/dev/gpfs-test         23T  1.1T   22T   5% /mnt/gpfs
[root@r6181-d5-us01 ~]#
[root@r6181-d5-us01 ~]# ps aux | sort -nrk 3,3 | head -n 5
root      97653 11.2  6.6 42665344 17520636 ?   S<Ll Jul13 166:23 /usr/lpp/mmfs/bin/mmfsd
root      69849  6.6  0.0        0        0 ?   S    Jul12 181:54 [vpthread-1-1]
root      69850  6.4  0.0        0        0 ?   S    Jul12 177:42 [vpthread-1-2]
root       3829  3.0  0.0        0        0 ?   S    Jun27 730:04 [rp_thread 7:0]
root       3826  3.0  0.0        0        0 ?   S    Jun27 728:08 [rp_thread 6:0]
[root@r6181-d5-us01 ~]# ls -l /usr/lpp/mmfs/bin/mmfsd
-r-x------ 1 root root 20667161 Jun  3  2015 /usr/lpp/mmfs/bin/mmfsd
[root@r6181-d5-us01 ~]#
```

- Outline
- Operating System Services
- ✓User and Operating-System Interface
- System Calls
- System Services
- Linkers and Loaders

USER AND OPERATING-SYSTEM INTERFACE

Graphical User Interface (GUI)

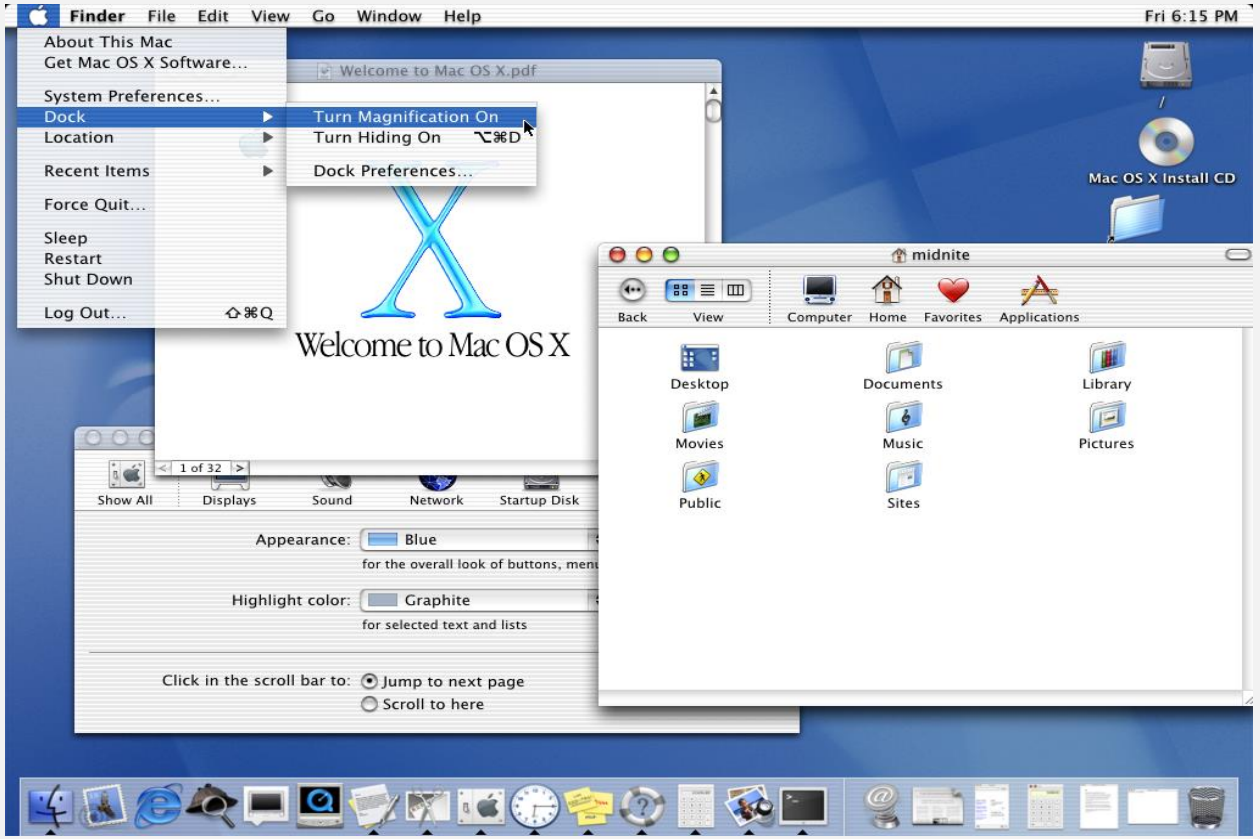
- User-friendly **desktop** metaphor interface
  - Usually mouse, keyboard, and monitor
  - **Icons** represent files, programs, actions, etc.
  - Various mouse buttons over objects in the interface cause various actions
    - provide information, options, execute function, open directory (known as a **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** is **"Aqua" GUI** interface with **UNIX kernel underneath and shells available**
  - **Unix and Linux** have **CLI with optional GUI interfaces**
    - CDE, KDE, GNOME

- Outline
- Operating System Services
- ✓User and Operating-System Interface
- System Calls
- System Services
- Linkers and Loaders

USER AND OPERATING-SYSTEM INTERFACE

The macOS X GUI



USER AND OPERATING-SYSTEM INTERFACE

- Outline
- Operating System Services
- ✓User and Operating-System Interface
- System Calls
- System Services
- Linkers and Loaders

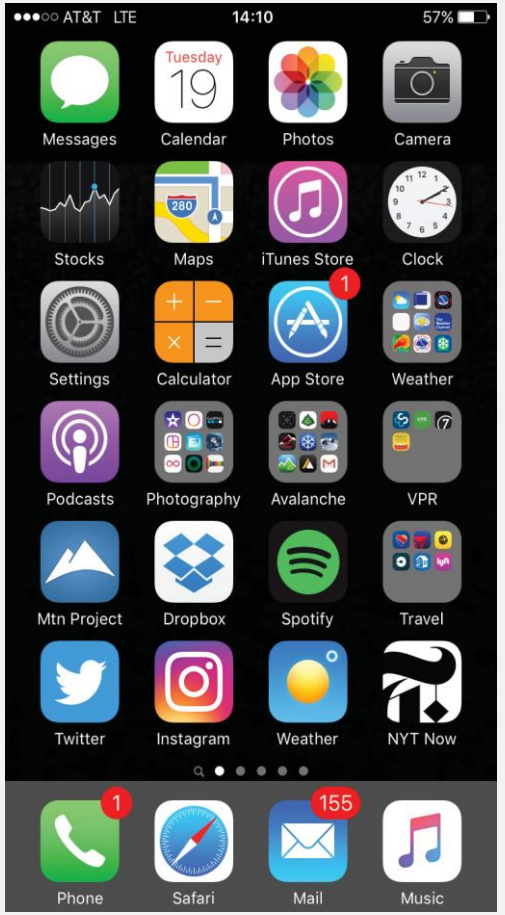
The macOS X GUI

- Outline
- Operating System Services
- ✓User and Operating-System Interface
- System Calls
- System Services
- Linkers and Loaders

USER AND OPERATING-SYSTEM INTERFACE

Touch-Screen Interface

- Touchscreen devices require new interfaces
  - Mouse not possible or not desired
  - Actions and selection based on **gestures**
  - Virtual keyboard for text entry
- Voice commands



- Outline
- Operating System Services
- User and Operating-System Interface
- ✓System Calls
- System Services
- Linkers and Loaders

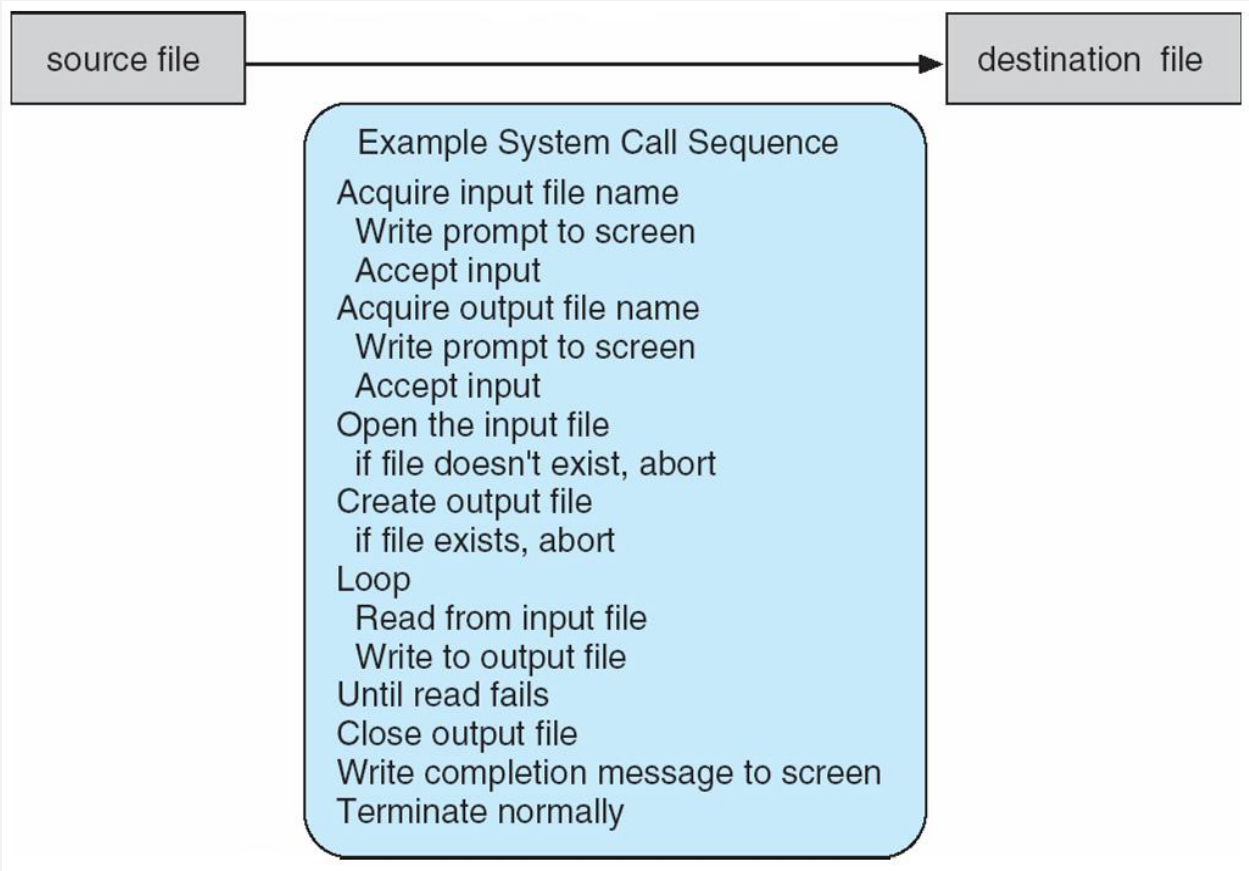
SYSTEM CALLS

- **Programming interface to the services** provided by the OS
- Typically written in **a high-level language** (C or C++)
- For certain **low-level tasks** (for example, tasks where hardware must be accessed directly) may have to be written using **assembly-language** instructions
- **Mostly accessed by programs** via a high-level **Application Programming Interface (API)** rather than direct system call use
- **Three most common APIs** are **Win32 API for Windows**, **POSIX API for POSIX-based systems** (including virtually all versions of UNIX, Linux, and macOS), and **Java API** for the Java Virtual Machine (JVM).
- **Important Note:**
  The system-call names used throughout this text are generic

- Outline
- Operating System Services
- User and Operating-System Interface
- ✓System Calls
- System Services
- Linkers and Loaders

SYSTEM CALLS: EXAMPLE

- System call sequence to copy the contents of one file to another file



- Outline
- Operating System Services
- User and Operating-System Interface
- ✓System Calls
- System Services
- Linkers and Loaders

EXAMPLE OF STANDARD API

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 into which the data will be read
- `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.

- Outline
- Operating System Services
- User and Operating-System Interface
- ✓System Calls
- System Services
- Linkers and Loaders

SYSTEM CALL IMPLEMENTATION

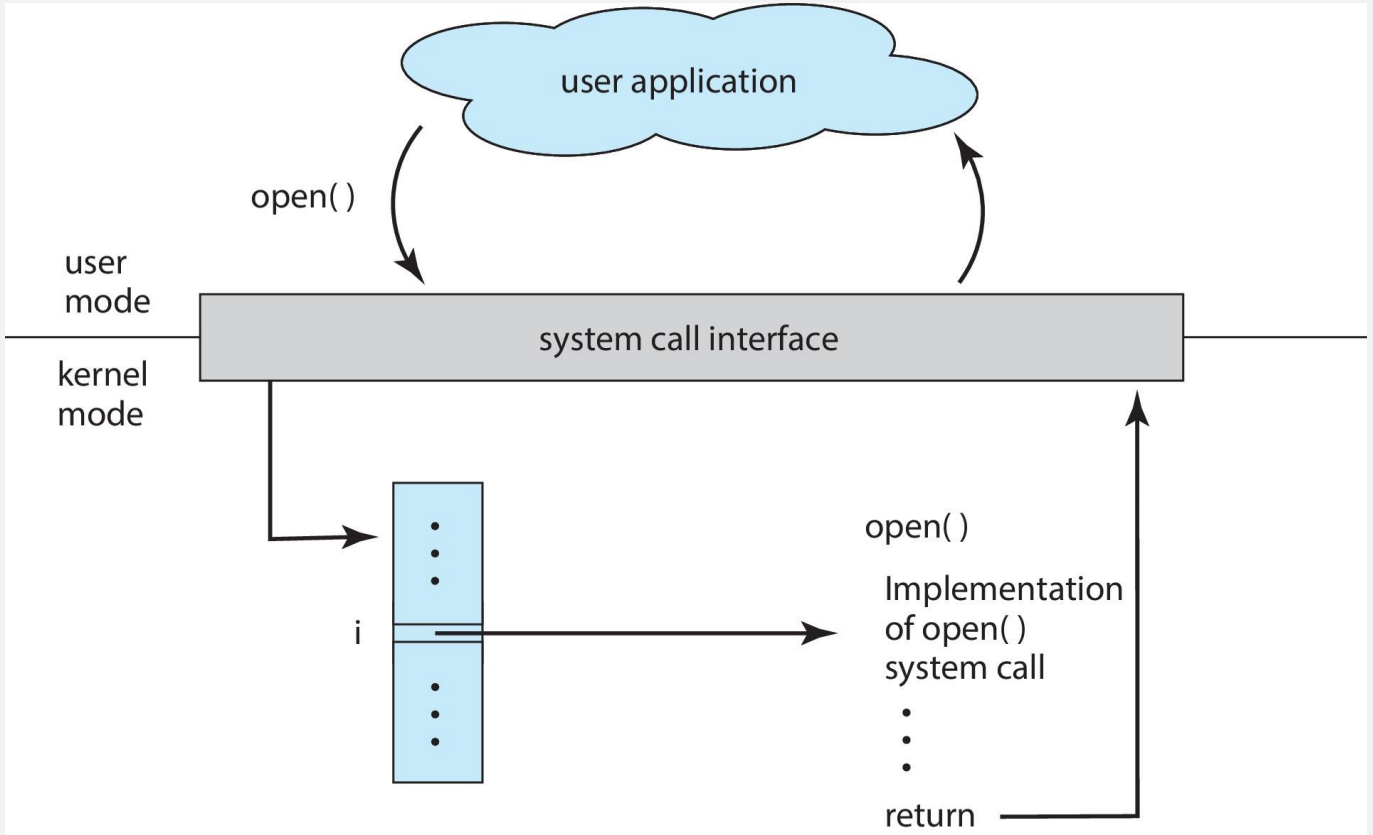
- Typically, a **number** is **associated with each system call**.
  - **System-call interface** maintains a table indexed according to these numbers.
- The **system call interface** invokes the intended system call in OS kernel and returns the status of the system call and any return values.
- The **caller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

- Outline
- Operating System Services
- User and Operating-System Interface
- ✓System Calls
- System Services
- Linkers and Loaders

SYSTEM CALLS

API-System Call Interface-OS Relationship

The handling of a user application invoking the open() system call



- Outline
- Operating System Services
- User and Operating-System Interface
- ✓System Calls
- System Services
- Linkers and Loaders

SYSTEM CALLS: PARAMETER PASSING

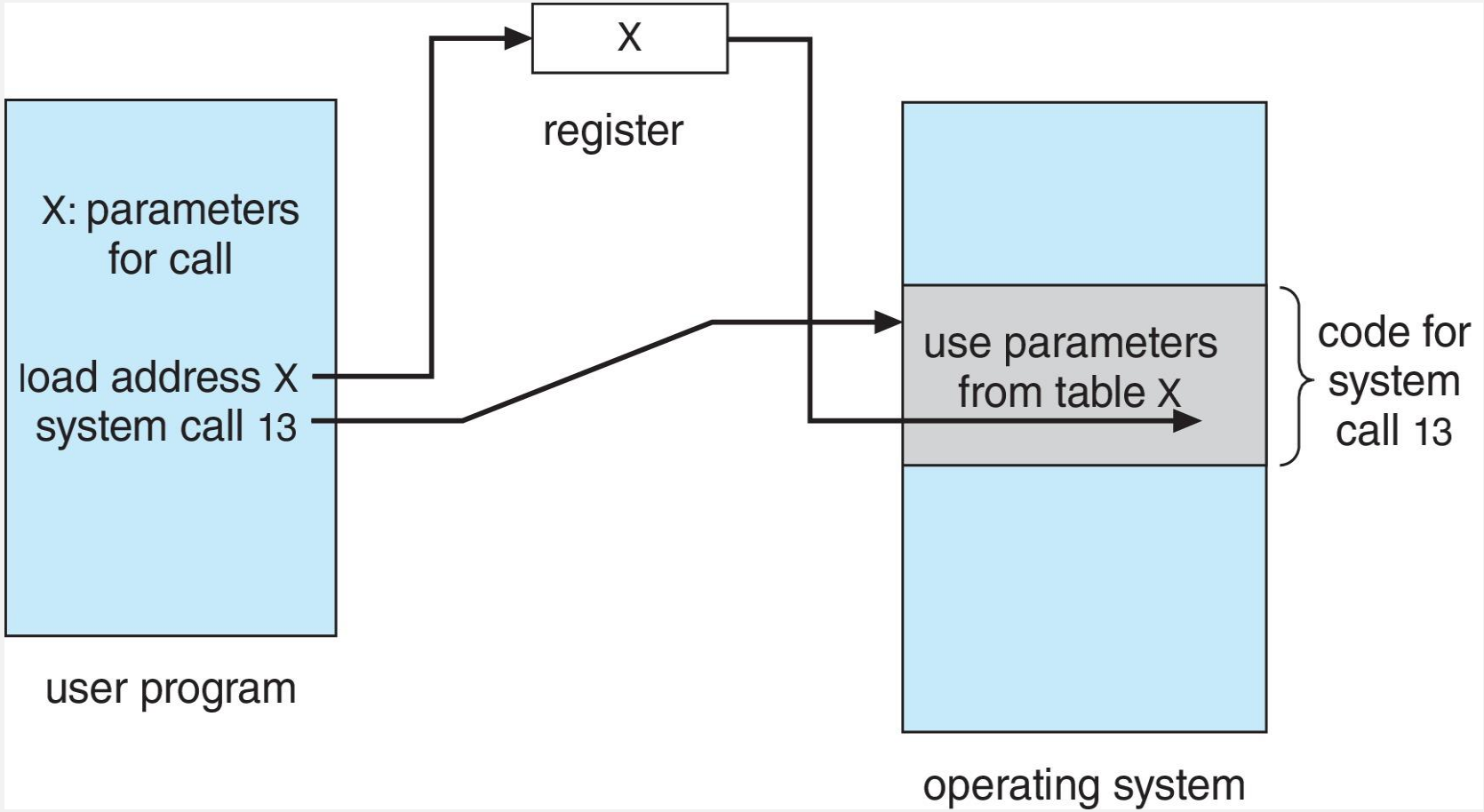
- More information is required than simply the identity of desired system call
  - Exact type and amount of information vary according to OS and call

- **Three general methods used to pass parameters to the OS:**
  - **Simplest: pass the parameters in registers**
    - In some cases, there 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

- Outline
- Operating System Services
- User and Operating-System Interface
- ✓System Calls
- System Services
- Linkers and Loaders

SYSTEM CALLS: PARAMETER PASSING

Passing of Parameters as a Table



- Outline
- Operating System Services
- User and Operating-System Interface
- ✓System Calls
- System Services
- Linkers and Loaders

TYPES OF SYSTEM CALLS

- **Process control**
  - create process, terminate process
  - end, abort
  - load, execute
  - get process attributes, set process attributes
  - wait for time
  - wait event, signal event
  - allocate and free memory
  - Dump memory if error
  - **Debugger** for determining **bugs, single step** execution
  - **Locks** for managing access to shared data between processes

- Outline
- Operating System Services
- User and Operating-System Interface
- ✓System Calls
- System Services
- Linkers and Loaders

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

- Outline
- Operating System Services
- User and Operating-System Interface
- ✓System Calls
- System Services
- Linkers and Loaders

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 if **message passing model** to **host name** or **process name**
    - From **client** to **server**
  - **Shared-memory model** create and gain access to memory regions
  - transfer status information
  - attach and detach remote devices

- Outline
- Operating System Services
- User and Operating-System Interface
- ✓System Calls
- System Services
- Linkers and Loaders

TYPES OF SYSTEM CALLS

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

- Outline
- Operating System Services
- User and Operating-System Interface
- ✓System Calls
- System Services
- Linkers and Loaders

SYSTEM CALLS

EXAMPLES OF WINDOWS AND UNIX SYSTEM CALLS

The following illustrates various equivalent system calls for Windows and UNIX operating systems.

	Windows	Unix
Process control	CreateProcess() ExitProcess() WaitForSingleObject()	fork() exit() wait()
File management	CreateFile() ReadFile() WriteFile() CloseHandle()	open() read() write() close()
Device management	SetConsoleMode() ReadConsole() WriteConsole()	ioctl() read() write()
Information maintenance	GetCurrentProcessID() SetTimer() Sleep()	getpid() alarm() sleep()
Communications	CreatePipe() CreateFileMapping() MapViewOfFile()	pipe() shm_open() mmap()
Protection	SetFileSecurity() InitializeSecurityDescriptor() SetSecurityDescriptorGroup()	chmod() umask() chown()

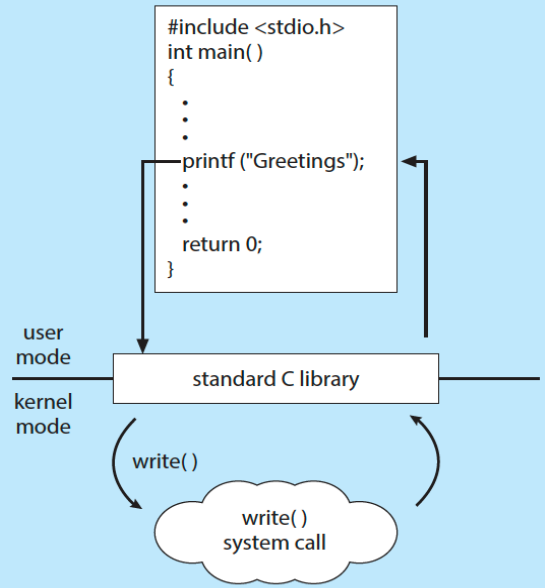
- Outline
- Operating System Services
- User and Operating-System Interface
- ✓System Calls
- System Services
- Linkers and Loaders

SYSTEM CALLS

C program invoking **printf()** library call, which calls **write()** system call

THE STANDARD C LIBRARY

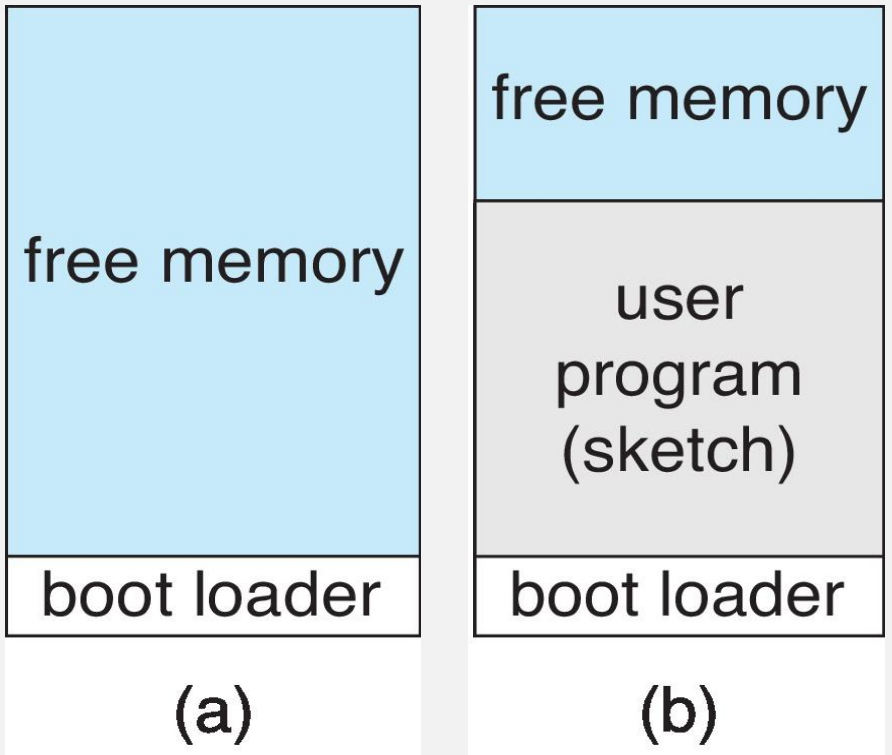
The standard C library provides a portion of the system-call interface for many versions of UNIX and Linux. As an example, let's assume a C program invokes the `printf()` statement. The C library intercepts this call and invokes the necessary system call (or calls) in the operating system—in this instance, the `write()` system call. The C library takes the value returned by `write()` and passes it back to the user program:



- Outline
- Operating System Services
- User and Operating-System Interface
- ✓System Calls
- System Services
- Linkers and Loaders

SYSTEM CALLS

- **ARDUINO**
  - Single-tasking
  - No operating system
  - Program (sketch) loaded via USB into flash memory
  - Single memory space
  - Boot loader loads program
  - Program exit → shell reloaded

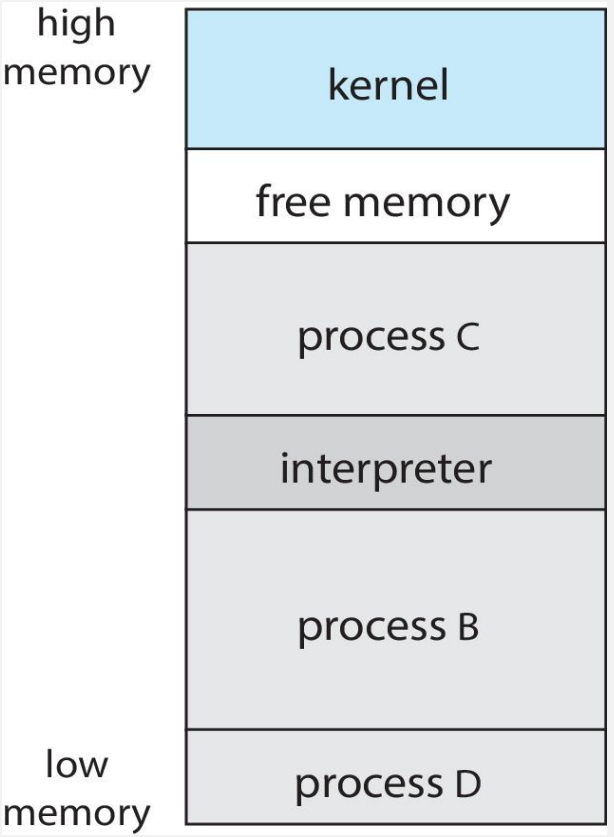


At system startup **Running a sketch**

- Outline
- Operating System Services
- User and Operating-System Interface
- ✓System Calls
- System Services
- Linkers and Loaders

SYSTEM CALLS

- **FREEBSD**
  - Unix variant
  - Multitasking
  - User login → invokes user's choice of shell
  - Shell executes **fork()** system call to create process
    - Executes **exec()** to load program into process
    - Shell waits for process to terminate or continues with user commands
  - Process exits with:
    - code = 0 → no error
    - code > 0 → error code



- Outline
- Operating System Services
- User and Operating-System Interface
- System Calls
- ✓System Services
- Linkers and Loaders

SYSTEM SERVICES

- **System services (programs)**, which are also known as **System Utilities**, provide a convenient environment for program development and execution.

- Categories of system services:
  - File management
  - Status information
  - File modification
  - Programming-language support
  - Program loading and execution
  - Communications
  - Background services
  - Application programs

- **Important Note:**
  Most users' view of the operating system is defined by system programs, not by the actual system calls

- Outline
- Operating System Services
- User and Operating-System Interface
- System Calls
- ✓System Services
- Linkers and Loaders

SYSTEM SERVICES

CATEGORIES OF SYSTEM SERVICES

- **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

- Outline
- Operating System Services
- User and Operating-System Interface
- System Calls
- ✓System Services
- Linkers and Loaders

SYSTEM SERVICES

CATEGORIES OF SYSTEM SERVICES

- **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, and computer systems
  - Allow users to send messages to one another's screens, browse web pages, send electronic-mail messages, log in remotely, transfer files from one machine to another

- Outline
- Operating System Services
- User and Operating-System Interface
- System Calls
- ✓System Services
- Linkers and Loaders

SYSTEM SERVICES

CATEGORIES OF SYSTEM SERVICES

- **Background Services:**
  - Launch at boot time
    - ✓Some for system startup, then terminate
    - ✓Some from system boot to shutdown
  - Provide facilities like disk checking, process scheduling, error logging, printing
  - Run in user context not kernel context
  - Known as **services**, **subsystems**, **daemons**

- **Application programs:**
  - Don't pertain to system
  - Run by users
  - Not typically considered part of OS
  - Launched by command line, mouse click, finger poke

- Outline
- Operating System Services
- User and Operating-System Interface
- System Calls
- System Services
- ✓Linkers and Loaders

LINKERS AND LOADERS

- **Source files (source codes)** are compiled into **object files** that are designed to be loaded into any **physical** memory location (**relocatable object file)**.

- **Linker** combines these **relocatable object files** into a **single binary executable file**.
  - **During the linking phase**, other **object files or libraries may be included** as well, such as the **standard C or math library**

- **Program** resides on **secondary storage** as binary executable, must be brought into memory by **loader** to be executed

- **Loader** loads the **binary executable file into memory**, where it is eligible to run on a CPU core.

- Outline
- Operating System Services
- User and Operating-System Interface
- System Calls
- System Services
- ✓Linkers and Loaders

LINKERS AND LOADERS

- **Relocation**
  - activity associated with **linking and loading**
  - **assigns final addresses** to program parts and adjusts code and data in program to match those addresses

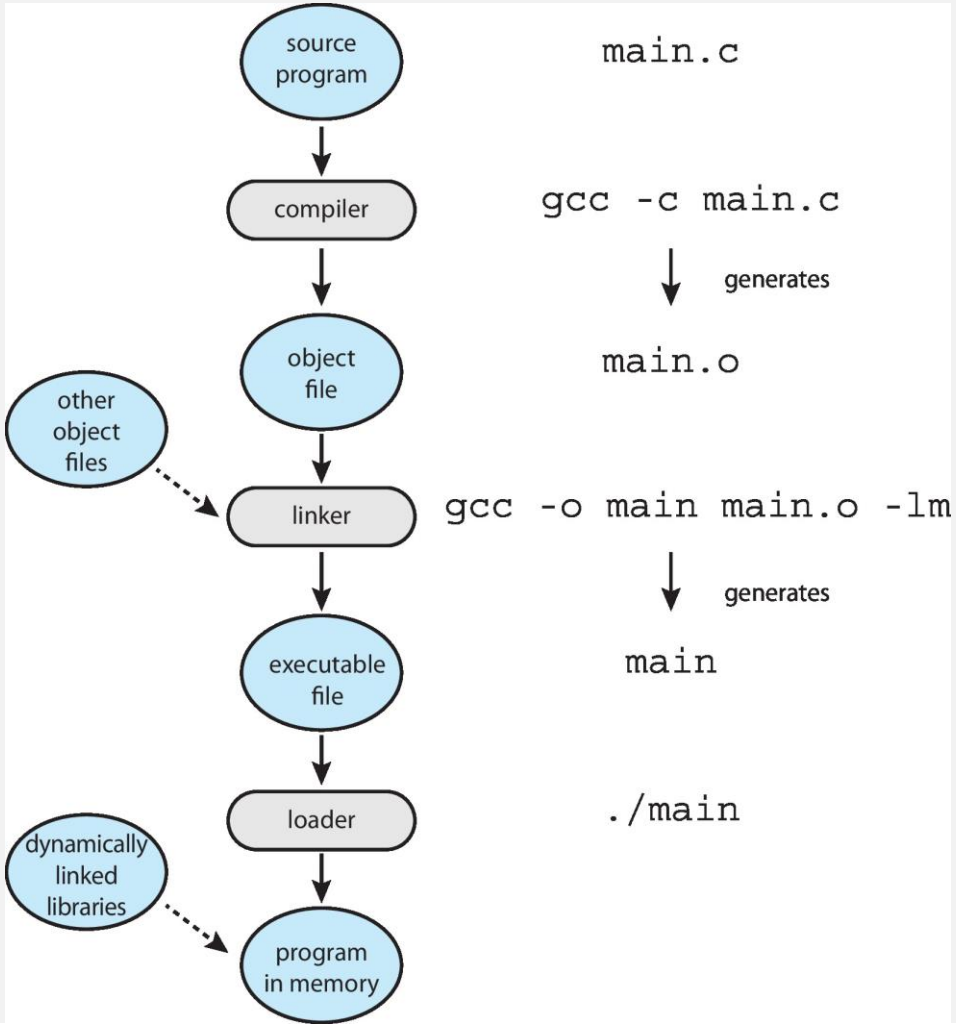
- Modern general purpose systems don't link libraries into executables
  - Rather, **dynamically linked libraries** (in Windows, **DLLs**) are loaded as needed, shared by all that use the same version of that same library (loaded once)

- **Object, executable files have standard formats**, so operating system knows how to load and start them

- Outline
- Operating System Services
- User and Operating-System Interface
- System Calls
- System Services
- ✓Linkers and Loaders

LINKERS AND LOADERS

THE ROLE OF THE LINKER AND LOADER



UNIVERSITY of SAN CARLOS
SCIENTIA • VIRTUS • DEVOTIO

DCISM
DEPARTMENT OF COMPUTER, INFORMATION SCIENCES AND MATHEMATICS

1

CS 3104
OPERATING SYSTEMS

*** END ***

THANK YOU!