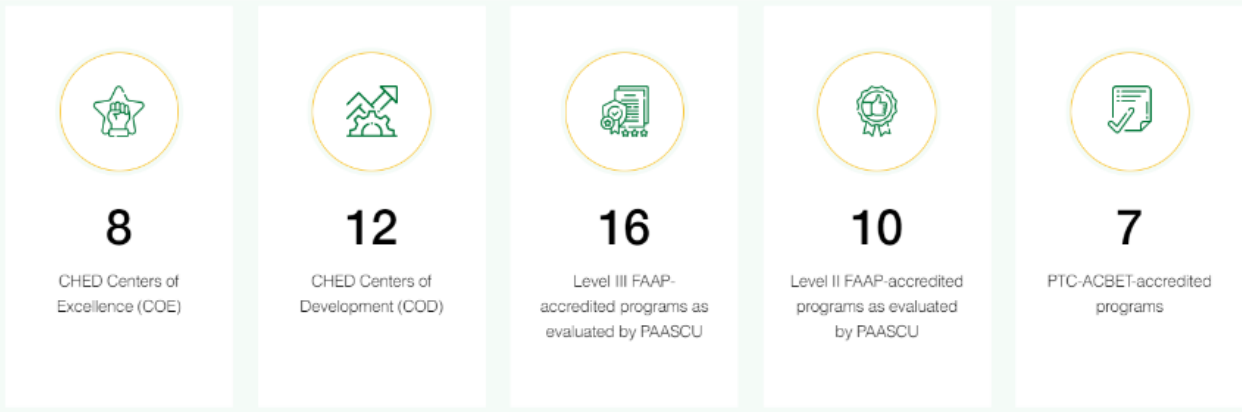
University of San Carlos (USC) is a Catholic educational institution administered since 1935 by Society of the Divine Word (SVD) missionaries.

A University since 1948, USC offers the complete educational package from kindergarten, including a Montessori academy, to graduate school.

Learn more about Education with a Mission and how we Witness to the Word.



Administered by the Society of the Divine Word (Societas Verbi Divini, SVD) since 1935, the University of San Carlos (USC) traces its roots to the Colegio de San Ildefonso founded in August 1595 in Cebu City.

The school closed in 1769 following the expulsion of Jesuit priests from the Philippines, and was reopened in 1783 as Colegio-Seminario de San Carlos which operated until the Colegio split from the seminary in 1930.

Following another brief closure during World War II, Colegio de San Carlos became a University in 1948.

Rapid growth in the ‘50s saturated the campus near the city center prompting expansion of the University to what was then called the Boys’ High School in 1956 (now North Campus), and in 1964 to the Teacher Education Center and Girls’ High School (now South Campus) and to Talamban Campus.

In 2008, the erstwhile SVD Formation Center was transformed into the Montessori Campus.

Total land area of the University’s five campuses is almost 88 hectares (or 217 acres), with about 78 hectares in Talamban Campus alone and potentially ample room for future growth.

Today, the University is one of the most respected higher education institutions in the Philippines, offering 45 undergraduate and 62 graduate programs.

Many of these programs have received Level II or Level III accreditation from the Federation of Accrediting Associations of the Philippines (FAAP), as evaluated by the Philippine Accrediting Association of Schools, Colleges, and Universities (PAASCU).

Seven engineering programs are also accredited by the Philippine Technological Council-Accreditation and Certification Board for Engineering and Technology (PTC-ACBET).

The Philippine Commission on Higher Education (CHED) also granted Autonomous Status to the University, and designated eight Centers of Excellence (COE) and 12 Centers of Development (COD) in USC.

Nearly 22,000 students in basic to graduate education are enrolled in the University, with almost 200 international students.

On average, the teacher-to-student ratio at USC is 1:20. USC is ranked between 451 to 500 in the QS University Rankings for Asia 2021.

Recognized as a research hub in southern Philippines, USC has drawn in external grants amounting to PHP298M (USD5.68M) between AY 2018-2021.

Internal research grants of over PHP46M (over USD962T) have also been awarded from the University Research Trust Fund within the same time period.

Research efforts are supported by a print collection of over 200,000 titles and almost 10,000 non-print volumes housed in the University’s Library System, along with subscriptions to 17 online journals.

USC also publishes two respected scholarly journals, The Philippine Scientist and the Philippine Quarterly of Culture and Society.

Additional support for researchers are available through offices or committees providing ethics review, intellectual property and innovation and technology support, and animal care and use.

Nineteen patents have been filed by the University since 2012, and one start-up company, Green Enviro Management Systems (GEMS), Inc., has been established.

Student support in the University includes an international students' office, online enrolment, dormitories, numerous cafes and canteens, and a transport system within Talamban Campus.

USC has more than 600 undergraduate and graduate scholars at any given time.

Since 2014, USC has been designated as a Donee Institution by the Philippine Council of Non-Governmental Organization Certification (PCNC).

Vision

The University of San Carlos sees:

A WORLD where the darkness of sin and the night of unbelief vanish before the light of the Word and the Spirit of grace

A SOCIETY where citizens are competent, noble in character and community oriented:

What they know, they apply justly and honestly.

What they do not know, they seek to learn.

What they do not have, they endeavor to acquire.

What they have, they share.

Mission

The University of San Carlos is a Catholic Institution of learning that embodies the principles of academic discipline of San Carlos Borromeo and the missionary charism of the Society of the Divine Word (SVD).

We aim to develop competent and socially responsible professionals and lifelong learners in an environment that fosters excellence in the academic core processes of teaching-learning, research, and community extension service.

Our mission is to provide timely, relevant, and transformative academic programs responsive to the needs of the local, national and global communities in rapidly changing world.

SVD Philippines Southern Province

SVD confreres in the Philippines Southern Province (PHS) are involved in education, formation, and pastoral work. USC is the largest educational institution in the SVD Philippines Southern Province.

SVD Generalate

Societas Verbi Divini (SVD) or the Society of the Divine Word is a missionary congregation founded in 1875 by St. Arnold Janssen. Today, SVD priests and brothers serve in 55 countries, including the Philippines where it is recognized as the largest male religious congregation.

Catholic Identity

The University's Catholic identity is embodied in two statements, Education with a Mission and Witness to the Word.

USC aims at the holistic formation of the individual that enlightens the mind, hones skills, and positively transforms character and behavior (from Dialogue with the Word, SVD Education Ministry as Mission of Dialogue, 2010, 103–104).

Every Carolinian becomes a catalyst towards the realization of the kingdom of God, promoting truth, love, justice, freedom, peace, and grace.

Carolinians in society have a sense of mission recognizable by their contribution as the "light," "salt," and "yeast" of the Gospel.

Graduates of USC are professionally competent and skilled (Scientia), have noble character and are value-driven (Virtus), and are dedicated to social transformation (Devotio).

Core Values

Integrity

Carolinians shall reflect in their personal and professional life the ideals of the Catholic university as "an academic community, which, in a rigorous and critical fashion, assists in the protection and advancement of human dignity and of a cultural heritage through research, teaching and extension services to the local, national and international communities" (Apostolic Exhortation Ex Conde Ecclesiae, 1990).

Excellence

Carolinians shall constantly strive to attain the highest standards in their respective fields.

Commitment

Carolinians shall bear in mind that the ultimate goal of the University is the transformation of communities, especially in the Visayas and Mindanao, through the provision of quality basic and higher education.

Social Responsibility

Carolinians shall strive to provide themselves and the students an understanding of and effective tools for addressing the prevailing social realities in the country. As far as possible, they shall volunteer their expertise and contribute to effective social and civic programs in the local community, through initiatives organized by themselves, their respective departments and relevant external organizations.

Evangelization

Carolinians shall seek to understand the values and mores of local cultures and enrich them through gospel values and the teachings of the Church. In a privileged manner, they shall, in solidarity with the Philippines Southern Province of the Society of Divine Word, support the missionary apostolate in the Visayas and Mindanao.

Leadership

Carolinians, not only in positions of authority but also in their own personal capacity, shall strive to set the conditions for reflection and learning on their respective educational tasks, both in their departments and in the local community, foster norms of behavior befitting a Witness to the Word and exhibit these norms in their own way of life.

Board of Trustees

Carmelita I. Quebengco
Chair, Board of Trustees
Chancellor Emeritus, De La Salle University

Hilario G. Davide Jr.
Chief Justice (ret.), Supreme Court of the Philippines

Fr. Cyrus T. Mercado, SVD
Provincial Treasurer, SVD-PH Southern Province

Fr. Ruel F. Lero, SVD
President,
Holy Name University

Conchita L. Manabat
Independent Director,
Philippine Dealing System

Fr. Narciso A. Cellan Jr., SVD
President,
University of San Carlos

Fr. Antonio M. Pernia, SVD
Dean of Studies, Divine Word Institute of Mission Studies

Fr. Rogelio N. Bag-ao, SVD
Provincial Superior, SVD-PH Southern Province

Jose R. Soberano III
Chairman and President, Cebu Landmasters, Inc.

Fr. Nielo M. Cantilado, SVD
President,
Divine Word College of Legazpi

Fr. Melchor P. Fuerzas, SVD
Corporate Secretary,
Asst. Vice President for Administration, University of San Carlos

USC Cabinet

Fr. Narciso A. Cellan Jr., SVD
President

Fr. Jesuraj Anthoniappen, SVD
Vice President for Academic Affairs

Fr. Generoso Ricardo B. Rebayla Jr., SVD
Vice President for Administration

Fr. Arthur Z. Villanueva, SVD
Vice President for Finance

Joan S. Largo
Assistant Vice President for Academic Affairs

Fr. Melchor P. Fuerzas, SVD
Assistant Vice President for Administration

The University has been granted by the Philippine Commission on Higher Education (CHED) Autonomous Status, which recognizes USC's "track record of exceptional program performance and institutional quality."

Twenty-three academic programs from 17 academic units have Level III accreditation from the Federation of Accrediting Associations of the Philippines (FAAP), as evaluated by the Philippine Accrediting Association of Schools, Colleges, and Universities (PAASCU). These academic department and sections include:

- **Anthropology**
- **Biology**
- **Business Administration**
- **Chemistry**
- **Communication, Languages and Literature**
- **Economics**
- **History**
- **Library Science**
- **Mathematics**
- **Nursing**
- **Philosophy**
- **Physics**
- **Political Science**
- **Psychology**
- **Science and Mathematics Education**
- **Sociology**
- **Teacher Education**

Ten other programs have FAAP Level II accreditation as evaluated by PAASCU, including:

- **Accountancy**
- **Chemical Engineering**
- **Civil Engineering**
- **Computer Engineering**
- **Electrical Engineering**
- **Graduate Arts and Sciences**
- **Graduate Education**
- **Industrial Engineering**
- **Mechanical Engineering**
- **Pharmacy**

Seven engineering programs (chemical, civil, computer, electrical, electronics, industrial, and mechanical engineering) have received full accreditation by the Philippine Technological Council-Accreditation and Certification Board for Engineering and Technology (PTC-ACBET).

The CHED has also designated eight academic programs as Centers of Excellence (COE), including:

- **Anthropology**
- **Business Administration**
- **Chemical Engineering**
- **Entrepreneurship**
- **Mechanical Engineering**
- **Office Administration**
- **Physics**
- **Teacher Education**

Twelve other programs are designated as CHED Centers of Development (COD):

- **Biology**
- **Chemistry**
- **Civil Engineering**
- **Computer Engineering**
- **Electrical Engineering**
- **Electronics Engineering**
- **Hotel and Restaurant Management**
- **Industrial Engineering**
- **Information Technology**
- **Marine Biology**
- **Philosophy**
- **Tourism**

USC reaps CHED awards

The University of San Carlos was given six awards by the Commission on Higher Education (CHED) during the celebration of the First National Higher Education Day and the 27th Founding Anniversary of the Commission held from May 17 to 24, 2021.

The University was cited for Exemplary Leadership, Institutional Performance in the QS Asia University Rankings, Program Excellence for its Centers of Excellence (COEs) and Centers of Development (CODs), and for Institutional Excellence as an Autonomous Higher Educational Institution (HEI). The Commission also cited USC as a partner for quality assurance in higher education.

University President Fr. Narciso A. Cellan Jr., SVD, D.Comm. received the awards during the blended (face-to-face and virtual) awarding ceremony held on May 22, 2021. Actual attendance for local HEI heads at the Activity Center of the Cebu Normal University was limited to 50 persons. Heads of HEIs from elsewhere in the region attended via video conferencing.

STUDENT SUPPORT SERVICES

In support of academic instruction, the University of San Carlos' Student Support and Services, as an integrated system, is committed to the delivery of quality services and programs essential in the formation of Carolinian graduates who are envisioned to become productive citizens of the country and the world.

We are committed to continuously improve our services, processes and our quality management system for the enhancement of student experience in the university; this may require the participation of our suppliers and other external providers of processes and services.

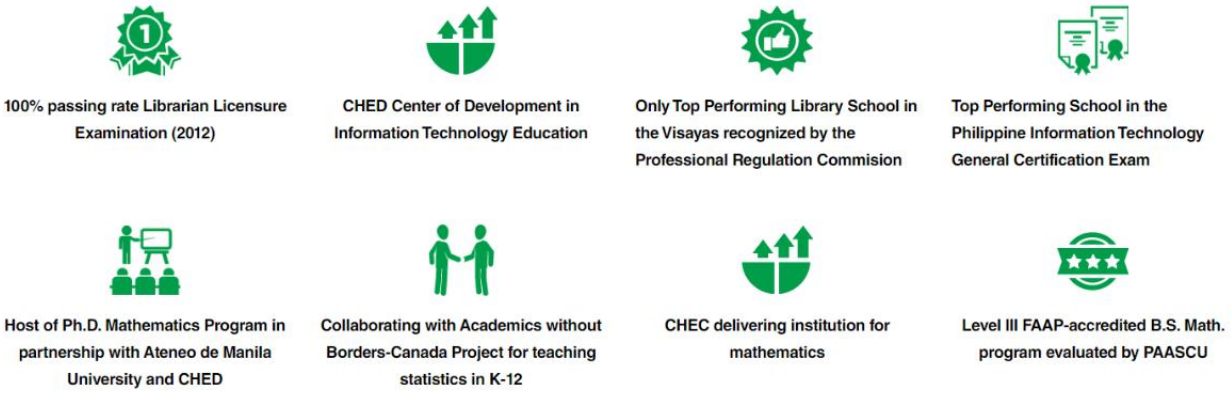
Feel free to access our dedicated portalLinks to an external site. to learn more.

Caloy is the client-facing and interactive version of the Quality Management System documentation of USC's student support and services.

It is an innovative visual communication tool specially designed for Carolinians to assist them when navigating for locations, processes and services around the campus.

Amidst the pandemic, the “Caloy on the webLinks to an external site.” was launched so that Carolinians can access it remotely.

DCISM



Overview

The long history of the Department of Computer, Information Sciences, and Mathematics (DCISM) began with the establishment of the Department of Mathematics in 1962 with Fr. Michael Richartz, SVD, Ph.D. as the first department chair, with an initial offering of the Bachelor of Science in Mathematics and Master of Science in Mathematics programs to address the lack of college mathematics teachers in the region.

In 1990, the Department of Mathematics was renamed the Department of Mathematics and Computer Science following the introduction of the Bachelor of Science in Computer Science curriculum prompted by strong demand from foreign direct investments and local industries. With the development of wireless technology and in response to the needs of the IT industry, the Bachelor of Science in Information Technology program was also offered in 2000 and the Master of Science in Information Technology program in 2003.

In 2004, the Higher Education Development Project of the Commission on Higher Education identified the Department of Mathematics and Computer Science as a delivering Higher Education Institution for its scholarship grant for Master of Mathematics which aims to improve the quality of mathematics instruction in higher education institutions. Due to the increasing number of students enrolled in Information and Computer Technology, the Computer Science section separated from the Department of Mathematics in 2010 with Jacqueline Fat-Yara as its first chair.

The department is a founding member of the Cebu Educational Development Foundation for Information Technology (CEDF-IT) and Oracle Academic Initiative (OAI). The department was recognized by the Commission on Higher Education (CHED) as a Center of Development for Information Technology Education in January 2016.

Meanwhile, the Department of Library Science was established in 1966 under its first chair, Nenita Po Sy. Library Science as a discipline had an even longer journey, beginning as a minor in the Bachelor of Science in Education program in 1950, then as a Bachelor of Arts program in 1958, while a Master of Science in Library Science program was offered in 1970. Prompted by the Commission on Higher Education, the bachelor's program became an A.B. in Library and Information Science in 1998, before standardization in 2005 resulted in the offering of the Bachelor of Library and Information Science, the Master of Library and Information Science, and the Master of Science in Library and Information Science programs.

The Department of Computer and Information Science (DCIS) was formed in June 2015 from the merger of two departments, namely Computer Science and Information Technology (CSIT) and Library and Information Science (LIS). In 2019, the official name is already the Department of Computer, Information Sciences and Mathematics (DCISM).

Currently, the Department of Computer, Information Sciences and Mathematics has the following academic sections, namely:

- Computer Science
- Information and Communications Technology
- Information Technology
- Information Systems
- Library and Information Science
- Mathematics

Programs Offered

- Master of Science in Information Technology (M.S. I.T.)
- Master of Science in Library and Information Science (M.S. L.I.S.)
- Master of Science in Mathematics (M.S. Math.)
- Bachelor of Library and Information Science (B. L.I.S.)
- Bachelor of Science in Computer Science (B.S. C.S.)
- Bachelor of Science in Information and Communications Technology (B.S. I.C.T.)
- Bachelor of Science in Information Technology (B.S. I.T.)
- Bachelor of Science in Information Systems (B.S. I.S.)
- Bachelor of Science in Mathematics (B.S. Math.)
- Associate in Computer Technology major in Multimedia Technology (A. C.T.-M.T.)

Career Tracks

Computer Science graduates are primarily employed as computer R&D professionals, software design engineers, and systems software developers, as well as applications software developers and computer programmers.

Information Technology graduates can find employment as web and applications developers, database administrators, network engineers, information security administrators, systems integrators, and IT auditors. They can also serve as QA specialists, systems analysts, technical support specialists, and computer programmers.

Information and Communications Technology graduates can find profitable employment as web designers, graphic artists, multimedia editors, associate programmers, technopreneurs, computer technicians, database designers, technical network support staff, junior systems administrators, technical service representatives, software testers, technical writers, applications developers, IT officers, and R&D or network engineers.

Employers of our graduates include the following companies:

- Accenture
- Advanced World Systems Inc.
- Alliance Software Inc.
- Azeus Systems Philippines
- Convergys
- Framgia Holdings PTE Ltd.
- Global Zeal
- IBM
- Kyocera
- Lexmark
- NCR
- NEC
- Nerubia Web Solutions Inc.
- NEXT IX
- NPAX Cebu Corporation
- Rococo Global Technologies Corp.
- Savvysherpa
- Sprasia Philippines

Library and Information Science graduates serve as academic librarians, archives managers, audio-visual administrators, database managers, directors of libraries, documentalists, information consultants, information specialists, learning resources coordinators, library system analysts, media specialists, multimedia librarians, school librarians, subject specialists, and teachers of Library and Information Science.

Mathematics graduates work as software design engineers, professors and lecturers, space analysts, insurance analysts, finance and risk analysts, statisticians, programmers, consultants, researchers, search engine optimization specialists, and market analysts in private industries or governments both in the country and overseas including Canada, Australia, Germany, Japan, Taiwan, and the United States.

Employers of our graduates include:

- SavvySherpa
- NEC
- Gaisano Metro
- N-Pax Cebu Corp.
- Advance World System
- Union Bank
- China Bank
- Bank of the Philippine Islands
- Banco de Oro
- Department of Foreign Affairs
- Philippine Amusement and Gaming Corp.
- The Aboitiz Group of Companies
- Social Security System
- B-Meg Philippines
- Insular Life
- Nestle
- Various colleges and universities

Research Projects

Building Higher Education's Capacity to Conduct Eye-Tracking Research using the Analysis of Novice Programmer Tracing and Debugging Skills as a Proof of Concept, a collaborative research project with Ateneo de Manila University

E-Participation 2.0: Connecting Diverse Philippine Populations for Disaster Risk Management with a Toolkit Integrating Text and Speech Analytics, a collaborative research project with National University under Philippine-California Advanced Research Institutes (PCARI) Project Commission on Higher Education

Underwater Video Inspection Systems (UVIS), in partnership with Remote Technology Solutions Pte Ltd (Singapore)

UNIVERSITY of SAN CARLOS
SCIENTIA • VIRTUS • DEVOTIO

DCISM
DEPARTMENT OF COMPUTER, INFORMATION SCIENCES AND MATHEMATICS

1

CS 3104
OPERATING SYSTEMS

CHAPTER 1

WHAT IS AN OPERATING SYSTEM?

- A software that manages a computer's hardware
- Provides an environment for application programs to run
- Acts as an intermediary between the computer user and the computer hardware
- Varies in accomplishing its tasks in a wide variety of computing environments

WHAT IS AN OPERATING SYSTEM?

- A **software** that includes the following:
  - the always running **kernel**
  - **middleware** frameworks that ease application development and provide features
  - **system programs** that aid in managing the system while it is running

- **Kernel:** the **one program running at all times on the computer**

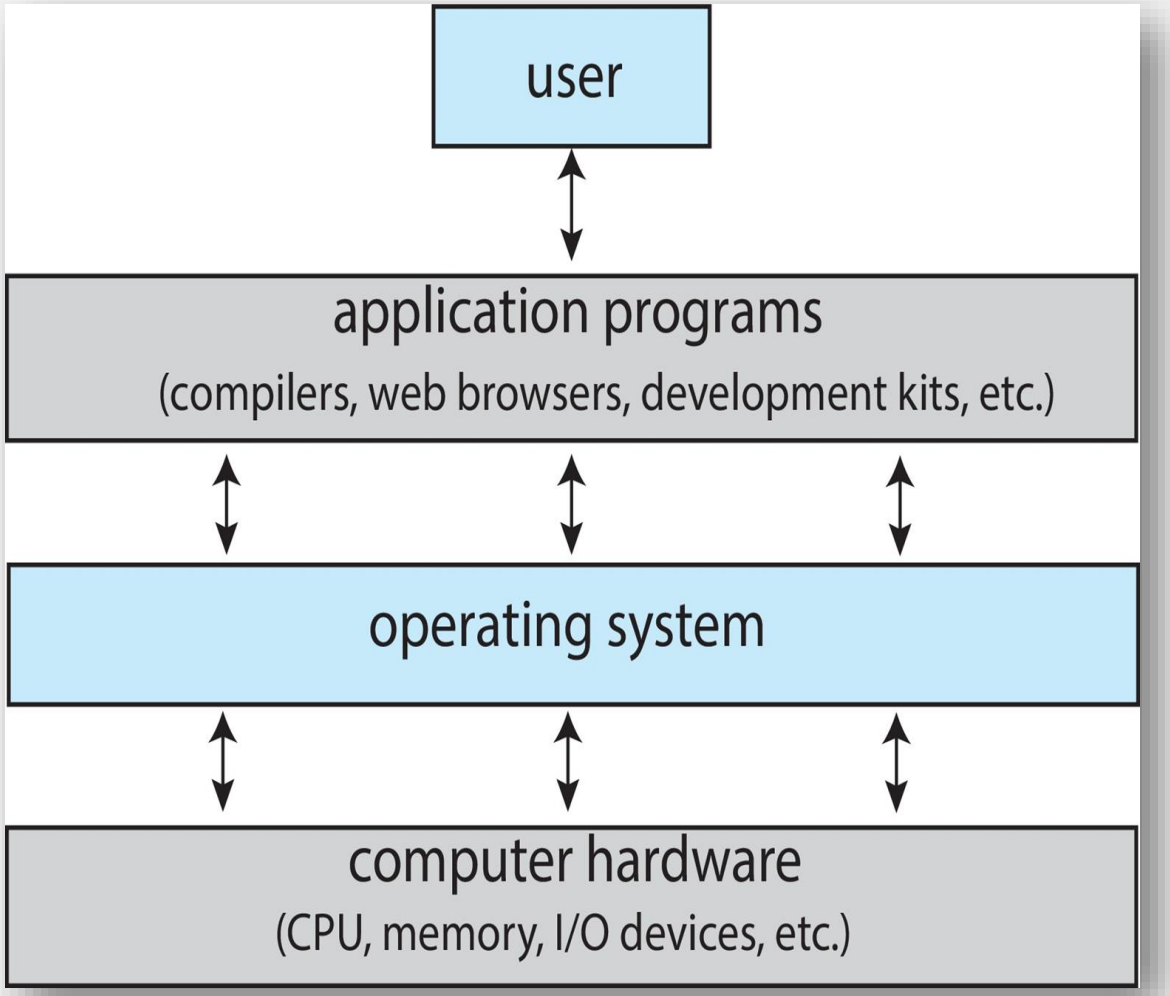
- **Middleware:**
  - a **set of software frameworks** that **provide additional services to application developers**

COMPUTER SYSTEM STRUCTURE

Computer System can be divided into four components:

- **Hardware:**
  - Provides basic computing resources: CPU, Memory, I/O devices
- **Operating System**
  - Controls and coordinates the use of hardware among various applications and users
- **Application Programs**
  - Define the ways in which the system resources are used to solve the computing problems of the users
  - Word processors, compilers, web browsers, database systems, video games
- **Users**
  - People, machines, other computers

ABSTRACT VIEW OF COMPONENTS OF COMPUTER



WHAT OPERATING SYSTEMS DO?

Users View

- Users want convenience
- Ease of use
- Good performance and security
- Don’t care about resource utilization

WHAT OPERATING SYSTEMS DO?

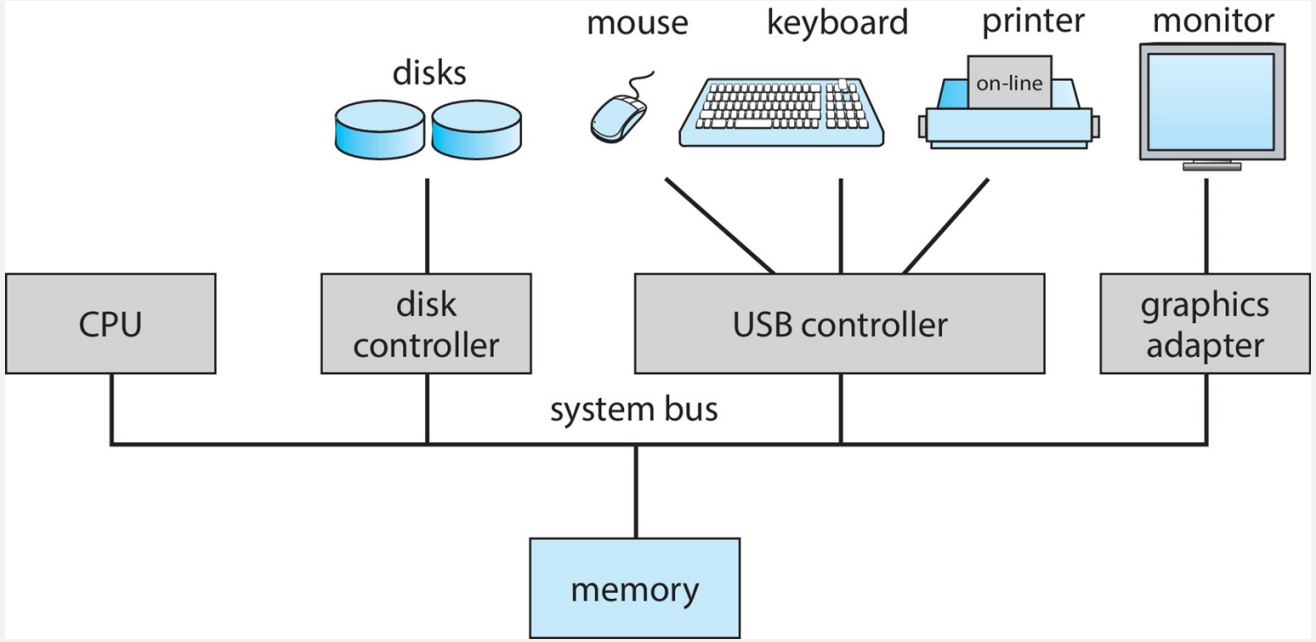
System View

- **OS as a Resource Allocator:**
  - The program most intimately involved with the hardware
  - Acts as the manager of the computer system's resources: CPU, Memory, I/O
  - Decides how to allocate the resources to specific programs and users so that it can operate the computer system efficiently and fairly.

- **OS as a Control Program:**
  - Manages the execution of user programs to prevent errors and improper use of the computer
  - Concerned with the operation and control of I/O devices

COMPUTER SYSTEM ORGANIZATION

A Typical PC Computer System



- **Computer-System Operation**
  - **One or more CPUs** and **device controllers** connect through the **common bus providing access to shared memory**
  - **Concurrent** execution of CPUs and devices are **competing for memory cycles**

COMPUTER-SYSTEM OPERATION

- **I/O** devices and the **CPU** can execute **concurrently**
- Each **device controller** is in charge of a specific device type
- Each **device controller** has a local buffer storage
- Each **device controller** type has an operating system **device driver** to manage it
- **CPU** moves data from/to main memory to/from local buffers
- **I/O** is from the device to local buffer of controller
- Device controller informs CPU that it has finished its operation by causing an **interrupt**

COMPUTER-SYSTEM OPERATION

- **Device Controller**
  - a hardware unit attached to the I/O bus of the computer and works like an interface between a device and a device driver
  - responsible for moving the data between the peripheral devices that it controls and its local buffer storage

- **Device Driver**
  - a specialized software program running as part of the OS that interacts with a device attached to a computer.
  - driver understands the device controller and provides the rest of the operating system with a uniform interface to the device

- **Interrupt** (sometimes referred to as a **trap** or **exception**)
  - a hardware signal from a device to the CPU which tells the CPU that the device needs attention and that the CPU should stop performing what it is doing and respond to the device

COMMON FUNCTIONS OF INTERRUPTS

- Interrupt transfers control to the **interrupt service routine** (**ISR**), generally, through the use of the **interrupt vector table**, which contains the addresses of all the service routines.
- Interrupt architecture must save the address of the interrupted instruction and the contents of the processor status register, so that it can restore them after servicing the interrupt.
- Incoming interrupts are disabled while another interrupt is being processed to prevent a lost interrupt.
- A software-generated interrupt may be raised either by an error or a user request (sometimes called a **trap** or **exception**).
- An OS is interrupt driven.

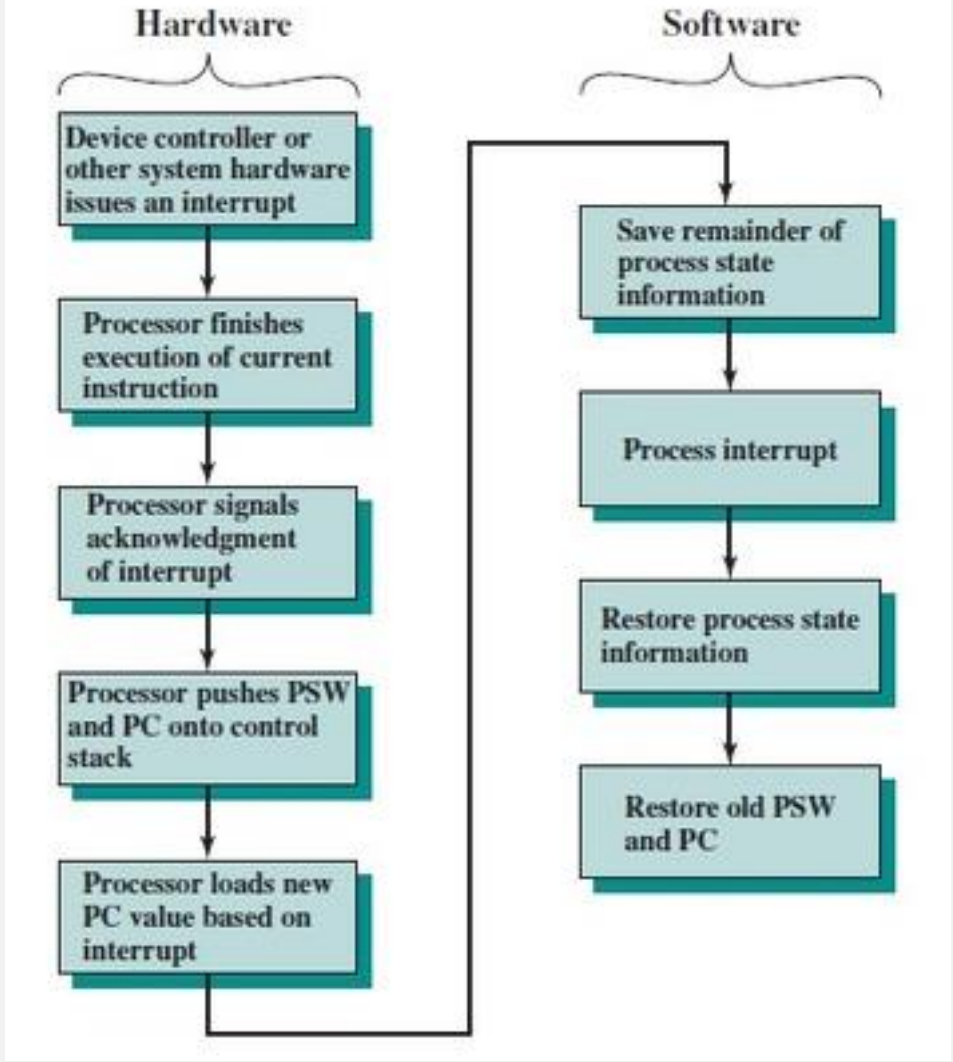
INTERRUPT

- **Interrupt Vector**
  - It is an address, stored as data, located at a particular memory location, which points to where the interrupt service routine can be found.
  - It is an address that informs the interrupt handler as to where to find the ISR.

- **Interrupt Vector Table**
  - It is a table of memory addresses of interrupt/exception handler routines.

- **Interrupt Service Routine (ISR)**
  - It is also known as **Interrupt Service Procedure** or **Interrupt Handler**
  - It is the software that attends to the cause of the interrupt.
  - It is a software routine that hardware invokes in response to an interrupt.

SIMPLE INTERRUPT PROCESSING

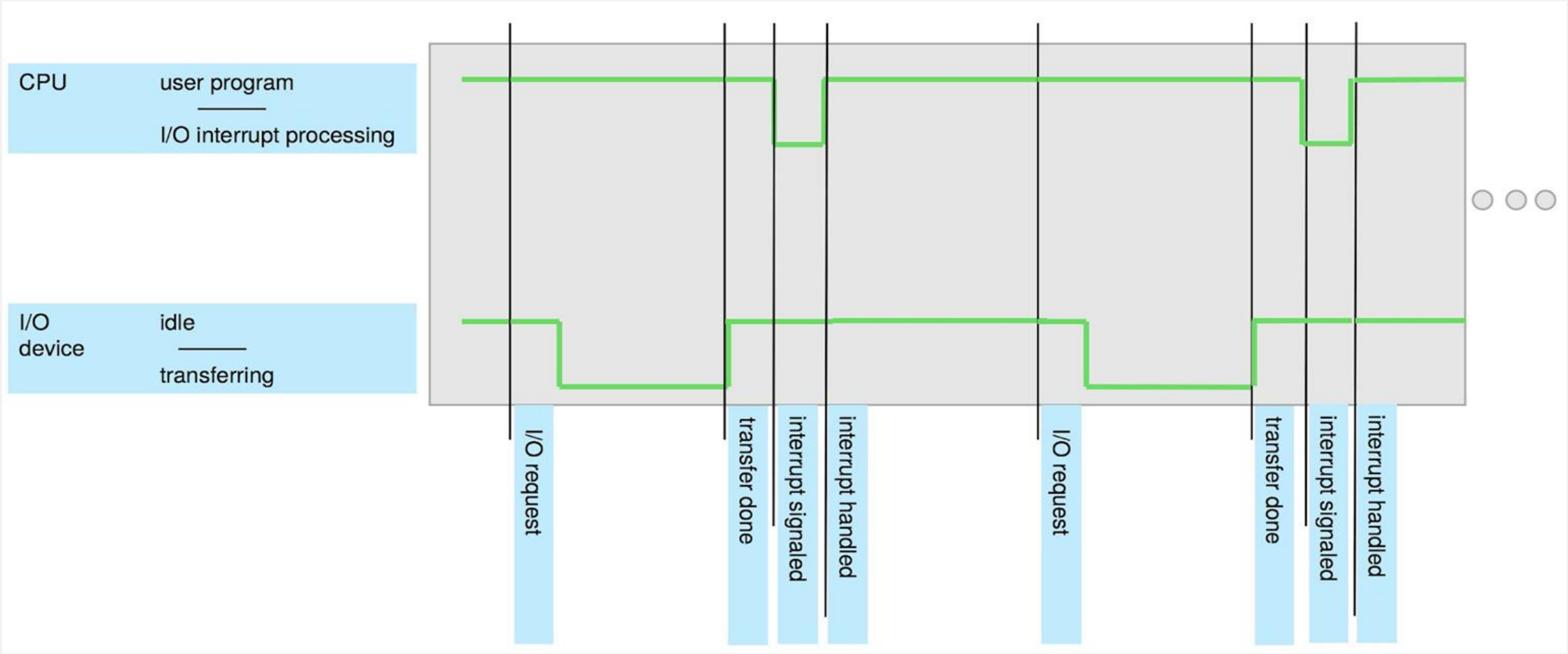


- **Program Status Word (PSW):**
  - A CPU register that contains **status bits** that **reflect the current CPU state**.

- **Program Counter (PC):**
  - A **special-purpose register** that is **used by the processor to hold the address of the next instruction to be executed**.

INTERRUPT TIMELINE

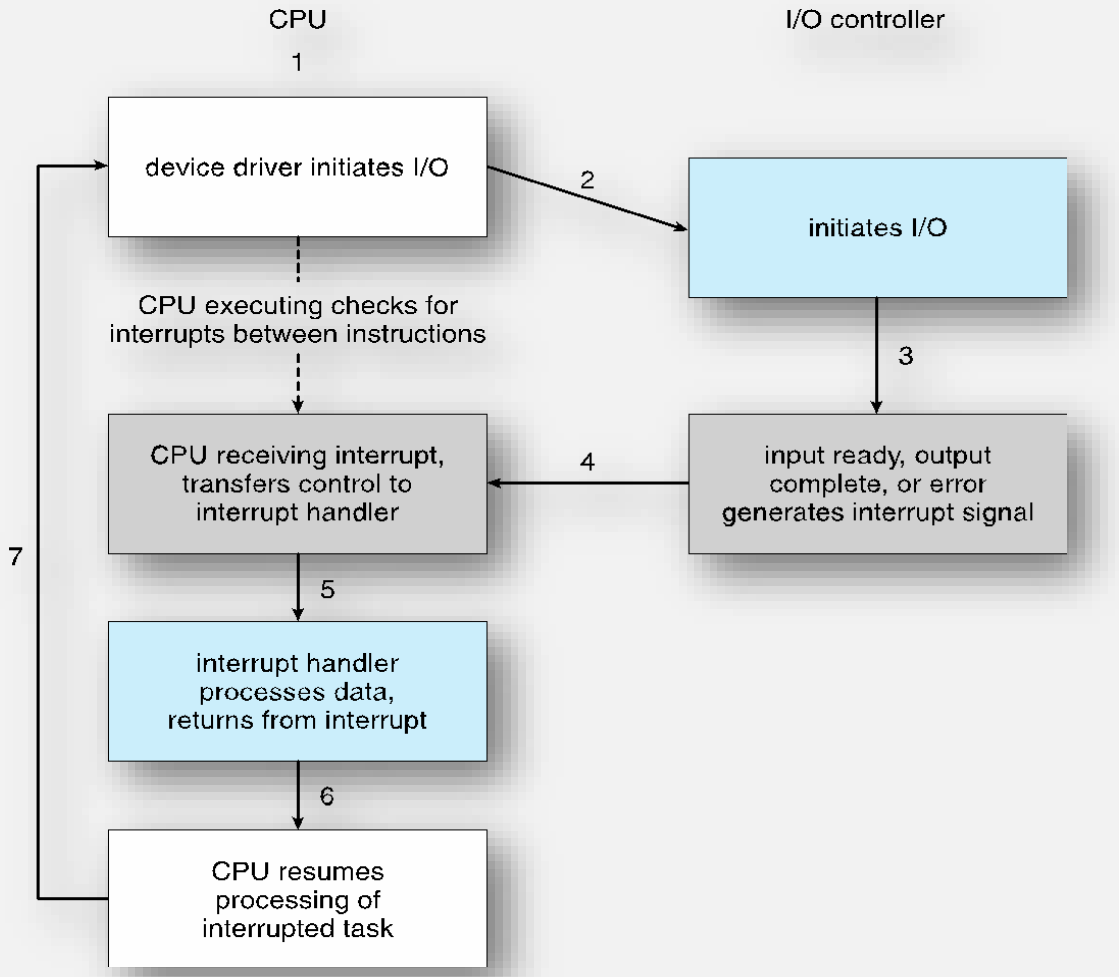
FOR A SINGLE PROGRAM DOING OUTPUT



BASIC INTERRUPT MECHANISM

- The **CPU hardware** has a **wire** called the **interrupt-request line** that the CPU senses after executing every instruction.
- When the **CPU detects** that a **controller has asserted a signal** on the **interrupt-request line**, it **reads the interrupt number** and **jumps** to the **interrupt-handler routine** by using that interrupt number as an **index into the interrupt vector**.
- It then **starts execution at the address associated with that index**.
- The **interrupt handler saves any state it will be changing during its operation**, **determines the cause of the interrupt**, **performs the necessary processing**, **performs a state restore**, and **executes a return from interrupt instruction to return the CPU to the execution state prior to the interrupt**.
- The **device controller** ***raises*** **an interrupt by asserting a signal on the interrupt request line**, the **CPU** ***catches*** **the interrupt and** ***dispatches*** **it to the interrupt handler**, and the **handler** ***clears*** **the interrupt by servicing the device**.

INTERRUPT-DRIVEN I/O CYCLE

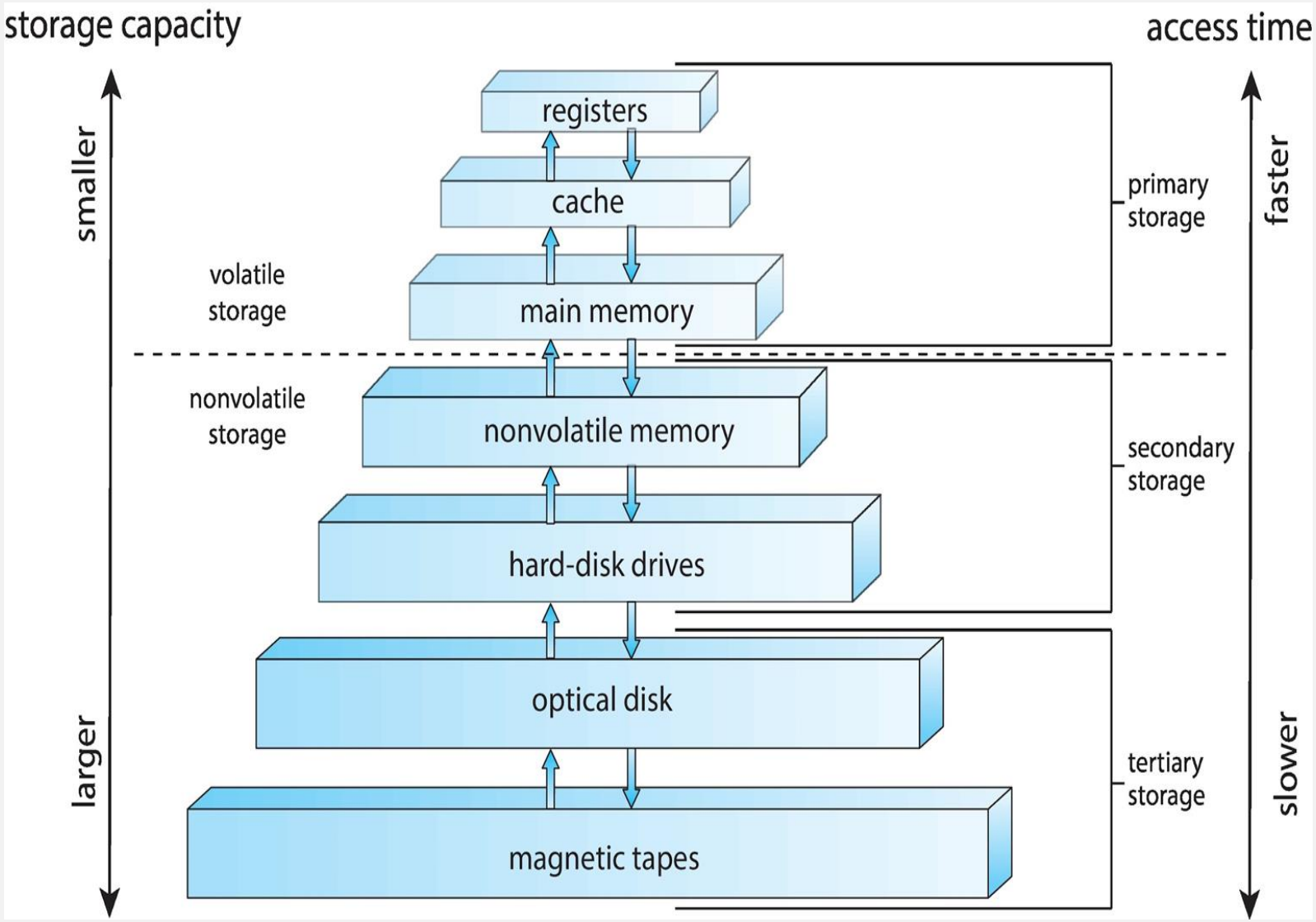


1. A device driver initiates an I/O request on behalf of a process.
2. The device driver signals the I/O controller for the proper device, which initiates the requested I/O.
3. The device signals the I/O controller that is ready to retrieve input, the output is complete or that an error has been generated.
4. The CPU receives the interrupt signal on the interrupt-request line and transfer control over the interrupt handler routine.
5. The interrupt handler determines the cause of the interrupt, performs the necessary processing and executes a **"*return from*"** interrupt instruction.
6. The CPU returns to the execution state prior to the interrupt being signaled.
7. The CPU continues processing until the cycle begins again.

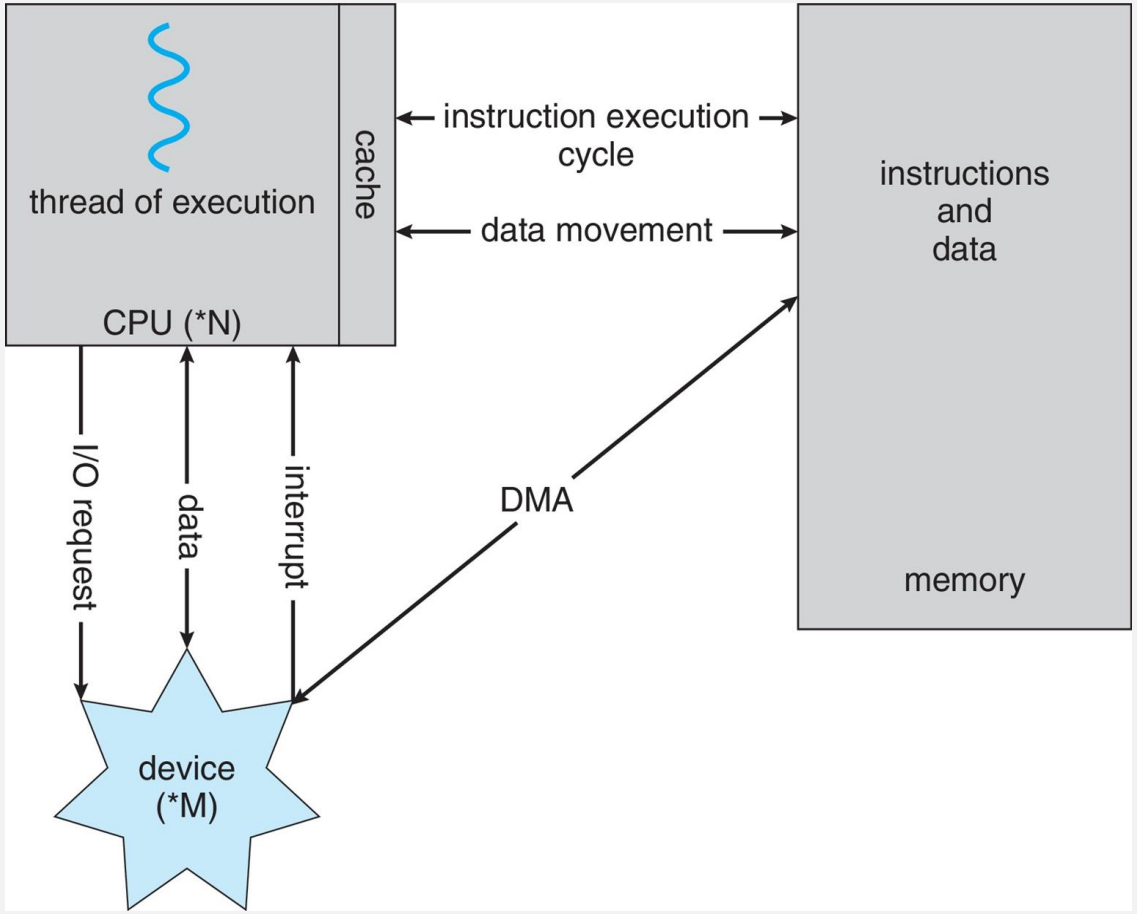
STORAGE STRUCTURE

- **Main Memory**
  - physical memory that is internal to the computer that is a computer's short-term storage.
  - the only large storage media that the CPU can access directly
  - random access and volatile
  - Random-Access Memory in the form of Dynamic Random-Access Memory (**DRAM**)
- **Secondary storage**
  - extension of main memory that provides large **nonvolatile** storage capacity
- **Hard Disk Drives** (**HDD**)
  - rigid metal or glass platters covered with magnetic recording material
  - disk surface is logically divided into **tracks**, which are subdivided into **sectors**
  - **disk controller** determines the logical interaction between the device and the computer
- **Non-Volatile Memory (NVM)**
  - storage devices faster than hard disks and nonvolatile
  - various technologies
  - becoming more popular as capacity and performance increases, price drops

STORAGE HIERARCHY



HOW A MODERN COMPUTER SYSTEM WORKS?



A von Neumann Architecture

DIRECT MEMORY ACCESS STRUCTURE

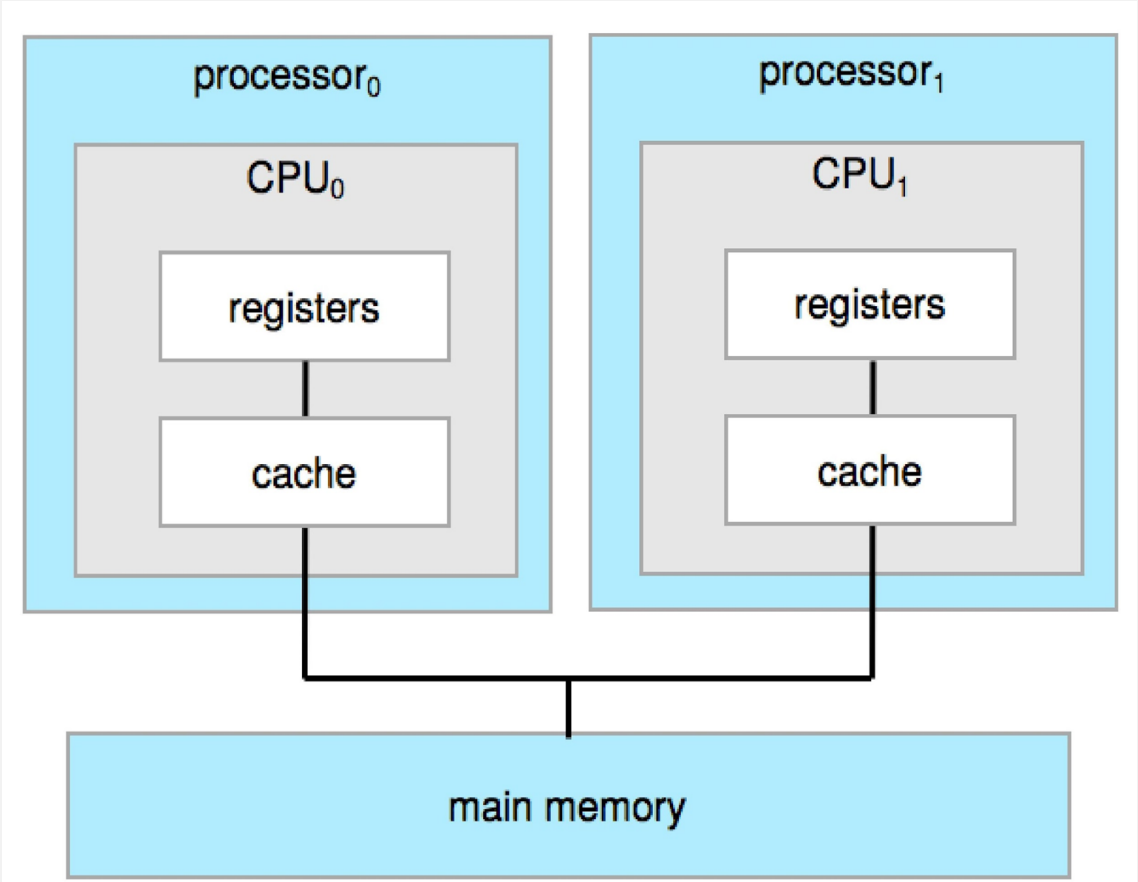
- Used for **high-speed I/O devices** able **to transmit information at close to memory speeds**
- **Device controller** transfers **blocks of data from buffer storage directly to main memory without CPU intervention**
- **Only one interrupt is generated per block**, rather than the one interrupt per byte

COMPUTER-SYSTEM ARCHITECTURE

- **Single-Processor Systems**
  - Computer systems that have a single processor (one CPU with a single processing **core**)
    - **Core:** the component that executes instructions and registers for storing data locally
  - Most systems use a single general-purpose processor
  - Most systems have special-purpose processors as well

- **Multiprocessor Systems**
  - Systems that have two (or more) processors, each with a single-core CPU
  - Systems that are growing in use and importance
  - Also known as **parallel systems** or **tightly-coupled systems**
  - **Two types:**
    - **Asymmetric Multiprocessing:** each processor is assigned a specific task
    - **Symmetric Multiprocessing:** each processor performs all tasks

SYMMETRIC MULTIPROCESSING (SMP) ARCHITECTURE



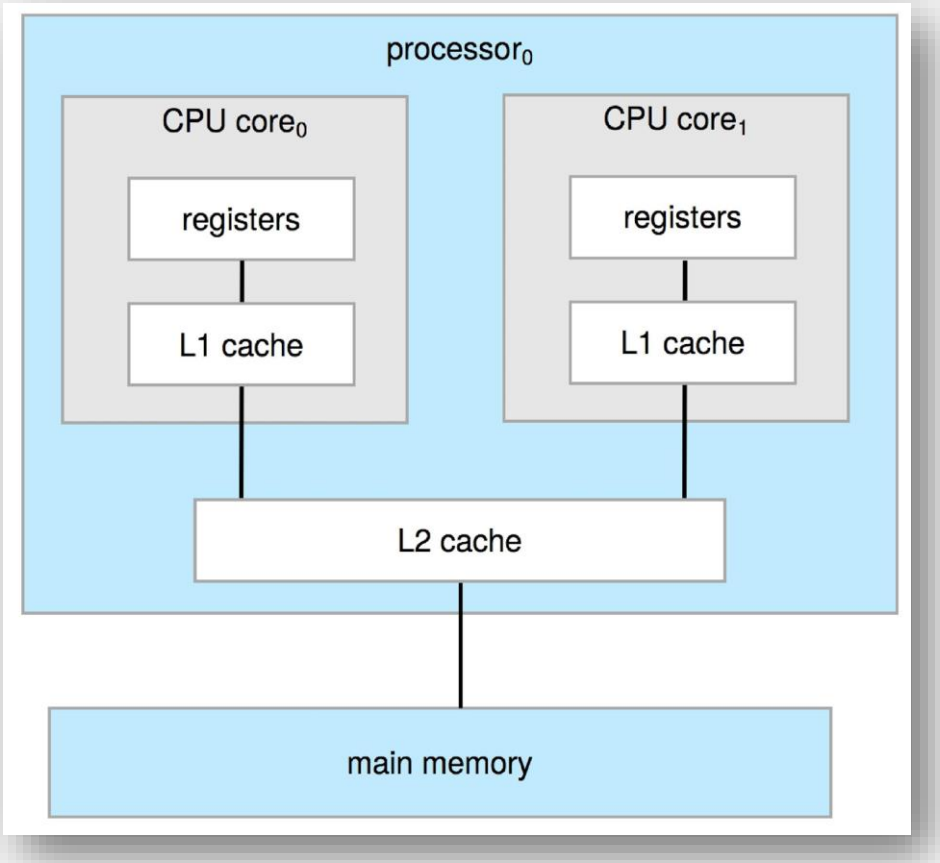
- A **multiprocessor system** in which **each peer CPU processor performs all tasks**, including **operating-system functions** and **user processes**.

A DUAL-CORE DESIGN

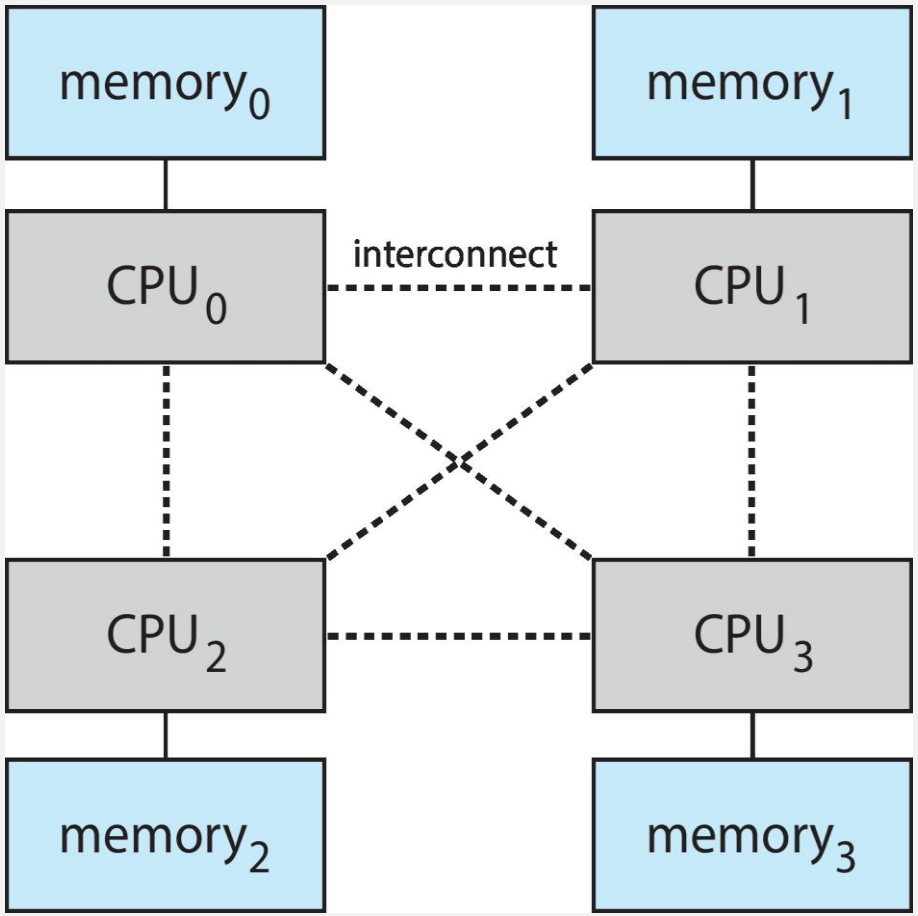
- **Multicore Systems:**
  - **multiple computing cores reside on a single chip**
  - can be **more efficient** than **multiple chips with single cores** because **on-chip communication** is **faster than between-chip communication**

- **Dual-Core Processor:**
  - A **single chip** that contains **two distinct processors** that **work simultaneously**

- **Core:** the **basic computation unit** of the CPU



NON-UNIFORM MEMORY ACCESS (NUMA) SYSTEM

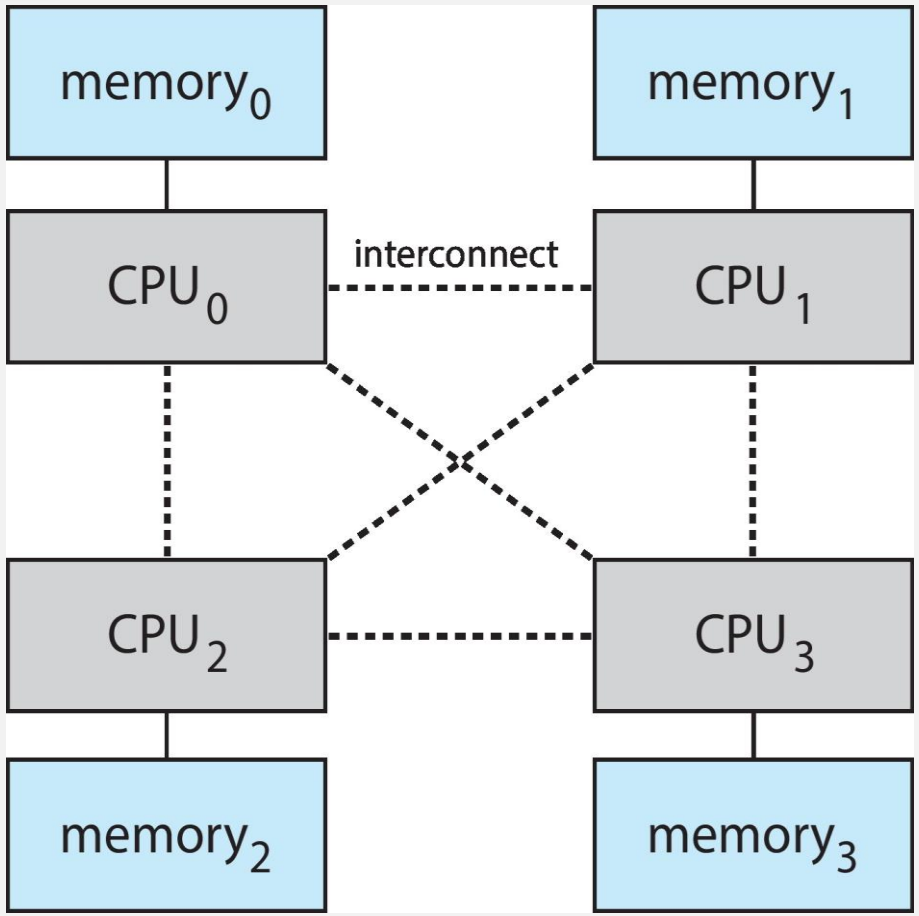


NUMA Multiprocessing Architecture

- **Fact:**
  Adding additional CPUs to a multiprocessor system will **increase computing power**

- **Issue:**
  The **above concept does not scale very well**, and **once too many CPUs are added**, **contention for the system bus** becomes a **bottleneck** and **performance begins to degrade**.

NON-UNIFORM MEMORY ACCESS (NUMA) SYSTEM

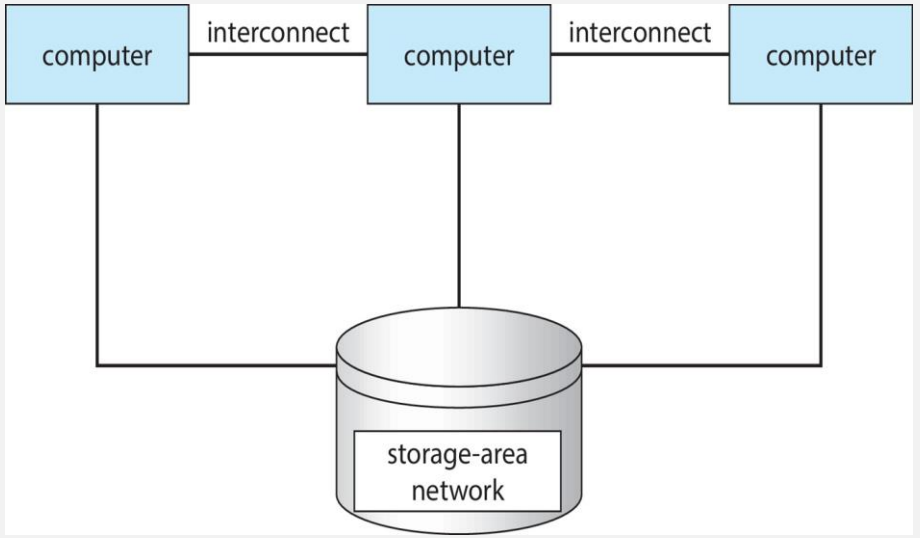


NUMA Multiprocessing Architecture

- **Solution (Alternative Approach): NUMA**
  - **Provide each CPU (or group of CPUs) with its own local memory** that is **accessed via a small, fast local bus**
  - The **CPUs are connected** by a **shared system interconnect**, so that **all CPUs share one physical address space**

CLUSTERED SYSTEMS

- Another **type of multiprocessor systems,** which **gathers together multiple CPUs**
- **Composed of two or more individual systems (or nodes) joined together**; **each node is typically a multicore system**
- Considered as **loosely coupled systems**
- Usually **share storage** via a **Storage Area Network (SAN)**
- **Closely linked** via a **Local Area Network** (**LAN**)



CHARACTERISTICS OF CLUSTERED SYSTEMS

- Provides a **high-availability service** which **survives failures**
  - Can be structured as follows:
    - **Asymmetric clustering:**
      - ✓has one machine in hot-standby mode while the other is running the applications
    - **Symmetric clustering**:
      - ✓has multiple nodes running applications, monitoring each other

- Some clusters are for **High-Performance Computing (HPC)**
  - **Applications** must be written to use **parallelization** (divides a program into separate components that run in parallel on individual cores in a computer or computers in a cluster)

- Some have **Distributed Lock Manager** (**DLM**) to **avoid conflicting operations**

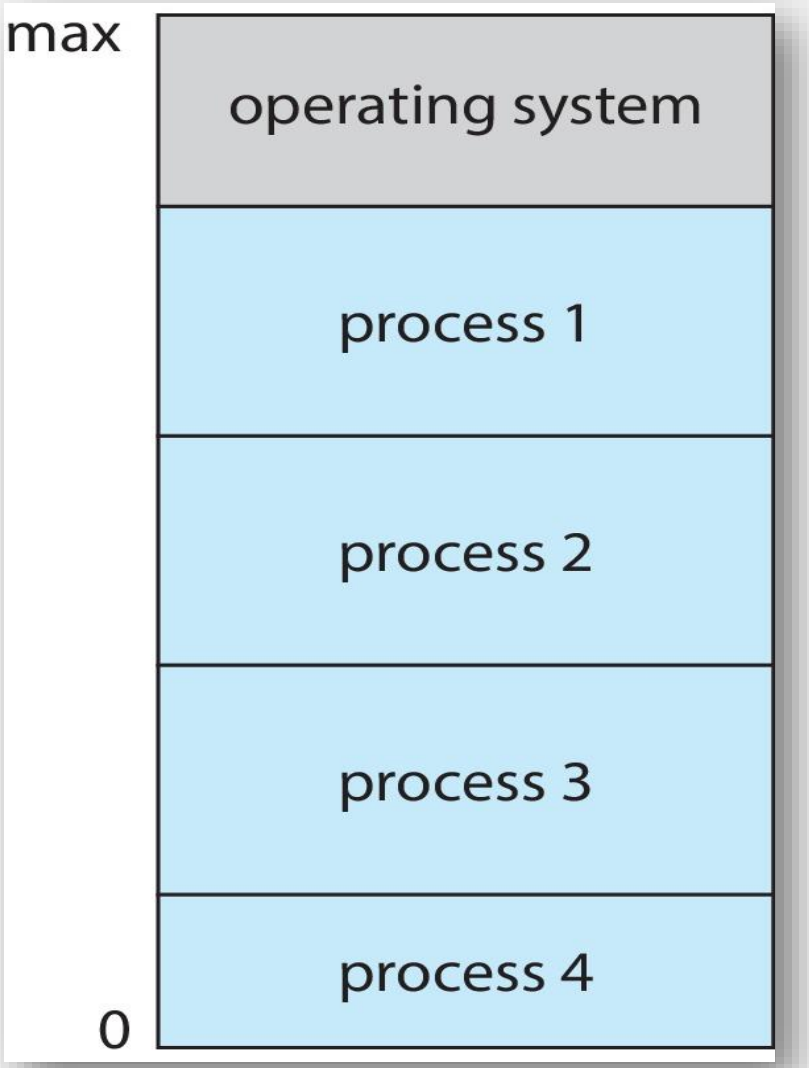
OPERATING-SYSTEM OPERATIONS

- **Bootstrap Program**
  - Needed to power-up or reboot the computer system
  - Typically stored in ROM, EPROM, or another non-volatile memory (in **firmware**)
  - Simple code to initialize the system
  - Loads operating system kernel and starts execution
- Starts **system daemons** (services provided outside of the kernel)
- Kernel is **interrupt driven** (hardware and software)
  - Hardware interrupt by one of the devices
  - Software interrupt (**exception** or **trap):**
    - Software error (e.g., division by zero)
    - Request for operating system service: **system call**
    - Other process problems include infinite loop, processes modifying each other or the operating system

MULTIPROGRAMMING

- **Multiprogramming** (**Batch system**): needed for efficiency
  - Single user cannot keep CPU and I/O devices busy at all times
  - Multiprogramming organizes jobs (code and data) so CPU always has one to execute
  - A subset of total jobs in system is kept in memory
  - One job selected and run via **job scheduling**
  - When it has to wait (for I/O for example), OS switches to another job

MEMORY LAYOUT FOR MULTIPROGRAMMED SYSTEM



MULTITASKING

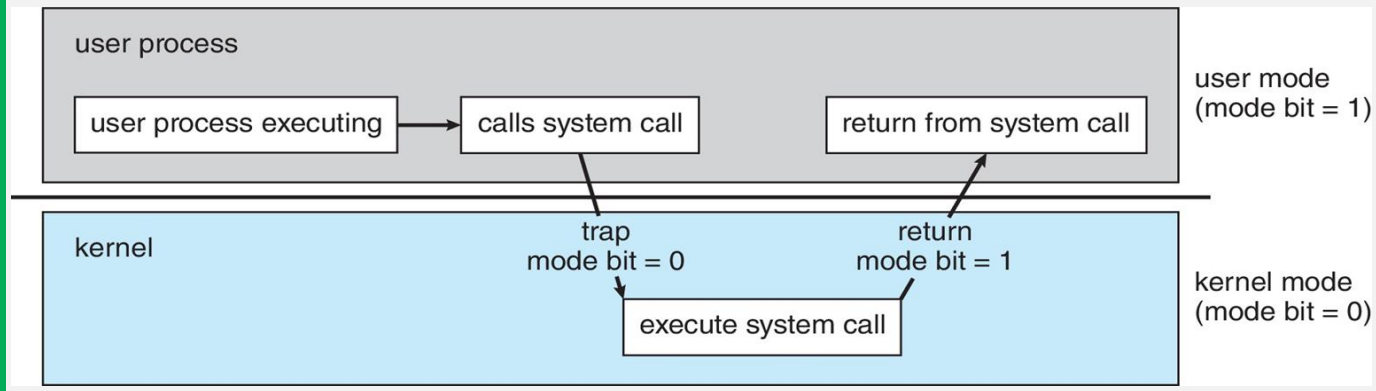
- **Multitasking (Timesharing)**:
  - Logical extension of multiprogramming
  - **CPU switches jobs so frequently** that users can interact with each job while it is running
  - **Response time** should be **< 1 second**
  - Each user has at least one program executing in memory (**process**)
  - If several jobs are ready to run at the same time (**CPU Scheduling**)
  - If processes don't fit in memory, **swapping** moves them in and out to run
  - **Virtual memory** allows execution of processes not completely in memory

DUAL-MODE AND MULTIMODE OPERATION

- **Dual-Mode** operation allows OS to protect itself and other system components
  - **User mode** and **Kernel mode**
  - **Mode bit** is provided by hardware
    - Provides ability to distinguish when system is running user code or kernel code
    - Some instructions are designated as **privileged** (only executable in kernel mode)
    - **System call changes mode to kernel, return from call resets it to user**

- Many CPUs support multimode operations
  - **Virtual Machine Manager** (**VMM**) can create and manage **VMs**

TRANSITION FROM USER TO KERNEL MODE



- **Life cycle of instruction execution in a computer system:**
  - **Initial control resides in the operating system**, where instructions are executed in **kernel mode**.
  - When **control is given to a user application**, the mode is set to **user mode**.
  - Eventually, **control is switched back to the OS** via an **interrupt, a trap, or a system call**.

- **System call:** a call (function) that is required to invoke the services provided by the operating system

- **How Does a System Call Work?**

1. Initially, a processor executes a user program in the user mode.
2. Then if the program needs the services of the operating system, the processor is interrupted by the system call.
3. A system call is always prioritized over the other executions and the system call is then executed in the kernel mode.
4. Once the system call is executed completely, the control goes back to the user mode.
5. And the execution of the program resumes back in the user mode.

TIMER

- **Timer** is used to **prevent infinite loop / process hogging resources**
  - Timer is set to interrupt the computer after a specified period
  - Keep a counter that is decremented by the physical clock
  - Operating system set the counter (privileged instruction)
  - When the counter reaches 0, an interrupt occurs
  - Set up before scheduling process to regain control or terminate program that exceeds allotted time

- **NOTE:**
  - **Process Hogging Resources:**
    - A process which consumes a large amount of system resources compared to its importance or function.

UNIVERSITY of SAN CARLOS
SCIENTIA • VIRTUS • DEVOTIO

DCISM
DEPARTMENT OF COMPUTER, INFORMATION SCIENCES AND MATHEMATICS

1

CS 3104
OPERATING SYSTEMS

*** END ***

THANK YOU!

UNIVERSITY of SAN CARLOS
SCIENTIA • VIRTUS • DEVOTIO

DCISM
DEPARTMENT OF COMPUTER, INFORMATION SCIENCES AND MATHEMATICS

2

CS 3104
OPERATING SYSTEMS

CHAPTER 1

RESOURCE MANAGEMENT: IMPORTANT FACTS

- An **operating system** is a **resource manager**.

- **Resources** that **OS must manage**:
  - system's CPU
  - memory space
  - file-storage space
  - I/O devices

PROCESS MANAGEMENT

- **Process**:
  - It is a **program in execution**.
  - It is a **unit of work within the system**

- **Program** is a **passive entity**
- **Process** is an **active entity**

- **Process needs resources** to accomplish its task:
  - CPU, memory, I/O, Files
  - Initialization data (input)

- **Important Note:**
  - When the **process terminates**, the **OS will reclaim** any **reusable resources**.

PROCESS MANAGEMENT

- **Single-threaded process** has one **program counter** specifying location of the next instruction to execute
- **Process** executes instructions **sequentially, one at a time**, until completion
- **Multithreaded process** has **one program counter** per thread
- Typically, a **system has many processes** running concurrently on one or more CPUs.
  - **Operating-system processes** (**those that execute system code)**
  - **User processes** (**those that execute user code**)
  - **Concurrency** is done by **multiplexing the CPUs** among the **processes / threads**

PROCESS MANAGEMENT ACTIVITIES

- The OS is responsible for the following activities in connection with process management:
  - Creating and deleting both user and system processes
  - Scheduling processes and threads on the CPUs
  - Suspending and resuming processes
  - Providing mechanisms for process synchronization
  - Providing mechanisms for process communication
  - Providing mechanisms for deadlock handling

MEMORY MANAGEMENT

- To **execute a program**, **all (or part) of the instructions must be in memory**
- **All (or part) of the data** that are **needed by the program must be in memory**

- **Memory management** determines:
  - what is in memory and
  - when optimizing CPU utilization and computer's response to users

- **Memory management activities:**
  - Keeping track of which parts of memory are currently being used and which process is using them
  - Deciding which processes (or parts of processes) and data to move into and out of memory
  - Allocating and deallocating memory space as needed

FILE-SYSTEM MANAGEMENT

- **OS** provides **uniform, logical view of information storage**
  - OS abstracts physical properties to logical storage unit (**file)**
  - Each medium is controlled by device (i.e., disk drive, tape drive)
    - **Varying properties** include:
      - **access speed**
      - **capacity**
      - **data-transfer rate**
      - **access method** (sequential or random)

- **File-System management:**
  - **Files** usually **organized into directories**
  - **Access control on most systems** to determine who can access what
  - **OS activities include:**
    - Creating and deleting files
    - Creating and deleting directories to organize files
    - Supporting primitives for manipulating files and directories
    - Mapping files onto mass storage
    - Backing up files on stable (nonvolatile) storage media

MASS-STORAGE MANAGEMENT

- Usually, **these are disks**:
  - used to store data that does not fit in main memory, or
  - data that must be kept for a “long” period of time

- Proper management is of central importance

- Entire speed of computer operation hinges on disk subsystem and its algorithms

MASS-STORAGE MANAGEMENT

- **OS activities with secondary storage management:**
  - Mounting and unmounting
  - Free-space management
  - Storage allocation
  - Disk scheduling
  - Partitioning
  - Protection

- Some storage need not be fast
  - Tertiary storage includes magnetic tape, optical storage, and Blu-ray drives
  - Not crucial but still must be managed (by OS or applications)

CACHE MANAGEMENT

- **Caching:**
  - Important principle of computer systems
  - Performed at many levels in a computer **(in hardware, operating system, software)**

- Information in use is copied from slower to faster storage temporarily

- **Faster storage (cache)** is checked first to determine if information is there
  - If it is, information is used directly from the cache (fast)
  - If not, information is copied to cache and it is used there

- **Cache is smaller** than the storage being cached
  - Cache management is important in design problem
  - Cache size and replacement policy are also considered

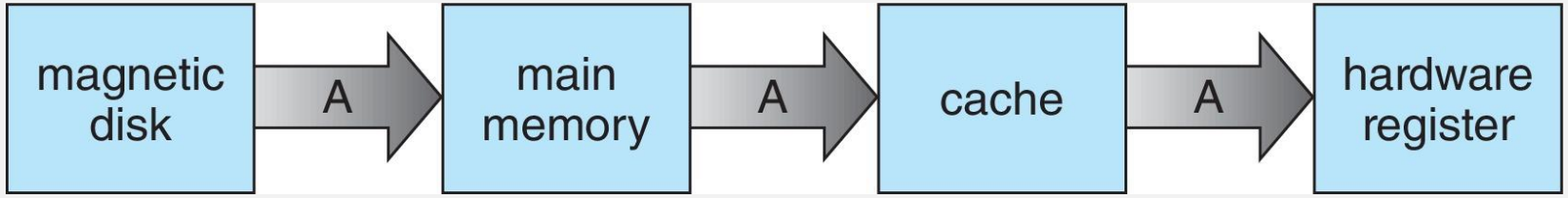
CHARACTERISTICS OF VARIOUS TYPES OF STORAGE

Level	1	2	3	4	5
Name	registers	cache	main memory	solid-state disk	magnetic disk
Typical size	< 1 KB	< 16MB	< 64GB	< 1 TB	< 10 TB
Implementation technology	custom memory with multiple ports CMOS	on-chip or off-chip CMOS SRAM	CMOS SRAM	flash memory	magnetic disk
Access time (ns)	0.25-0.5	0.5-25	80-250	25,000-50,000	5,000,000
Bandwidth (MB/sec)	20,000-100,000	5,000-10,000	1,000-5,000	500	20-150
Managed by	compiler	hardware	operating system	operating system	operating system
Backed by	cache	main memory	disk	disk	disk or tape

- **Movement between levels of storage hierarchy can be explicit or implicit**

MIGRATION OF DATA "A" FROM DISK TO REGISTER

- **Multitasking environments** must be careful to use most recent value, no matter where it is stored in the storage hierarchy



- **Multiprocessor environment** must provide **cache coherency** in hardware such that all CPUs have the most recent value in their cache

- **Distributed environment** situation **is even more complex**
  - Several copies of a datum can exist
  - Various solutions are covered in Chapter 19

I/O SUBSYSTEM

- **One purpose of OS** is to **hide peculiarities of hardware devices from the user**

- **Peculiarities of I/O devices are hidden** from the bulk of the operating system itself by the **I/O subsystem**

- **I/O subsystem** is responsible for:
  - **Memory management of I/O** including;
    - **buffering** (storing data temporarily while it is being transferred),
    - **caching** (storing parts of data in faster storage for performance),
    - **spooling** (the overlapping of output of one job with input of other jobs)
  - **General device-driver interface**
  - **Drivers for specific hardware devices**

SECURITY AND PROTECTION

- **Security:** defense of the system against internal and external attacks
  - Huge range: denial-of-service, worms, viruses, identity theft, theft of service

- **Protection:** any mechanism for controlling access of processes or users to resources defined by the OS

- **Systems** generally first distinguish among users, to determine who can do what:
  - **User identities** (user IDs, security IDs) include name and associated number, one per user
  - **User ID** then associated with all files, processes of that user to determine access control
  - **Group Identifier** (**group ID**) allows set of users to be defined and controls management, then also associated with each process, file
  - **Privilege escalation** allows user to change to effective ID with more rights

VIRTUALIZATION

- The **creation of a virtual (rather than actual) version of something**, such as an operating system (OS), a server, a desktop, a storage device or network resources.
- A **technology** that **allows us to abstract the hardware of a single computer** into **several different execution environments**
- Creating the illusion that each separate environment is running on its own private computer
- Allows operating systems to run as applications within other Operating Systems
- Vast and growing industry

VIRTUALIZATION

- **Virtualization software is one member of a class that also includes emulation**
  - **Emulation** is **used when source CPU type is different from target type**
    - (i.e., PowerPC to Intel x86)
    - generally, it is the slowest method
    - emulated code may run much more slowly than the native code

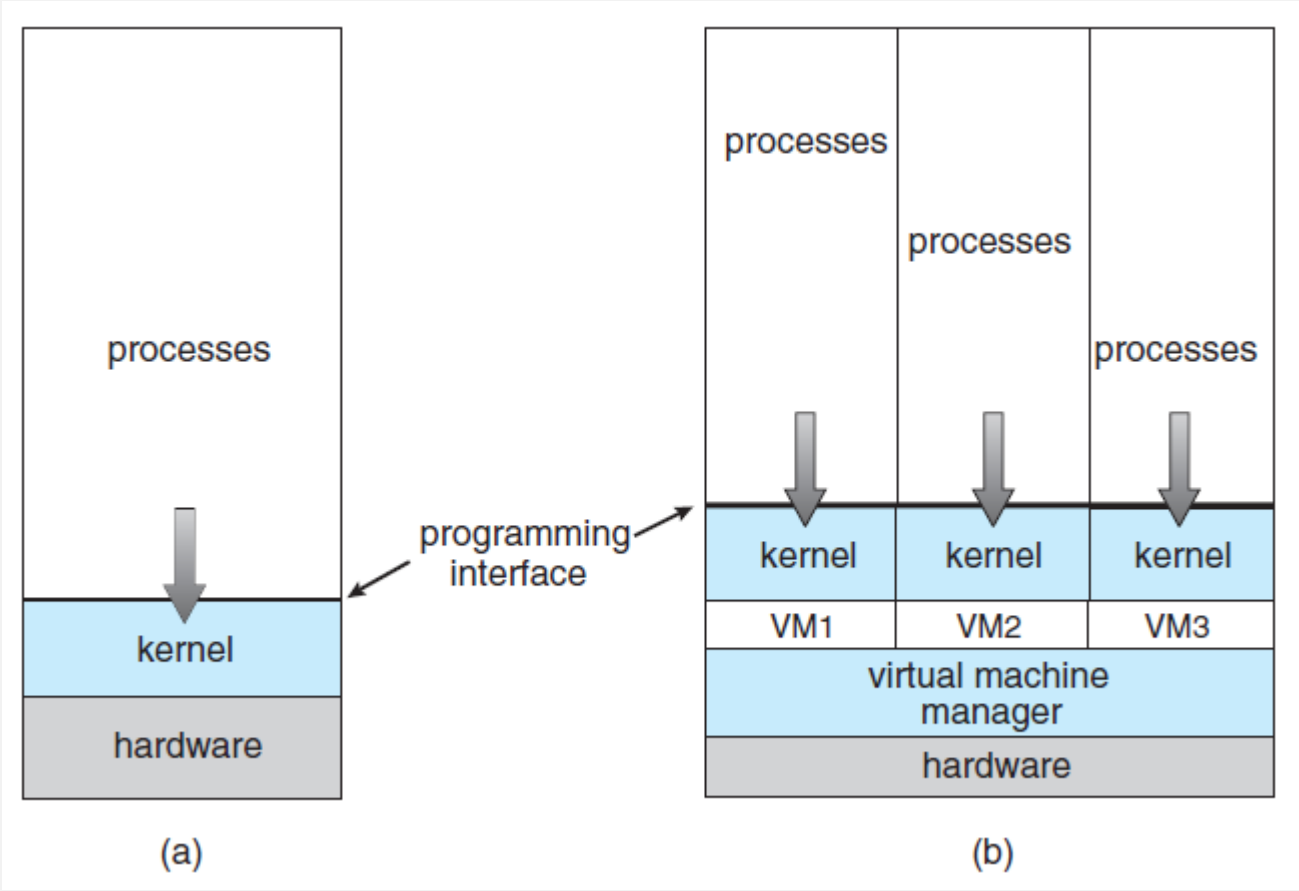
- **Virtualization: OS natively compiled for CPU**, running **guest** Operating Systems also natively compiled
  - Consider **VMware running WinXP guests**, **each running applications**, **all on native WinXP host OS**
  - **VMM** (**V**irtual **M**achine **M**anager) provides **virtualization services**

VIRTUALIZATION

- **Use cases** that **involve laptops and desktops running multiple Operating Systems for exploration or compatibility**
  - **Apple laptop** running **Mac OS X host**, **Windows as a guest**
  - **Developing apps** for multiple Operating Systems without having multiple systems
  - **QA testing applications** without having multiple systems
  - **Executing and managing compute environments** within **data centers**

- **VMM** can **run natively**, in which case **they are also the host**
  - There is **no general purpose host** then (**VMware ESX** and **Citrix XenServer**)

COMPUTING ENVIRONMENTS: VIRTUALIZATION



A computer running (a) a single operating system and (b) three virtual machines.

DISTRIBUTED SYSTEMS

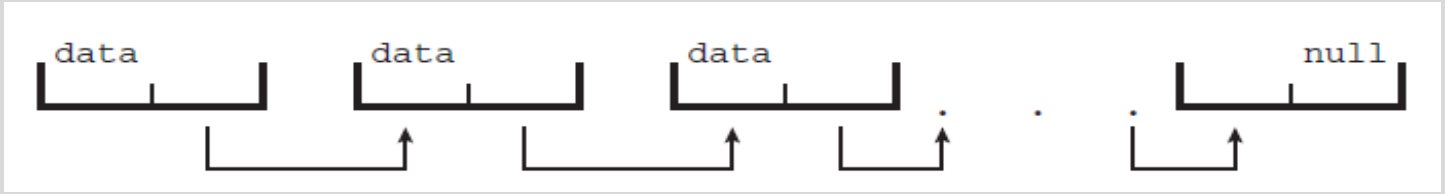
- **Distributed computing**: a field of **computer science** that **studies distributed systems**

- **Distributed Systems:**
  - **Collection** of **physically separate, possibly heterogeneous computer systems networked together**
    - **Network** is a **communication path between 2 or more systems**
      - **TCP/IP** is the most common network protocol used
        - **Local Area Network (LAN)**
        - **Wide Area Network (WAN)**
        - **Metropolitan Area Network (MAN)**
        - **Personal Area Network (PAN)**

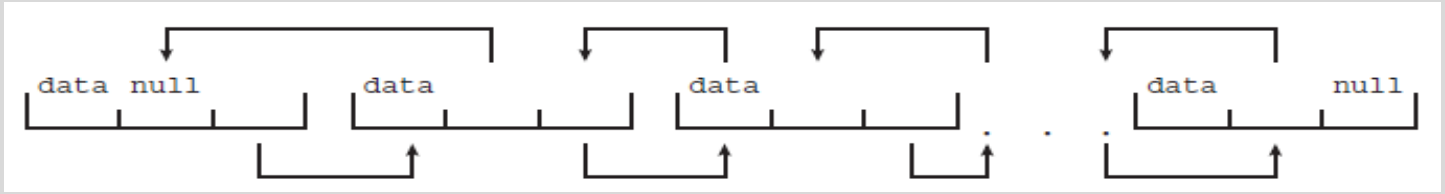
- **Network Operating System:** provides features between systems across network
  - Communication scheme allows systems to exchange messages
  - Illusion of a single system

KERNEL DATA STRUCTURES

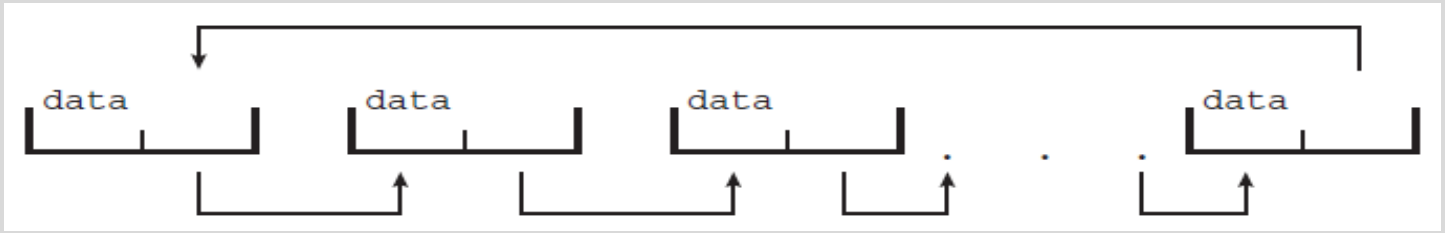
- **Many similar to standard programming data structures:**
  - **Singly linked list**



  - **Doubly linked list**

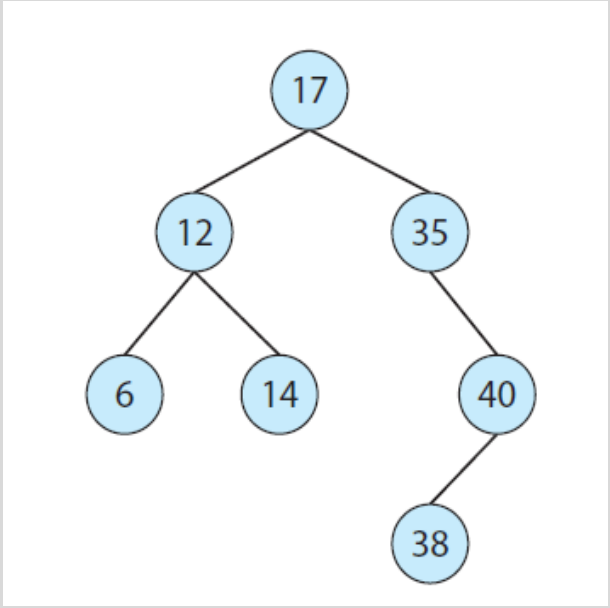


  - **Circularly linked list**



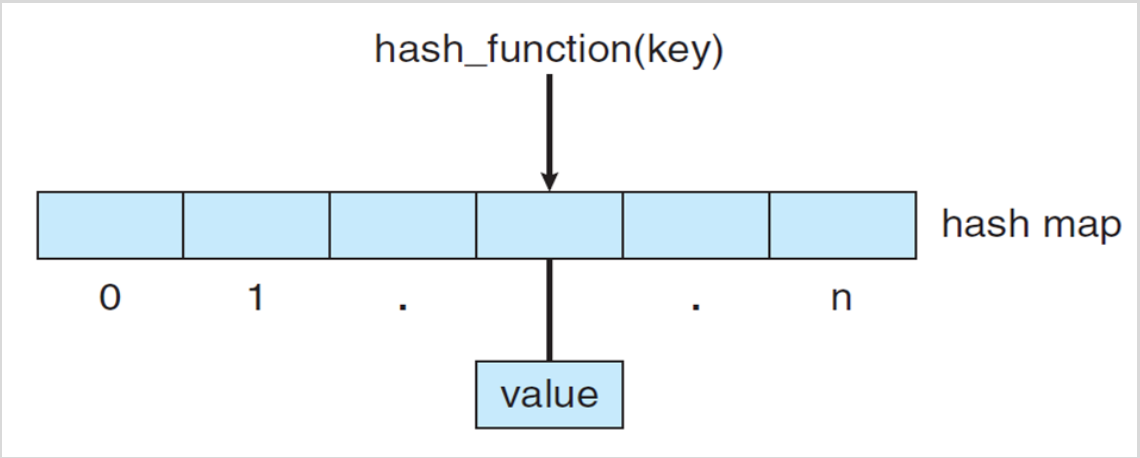
KERNEL DATA STRUCTURES

- **Binary Search Tree:** left <= right
  - **Search performance** is *O(n)*
  - **Balanced binary search tree** is *O(log n)*



KERNEL DATA STRUCTURES

- **Hash function** can create a **hash map**
- **Hash Map:**
  - which **associates** (or **maps**) [**key:value**] **pairs** using **a hash function**



- **Hash function:**
  - **takes data as its input, performs a numeric operation on the data**, and **returns a numeric value**
  - this **numeric value** can then be **used as an index into a table** (typically an array) **to quickly retrieve the data**

- **Example:**
  Suppose that a user name is mapped to a password
  - **Password authentication** then proceeds as follows: **a user enters her user name and password**
  - **Hash function** is **applied to the user name**, which is then **used to retrieve the password**
  - The **retrieved password** is then **compared with the password entered by the user** for **authentication**

KERNEL DATA STRUCTURES

- **Bitmap:**
  - a **string of *n* binary digits** representing **the status of *n* items**

- **Example:**
  Suppose we have several resources; availability of each resource is **indicated by the value of a binary digit**: **0** means **resource is available**, while **1** indicates that **it is unavailable (or vice versa)**

- Consider the **bitmap** shown below:
  001011101
  - Resources 2, 4, 5, 6, and 8 are unavailable
  - Resources 0, 1, 3, and 7 are available

- **Bitmaps:** commonly used when there is a need to represent the availability of a large number of resources.
- **Example:** Disk drives
  - Medium-sized disk drive is divided into several thousand individual units (**disk blocks**)
  - A **bitmap** can be used to **indicate the availability of each disk block**

LINUX KERNEL DATA STRUCTURES

- The **data structures** used in the Linux kernel are available in the kernel source code.
- The ***include*** file **<linux/list.h>** provides details of the linked-list data structure used throughout the kernel.
- A **queue** in Linux is known as a **kfifo**, and its implementation can be found in the **kfifo.c** file in the **kernel** directory of the source code.
- Linux also provides a balanced binary search tree implementation using ***red-black trees***.
- **Details** can be found in the include file **<linux/rbtree.h>**.

COMPUTING ENVIRONMENTS: TRADITIONAL

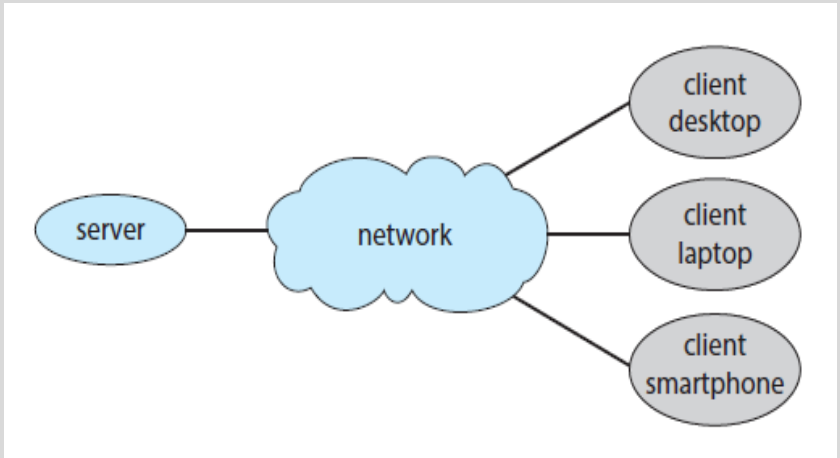
- **Stand-alone** general-purpose machines
- But **blurred**, as most systems interconnect with others (i.e., the Internet)
- **Portals** provide web access to internal systems
- **Network computers** (**thin clients**) are like Web terminals
- **Mobile computers** interconnect via **wireless networks**
- **Networking** becoming ubiquitous (even home systems use **firewalls** to protect home computers from Internet attacks)

COMPUTING ENVIRONMENTS: MOBILE

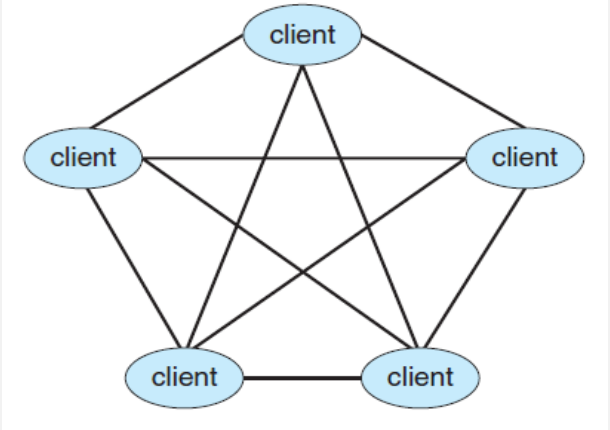
- Handheld smartphones, tablets, etc.
- What is the functional difference between them and a “traditional” laptop?
- Extra feature – more OS features (GPS, gyroscope)
- Allows new types of apps like ***augmented reality***
- Use IEEE 802.11 wireless, or cellular data networks for connectivity
- Leaders are **Apple iOS** and **Google Android**

COMPUTING ENVIRONMENTS: CLIENT-SERVER

- **Client-Server Computing:**
  - Dumb terminals supplanted by smart PCs
  - Many systems now **servers**, responding to requests generated by **clients**
    - **Compute-server system:** provides an interface to client to request services
      - ✓(i.e., database)
    - **File-server system:** provides interface for clients to store and retrieve files



COMPUTING ENVIRONMENTS: PEER-TO-PEER



- Another model of distributed system
- P2P does not distinguish clients and servers
  - Instead all nodes are considered peers
  - May each act as client, server or both
  - Node must join P2P network
    - Registers its service with central lookup service on network, or
    - Broadcast request for service and respond to requests for service via ***discovery protocol***
    - Examples include Napster and Gnutella, **Voice over IP** (**VoIP**) such as Skype

COMPUTING ENVIRONMENTS: CLOUD COMPUTING

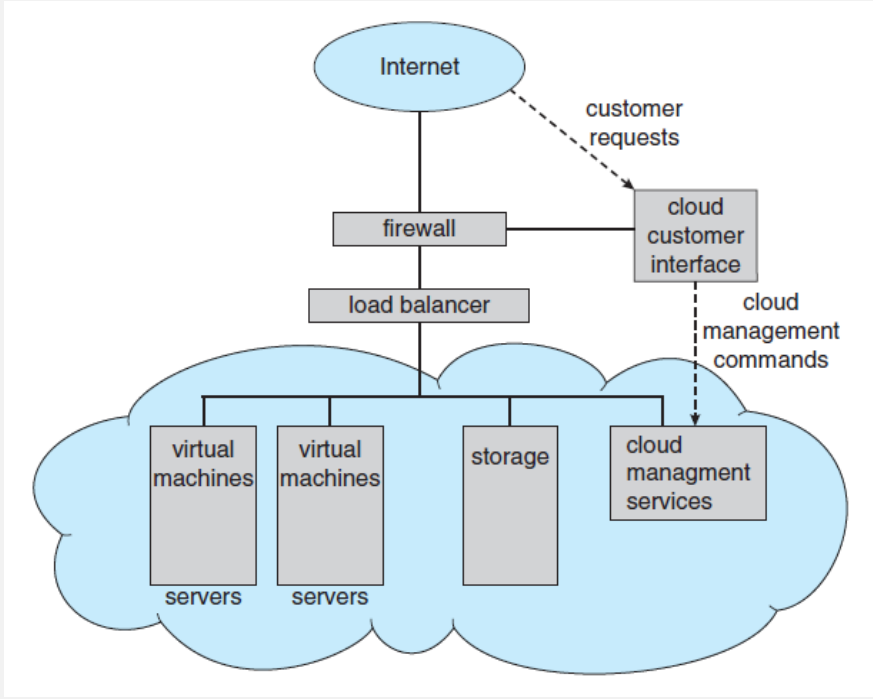
- Delivers computing, storage, even apps as a service across a network
- Logical extension of virtualization because it uses virtualization as the base for its functionality
- Amazon **EC2** has thousands of servers, millions of virtual machines, petabytes of storage available across the Internet, pay is based on usage

COMPUTING ENVIRONMENTS: CLOUD COMPUTING

- **Many types:**
  - **Public cloud:** available via Internet to anyone willing to pay
  - **Private cloud:** run by a company for the company's own use
  - **Hybrid cloud:** includes both public and private cloud components
  - **S**oftware **a**s **a** **S**ervice (**SaaS**): one or more applications available via the Internet (i.e., word processor)
  - **P**latform **a**s **a** **S**ervice (**PaaS**): software stack ready for application use via the Internet (i.e., a database server)
  - **I**nfrastructure **a**s **a** **S**ervice (**IaaS**): servers or storage available over Internet (i.e., storage available for backup use)

COMPUTING ENVIRONMENTS: CLOUD COMPUTING

- **Cloud computing environments:**
  - composed of traditional Operating Systems, plus VMMs, plus cloud management tools
    - Internet connectivity requires security like firewalls
    - Load balancers spread traffic across multiple applications



REAL-TIME EMBEDDED SYSTEMS

- **Real-time embedded systems** are the most prevalent form of computers
  - Vary considerable, special purpose, limited purpose OS: **real-time OS**
  - Use expanding

- Many other special computing environments as well
  - Some have Operating System
  - Some perform tasks without an OS

- Real-time OS has well-defined fixed time constraints
  - Processing ***must*** be done within constraint
  - Correct operation only if constraints met

FREE AND OPEN-SOURCE OPERATING SYSTEMS

- **Operating systems** made available in source-code format rather than just binary **closed-source** and **proprietary**

- Counter to the **copy protection** and **D**igital **R**ights **M**anagement **(DRM)** movement

- Started by **F**ree **S**oftware **F**oundation **(FSF)**, which has **"copyleft" GNU Public License (GPL)**
  - Free software and open-source software are two different ideas championed by different groups of people

FREE AND OPEN-SOURCE OPERATING SYSTEMS

- Examples include **GNU/Linux** and **BSD UNIX** (including core of **Mac OS X**), and **many more**

- Can use **VMM** like **VMware Player** (Free on Windows), **Virtualbox** (open source and free on many platforms - https://www.virtualbox.org)
  - Used to run guest operating systems for exploration

UNIVERSITY of SAN CARLOS
SCIENTIA • VIRTUS • DEVOTIO

DCISM
DEPARTMENT OF COMPUTER, INFORMATION SCIENCES AND MATHEMATICS

2

CS 3104
OPERATING SYSTEMS

*** END ***

THANK YOU!

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

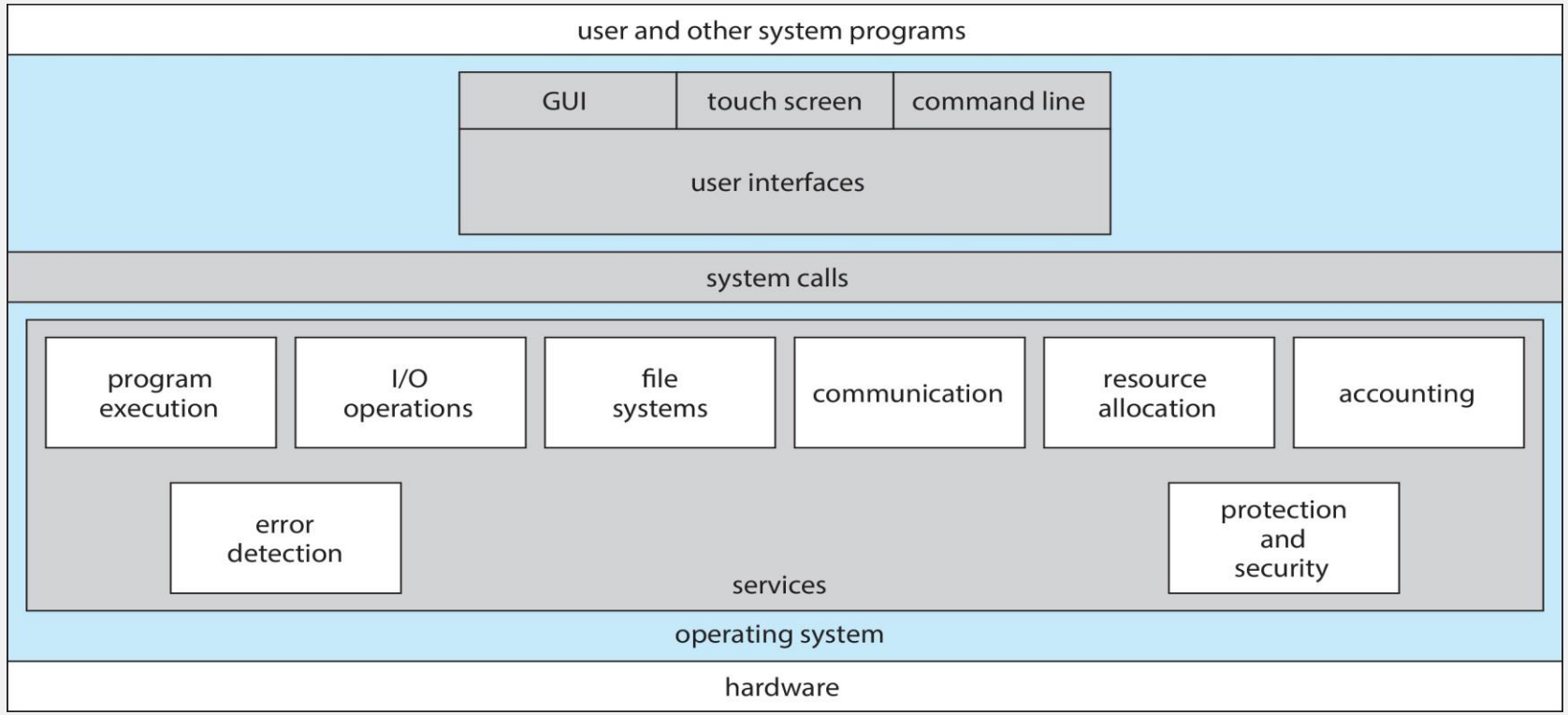
OPERATING SYSTEM SERVICES

- **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)

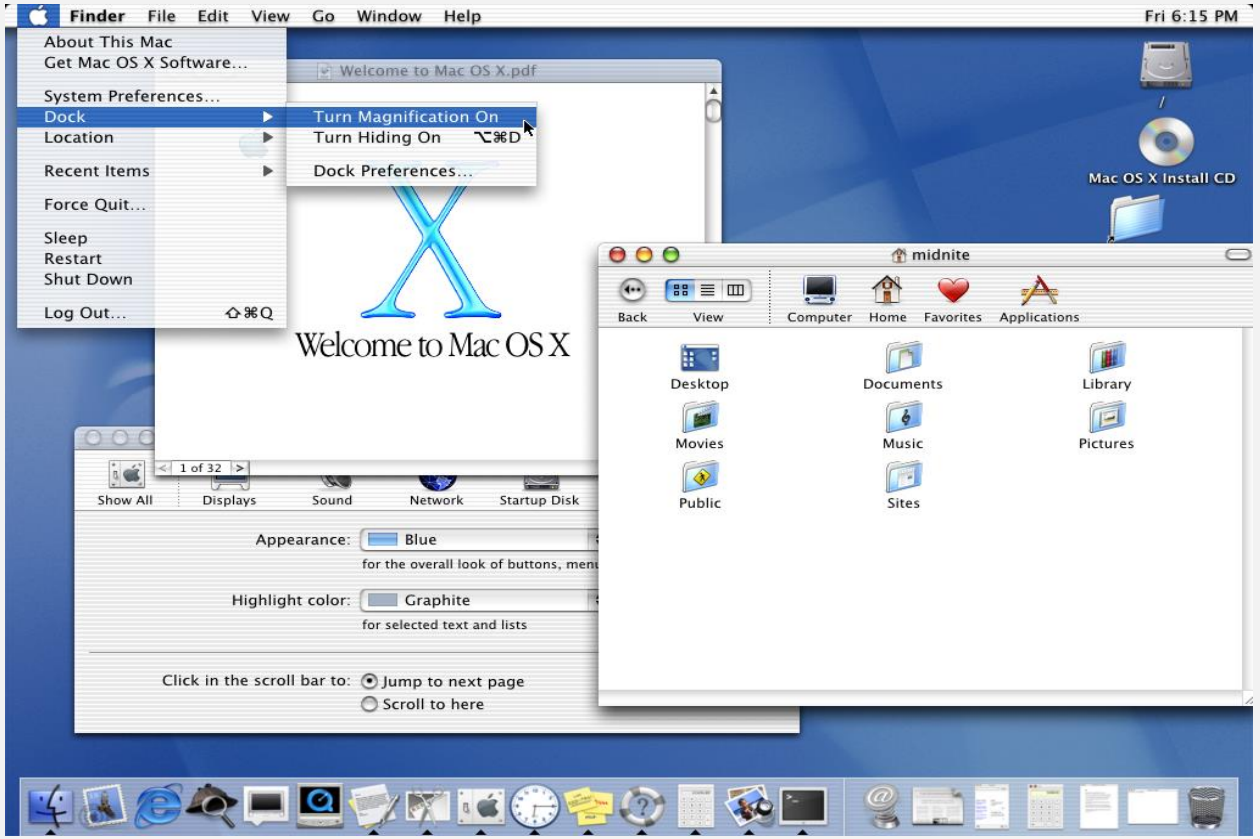
- User-friendly **desktop** metaphor interface
  - Usually mouse, keyboard, and monitor
  - **Icons** represent files, programs, actions, etc.
  - Various mouse buttons over objects in the interface cause various actions
    - provide information, options, execute function, open directory (known as a **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

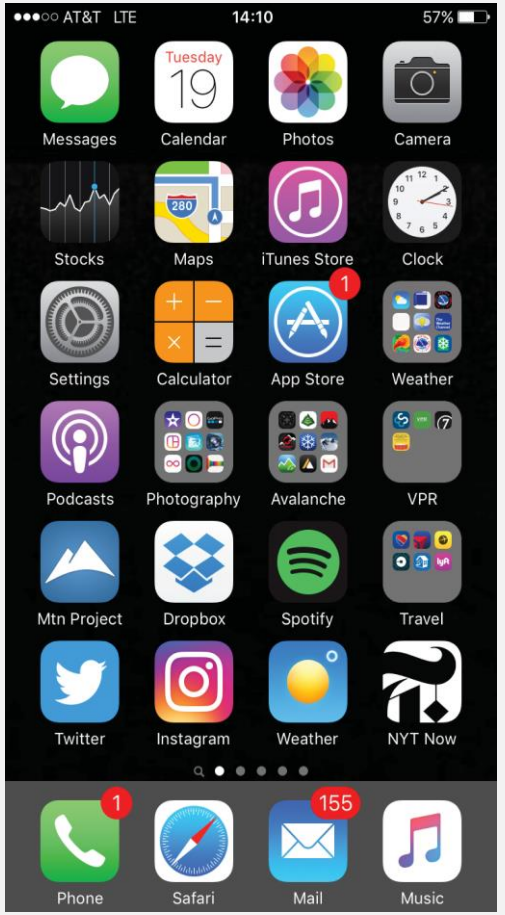
The macOS X GUI

- 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

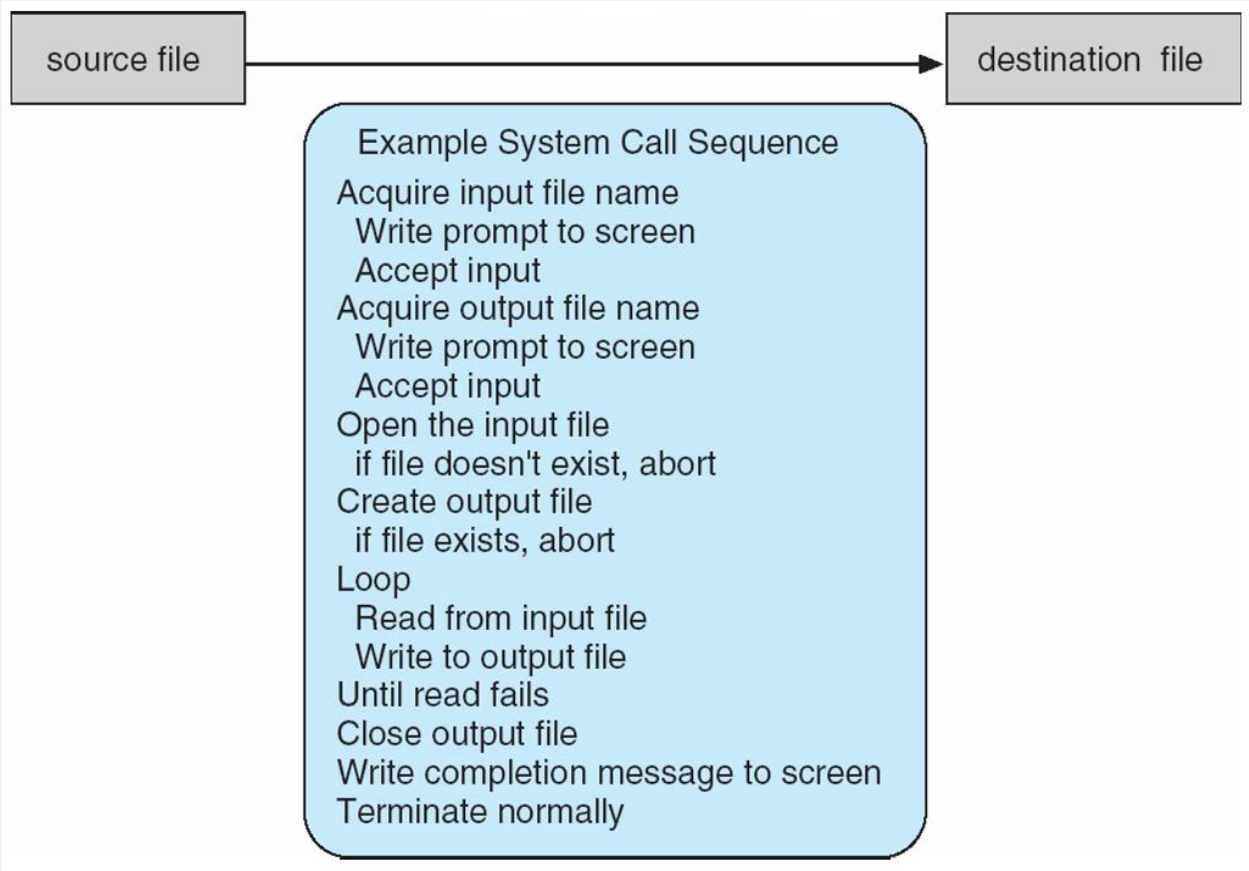
SYSTEM CALLS

- **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

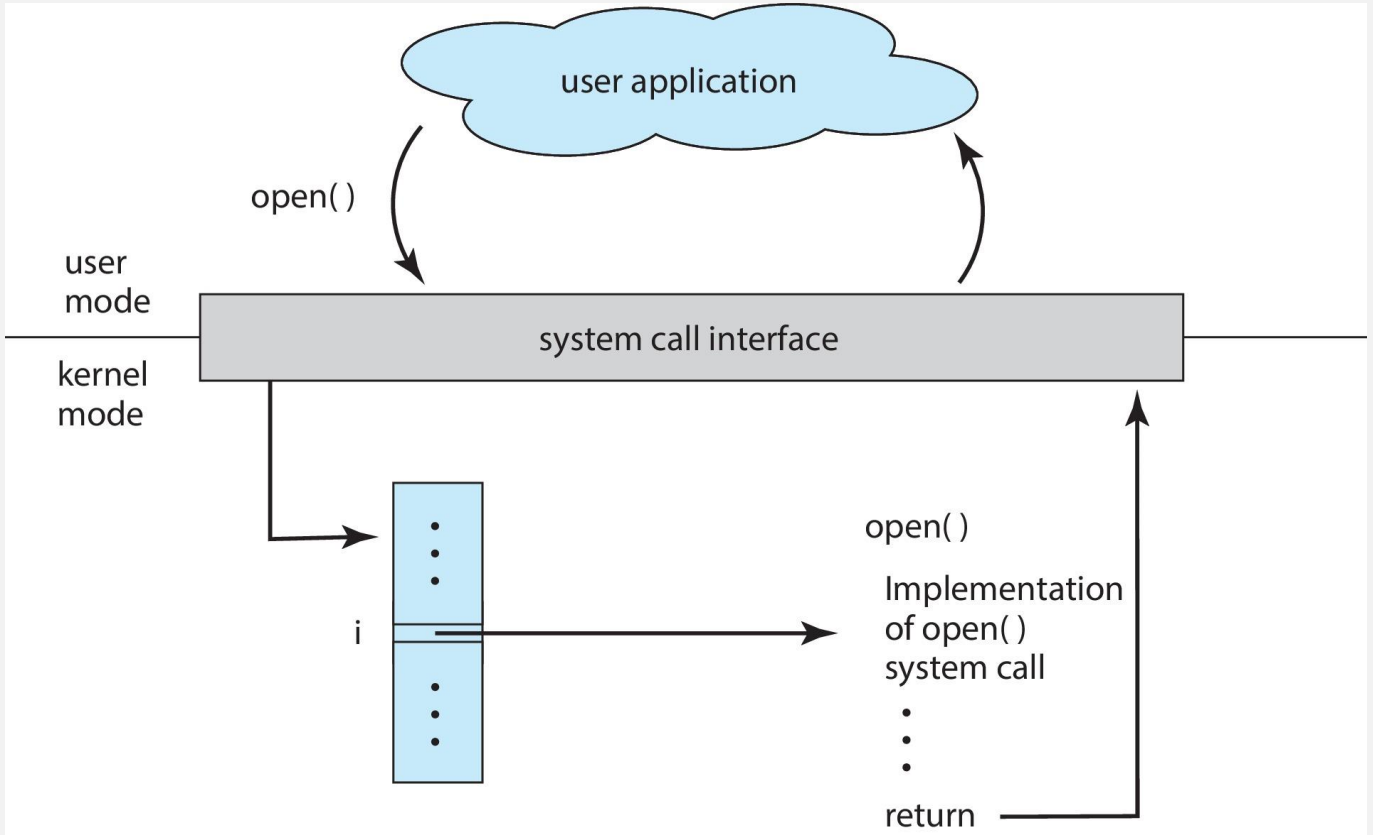
- Typically, a **number** is **associated with each system call**.
  - **System-call interface** maintains a table indexed according to these numbers.
- The **system call interface** invokes the intended system call in OS kernel and returns the status of the system call and any return values.
- The **caller 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

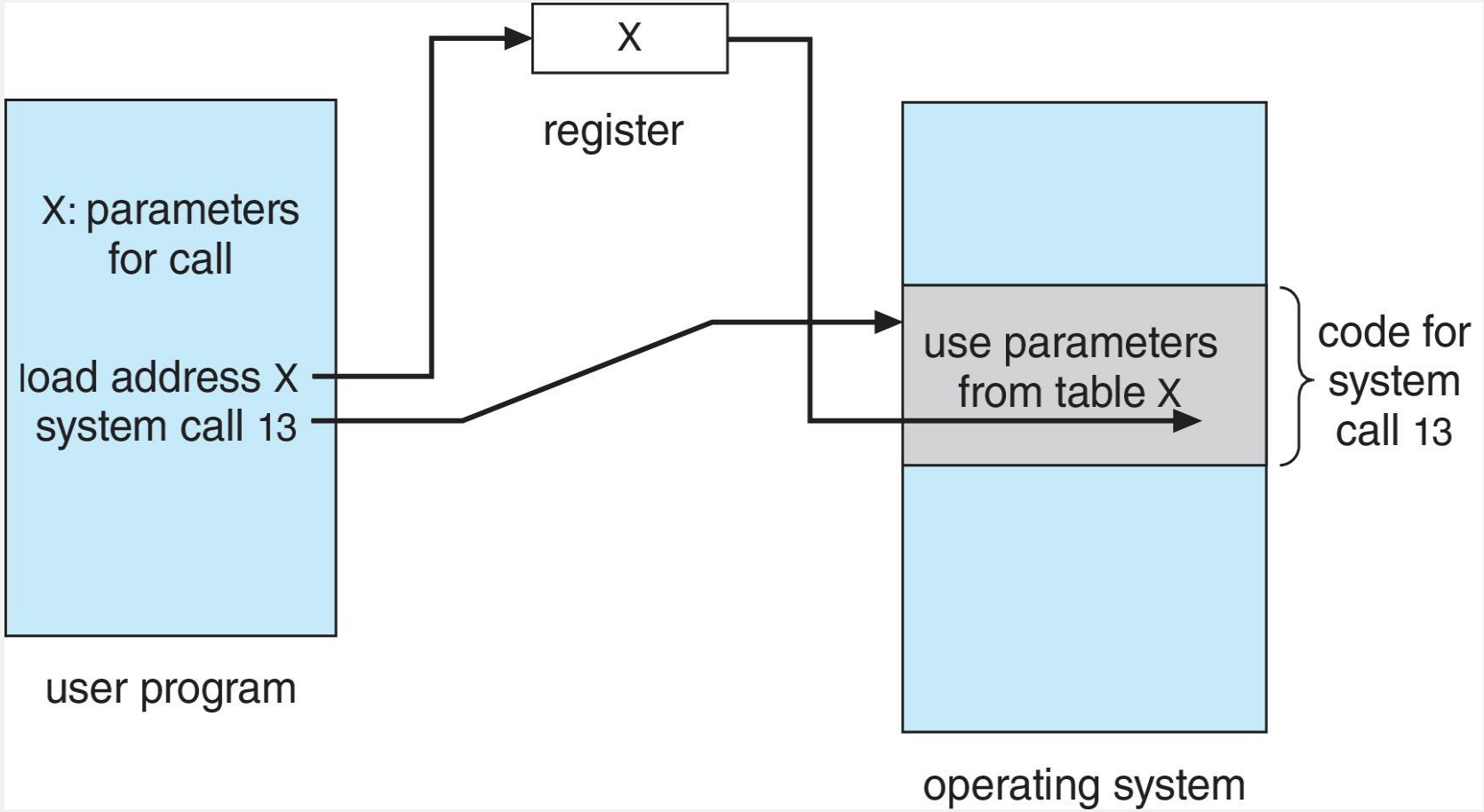
- More information is required than simply the identity of desired system call
  - Exact type and amount of information vary according to OS and call

- **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()

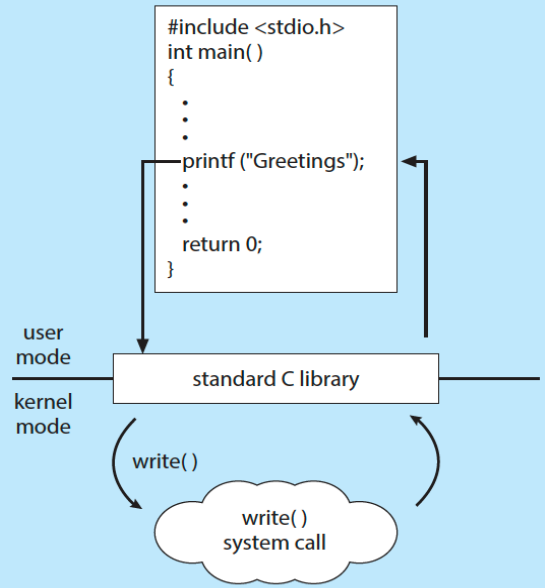
- Outline
- Operating System Services
- User and Operating-System Interface
- ✓System Calls
- System Services
- Linkers and Loaders

SYSTEM CALLS

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

THE STANDARD C LIBRARY

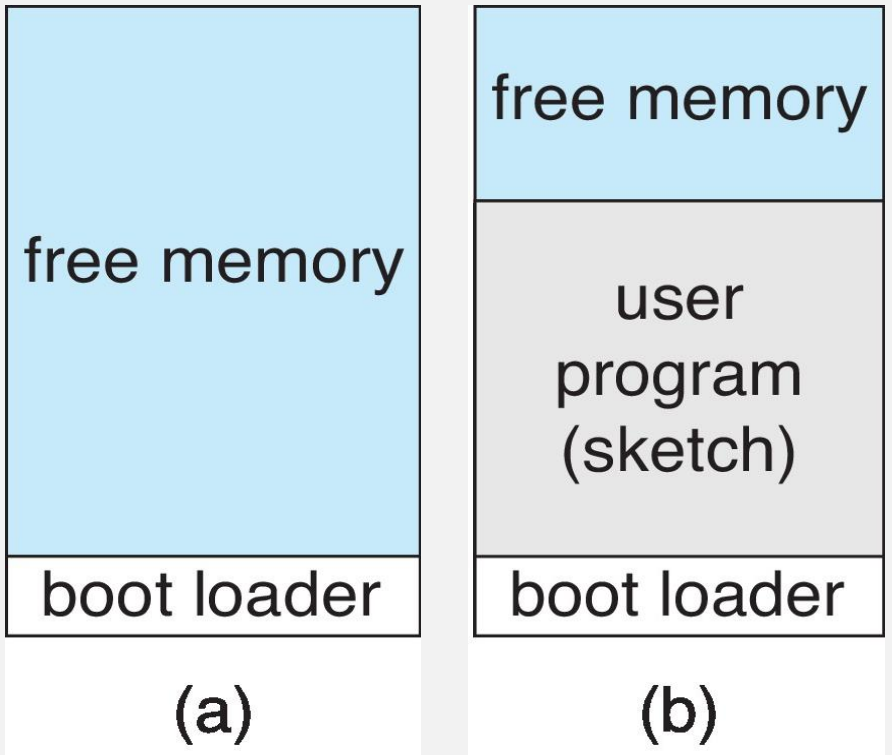
The standard C library provides a portion of the system-call interface for many versions of UNIX and Linux. As an example, let's assume a C program invokes the `printf()` statement. The C library intercepts this call and invokes the necessary system call (or calls) in the operating system—in this instance,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

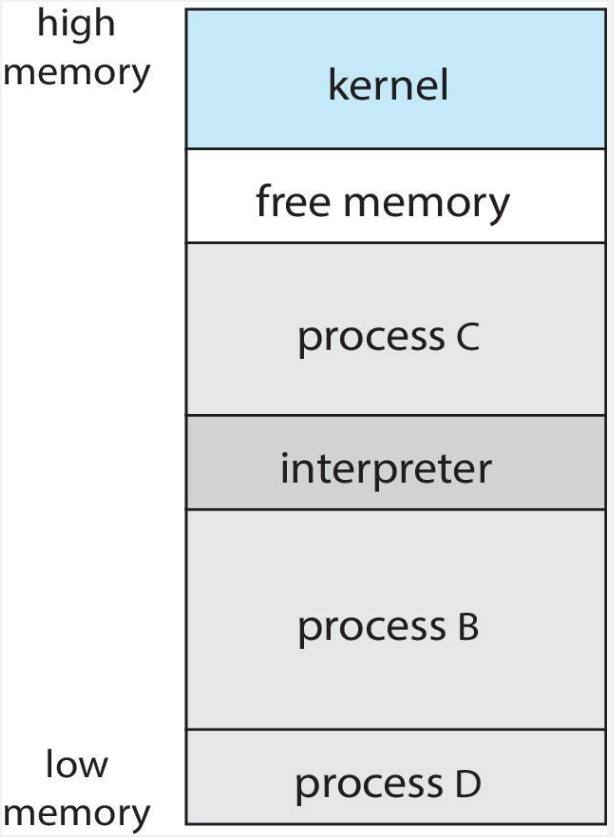


At system startup **Running a sketch**

- 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

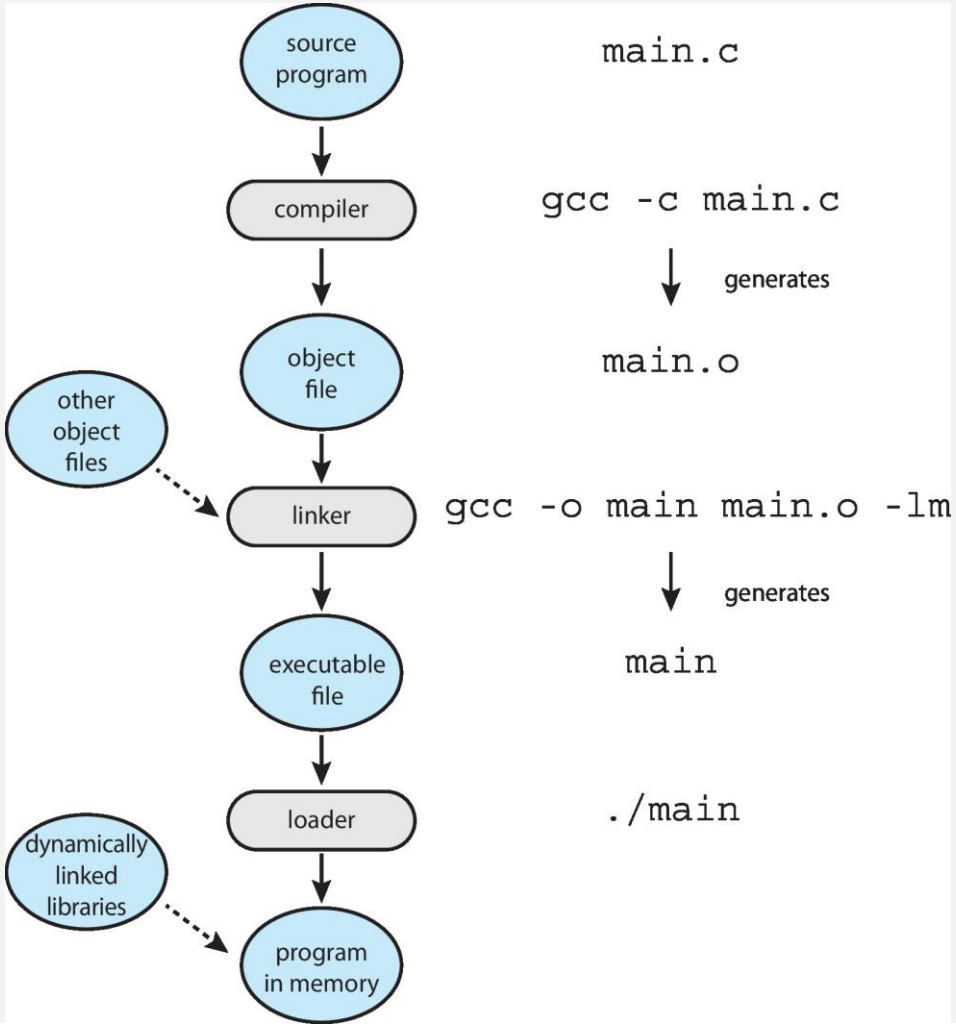
- Modern general purpose systems don't link libraries into executables
  - Rather, **dynamically linked libraries** (in Windows, **DLLs**) are loaded as needed, shared by all that use the same version of that same library (loaded once)

- **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!

UNIVERSITY of SAN CARLOS
SCIENTIA • VIRTUS • DEVOTIO

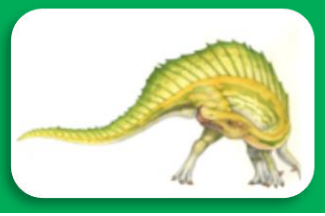
DCISM
DEPARTMENT OF COMPUTER, INFORMATION SCIENCES AND MATHEMATICS

2

CS 3104
OPERATING SYSTEMS

CHAPTER 2

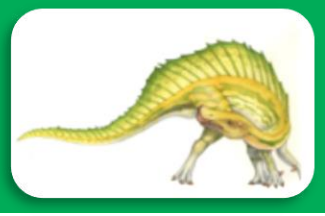
OPERATING-SYSTEM STRUCTURES



OUTLINE

- Why Applications are Operating-System Specific?
- Operating-System Design and Implementation
- Operating-System Structure
- Building and Booting an Operating System
- Operating-System Debugging

- Outline
- ✓Why Applications are Operating-System Specific?
- Operating-System Design and Implementation
- Operating-System Structure
- Building and Booting an Operating System
- Operating-System Debugging

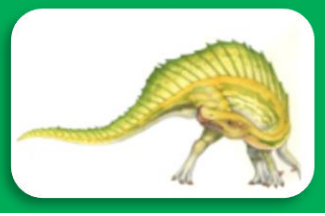


WHY APPLICATIONS ARE OPERATING-SYSTEM SPECIFIC?

- **Fundamental Fact:**
  - Apps compiled on one operating system are not usually executable on other operating systems

- **Problem:**
  - Each operating system provides its own unique system calls
    - Own file formats, etc.

- Outline
- ✓Why Applications are Operating-System Specific?
- Operating-System Design and Implementation
- Operating-System Structure
- Building and Booting an Operating System
- Operating-System Debugging

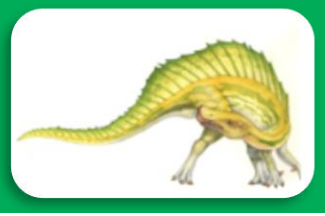


WHY APPLICATIONS ARE OPERATING-SYSTEM SPECIFIC?

- **Apps can be multi-operating system:**
  1. Apps written in interpreted language (like Python or Ruby) that has an interpreter available on multiple operating systems
  2. Apps written in language that includes a VM containing the running app (like Java)
  3. Application developer can use a standard language (like C) or API, compile separately on each operating system to run on each

- **Application Binary Interface (ABI):**
  - architecture equivalent of API (**API:** application level)
  - defines how different components of binary code can interface for a given operating system on a given architecture, CPU, etc.

- Outline
- ✓Why Applications are Operating-System Specific?
- Operating-System Design and Implementation
- Operating-System Structure
- Building and Booting an Operating System
- Operating-System Debugging



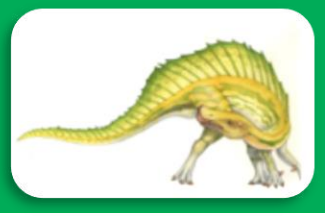
WHY APPLICATIONS ARE OPERATING-SYSTEM SPECIFIC?

Important Notes

- Unless an interpreter, RTE, or binary executable file is written for and compiled on a specific operating system on a specific CPU type (such as Intel x86 or ARMv8), the application will fail to run.

- Imagine the amount of work that is required for a program such as the Firefox browser to run on Windows, macOS, various Linux releases, iOS, and Android, sometimes on various CPU architectures.

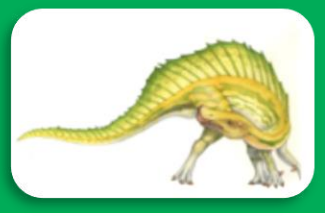
- Outline
- Why Applications are Operating-System Specific?
- ✓Operating-System Design and Implementation
- Operating-System Structure
- Building and Booting an Operating System
- Operating-System Debugging



OPERATING-SYSTEM DESIGN AND IMPLEMENTATION

- Problems in the design and implementation of OS that are not “solvable”, but some approaches have proven successful
- Internal structure of different Operating Systems can vary widely
- Start the design by defining goals and specifications
- Affected by choice of hardware and type of system
  - traditional desktop/laptop, mobile, distributed, or real time

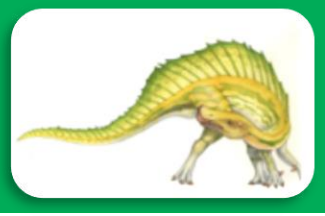
- Outline
- Why Applications are Operating-System Specific?
- ✓Operating-System Design and Implementation
- Operating-System Structure
- Building and Booting an Operating System
- Operating-System Debugging



OPERATING-SYSTEM DESIGN AND IMPLEMENTATION

- **User Goals:**
  - operating system should be convenient to use, easy to learn, reliable, safe, and fast
- **System Goals:**
  - operating system should be easy to design, implement, and maintain, as well as flexible, reliable, error-free, and efficient
- **Important Notes:**
  - requirements are vague and may be interpreted in various ways
  - no unique solution to the problem of defining the requirements for an operating system
  - wide range of systems in existence shows that different requirements can result in a large variety of solutions for different environments

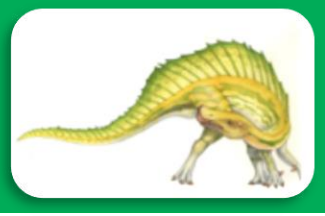
- Outline
- Why Applications are Operating-System Specific?
- ✓Operating-System Design and Implementation
- Operating-System Structure
- Building and Booting an Operating System
- Operating-System Debugging



OPERATING-SYSTEM DESIGN AND IMPLEMENTATION

- Important principle to separate:
  Policy: ***What*** will be done?
  Mechanism: ***How*** to do it?
- **Mechanisms** determine how to do something
- **Policies** decide what will be done
- The **separation of policy from mechanism** is a very important principle, it allows maximum flexibility if policy decisions are to be changed later (example: timer)
- **Specifying and designing an OS** is **highly creative task** of **software engineering**

- Outline
- Why Applications are Operating-System Specific?
- ✓Operating-System Design and Implementation
- Operating-System Structure
- Building and Booting an Operating System
- Operating-System Debugging



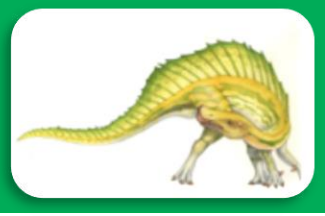
OPERATING-SYSTEM DESIGN AND IMPLEMENTATION

OS Implementation

- **Operating Systems** are **collections of many programs**
  - written by many people over a long period of time
  - difficult to make general statements on their implementation

- **Much variation:**
  - Early Operating Systems were written in Assembly Language
  - Then system programming languages like Algol, PL/1
  - Now in C, C++

- Outline
- Why Applications are Operating-System Specific?
- ✓Operating-System Design and Implementation
- Operating-System Structure
- Building and Booting an Operating System
- Operating-System Debugging

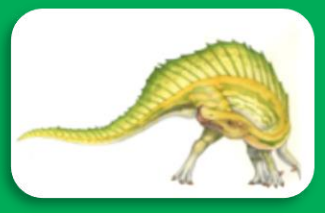


OPERATING-SYSTEM DESIGN AND IMPLEMENTATION

OS Implementation

- Actually, usually a **mix of languages**:
  - Lowest levels of the kernel in assembly
  - Main body in C
  - Systems programs in C, C++, scripting languages like PERL, Python, shell scripts
- **More high-level language** are easier to **port** to other hardware
  - but slower
- **Emulation** can allow an OS to run on non-native hardware

- Outline
- Why Applications are Operating-System Specific?
- ✓Operating-System Design and Implementation
- Operating-System Structure
- Building and Booting an Operating System
- Operating-System Debugging

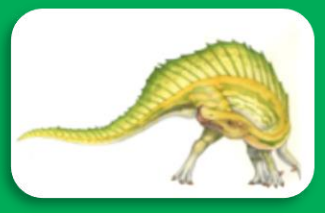


OPERATING-SYSTEM DESIGN AND IMPLEMENTATION

Important Notes

- Major performance improvements in operating systems are more likely to be the result of **better data structures and algorithms** than of **excellent assembly-language code**.
- In addition, although operating systems are large, **only a small amount of the code is critical to high performance**; the **interrupt handlers, I/O manager, memory manager, and CPU scheduler** are probably the most critical routines.
- After the system is written and is working correctly, bottlenecks can be identified and can be refactored to operate more efficiently

- Outline
- Why Applications are Operating-System Specific?
- Operating-System Design and Implementation
- ✓Operating-System Structure
- Building and Booting an Operating System
- Operating-System Debugging



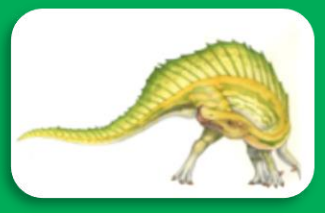
OPERATING-SYSTEM STRUCTURE

- General-purpose OS is a very large and complex program

- Various **ways to structure** Operating Systems:
  - **Simple structure:** MS-DOS
  - **More complex:** UNIX
  - **Layered:** An abstraction
  - **Microkernel:** Mach

Note: Not all versions of Mach are microkernels

- Outline
- Why Applications are Operating-System Specific?
- Operating-System Design and Implementation
- ✓Operating-System Structure
- Building and Booting an Operating System
- Operating-System Debugging

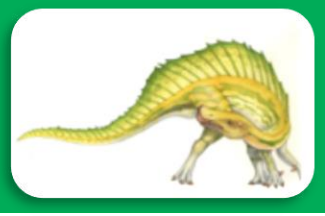


OPERATING-SYSTEM STRUCTURE

Monolithic Structure

- **Monolithic Structure**:
  - common technique for designing OS
  - place all of the functionality of the kernel into a single, static binary file that runs in a single address space
  - often known as a tightly coupled system because changes to one part of the system can have wide-ranging effects on other parts
  - **example: Original UNIX OS**

- Outline
- Why Applications are Operating-System Specific?
- Operating-System Design and Implementation
- ✓Operating-System Structure
- Building and Booting an Operating System
- Operating-System Debugging

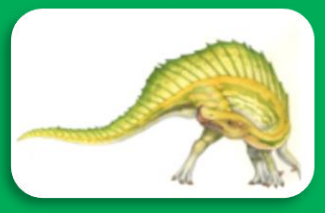


OPERATING-SYSTEM STRUCTURE

Monolithic Structure

- **UNIX OS**
  - limited by hardware functionality
    - the original UNIX operating system had limited structuring
  - **Consists of two separable parts:**
    - **Systems programs**
    - **The kernel**
      - ✓Consists of everything below the system-call interface and above the physical hardware
      - ✓Provides the file system, CPU scheduling, memory management, and other operating-system functions; a large number of functions for one level

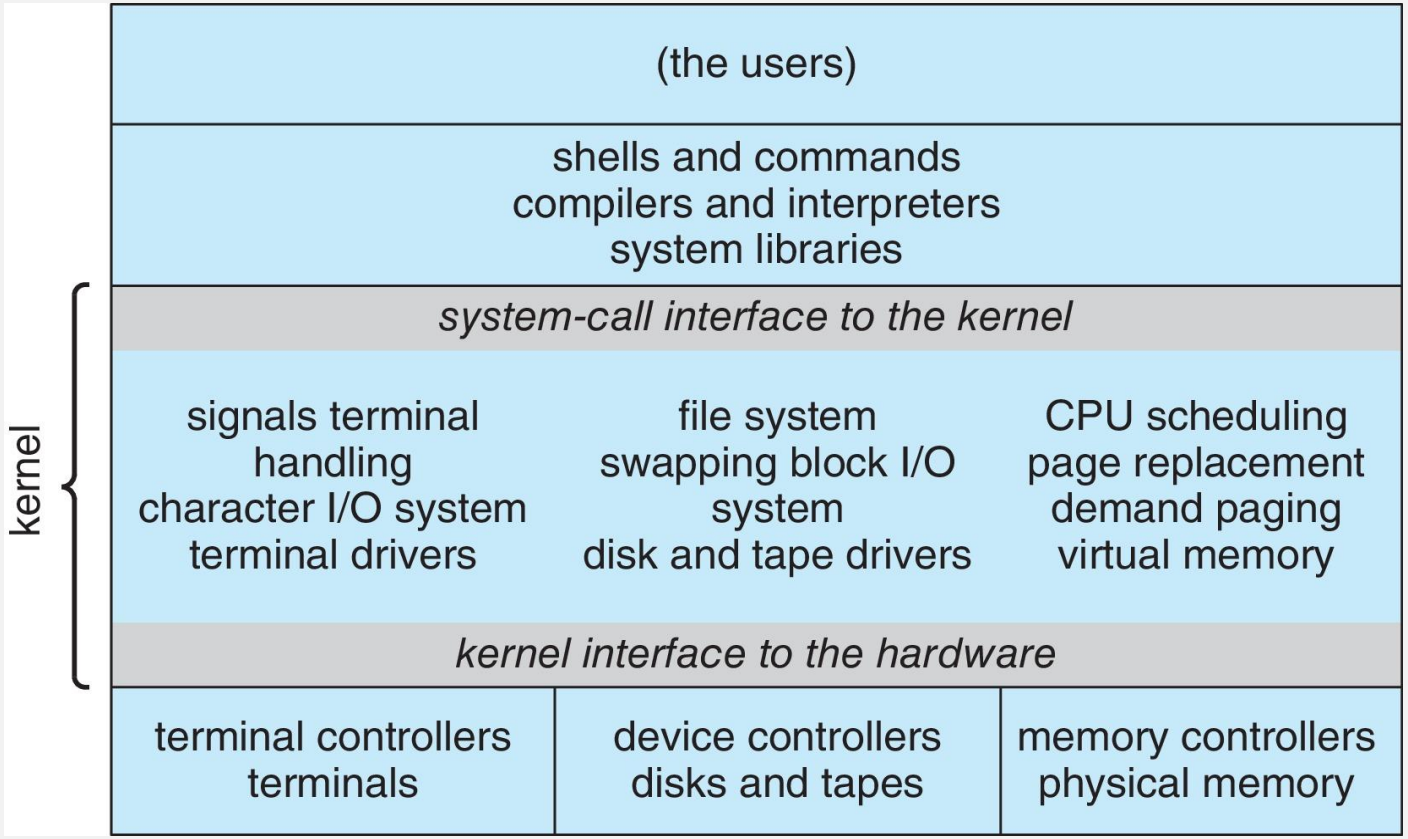
- Outline
- Why Applications are Operating-System Specific?
- Operating-System Design and Implementation
- ✓Operating-System Structure
- Building and Booting an Operating System
- Operating-System Debugging



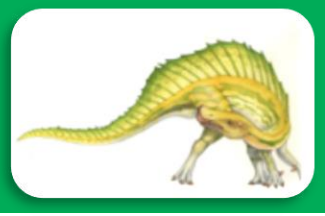
OPERATING-SYSTEM STRUCTURE

Traditional UNIX System Structure

Beyond simple but not fully layered



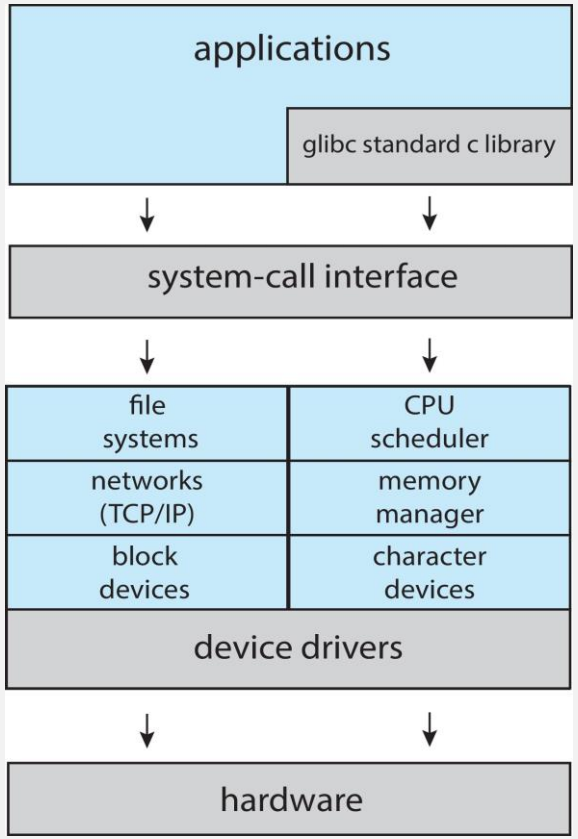
- Outline
- Why Applications are Operating-System Specific?
- Operating-System Design and Implementation
- ✓Operating-System Structure
- Building and Booting an Operating System
- Operating-System Debugging



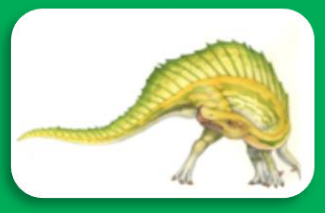
OPERATING-SYSTEM STRUCTURE

Linux System Structure

Monolithic plus modular design



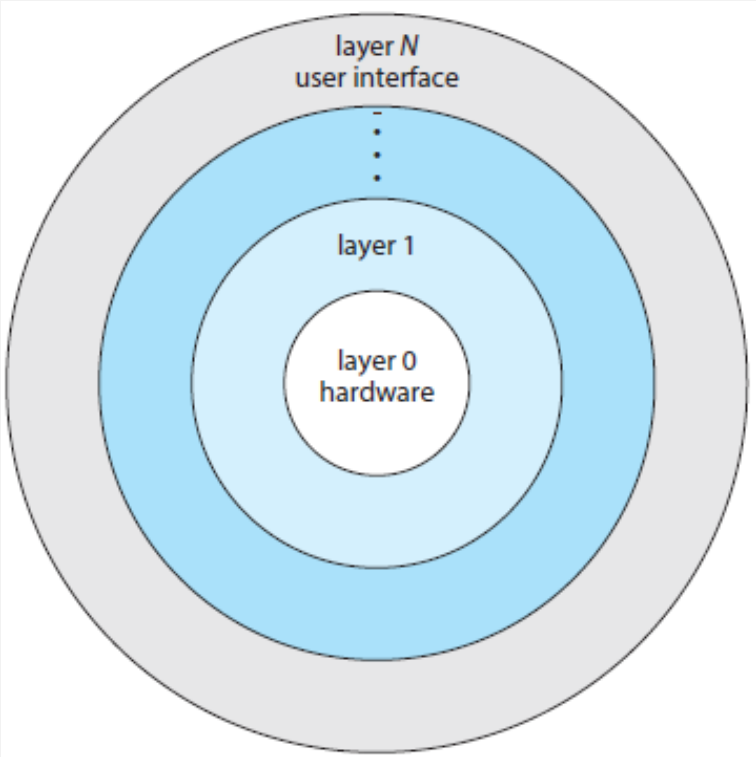
- Outline
- Why Applications are Operating-System Specific?
- Operating-System Design and Implementation
- ✓Operating-System Structure
- Building and Booting an Operating System
- Operating-System Debugging



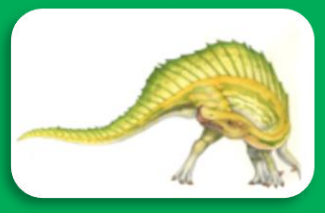
OPERATING-SYSTEM STRUCTURE

Layered Approach

- The operating system is divided into a number of layers (levels), each is built on top of lower layers.
- The **bottom layer** (**layer 0**), is **the hardware**; the **highest** (layer N) is **the user interface**.
- With modularity (**modular approach**), layers are selected such that **each** uses **functions** (operations) and **services** of only lower-level layers



- Outline
- Why Applications are Operating-System Specific?
- Operating-System Design and Implementation
- ✓Operating-System Structure
- Building and Booting an Operating System
- Operating-System Debugging

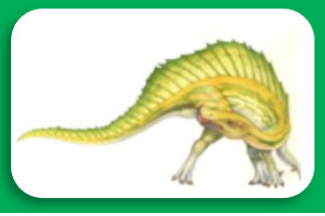


OPERATING-SYSTEM STRUCTURE

Microkernels

- A **piece of software or even code** that contains the **near-minimum amount of functions and features** required to **implement an operating system**.
- **Moves** as much **from the kernel** into **user space**
- **Mach:** example of **microkernel**
  - Mac OS X kernel (**Darwin**) is partly based on Mach
- **Communication takes place between user modules** using **message passing**

- Outline
- Why Applications are Operating-System Specific?
- Operating-System Design and Implementation
- ✓Operating-System Structure
- Building and Booting an Operating System
- Operating-System Debugging



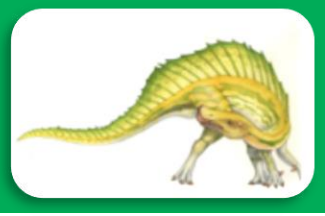
OPERATING-SYSTEM STRUCTURE

Microkernels

- **Benefits:**
  - Easier to extend a microkernel
  - Easier to port the operating system to new architectures
  - More reliable (less code is running in kernel mode)
  - More secure

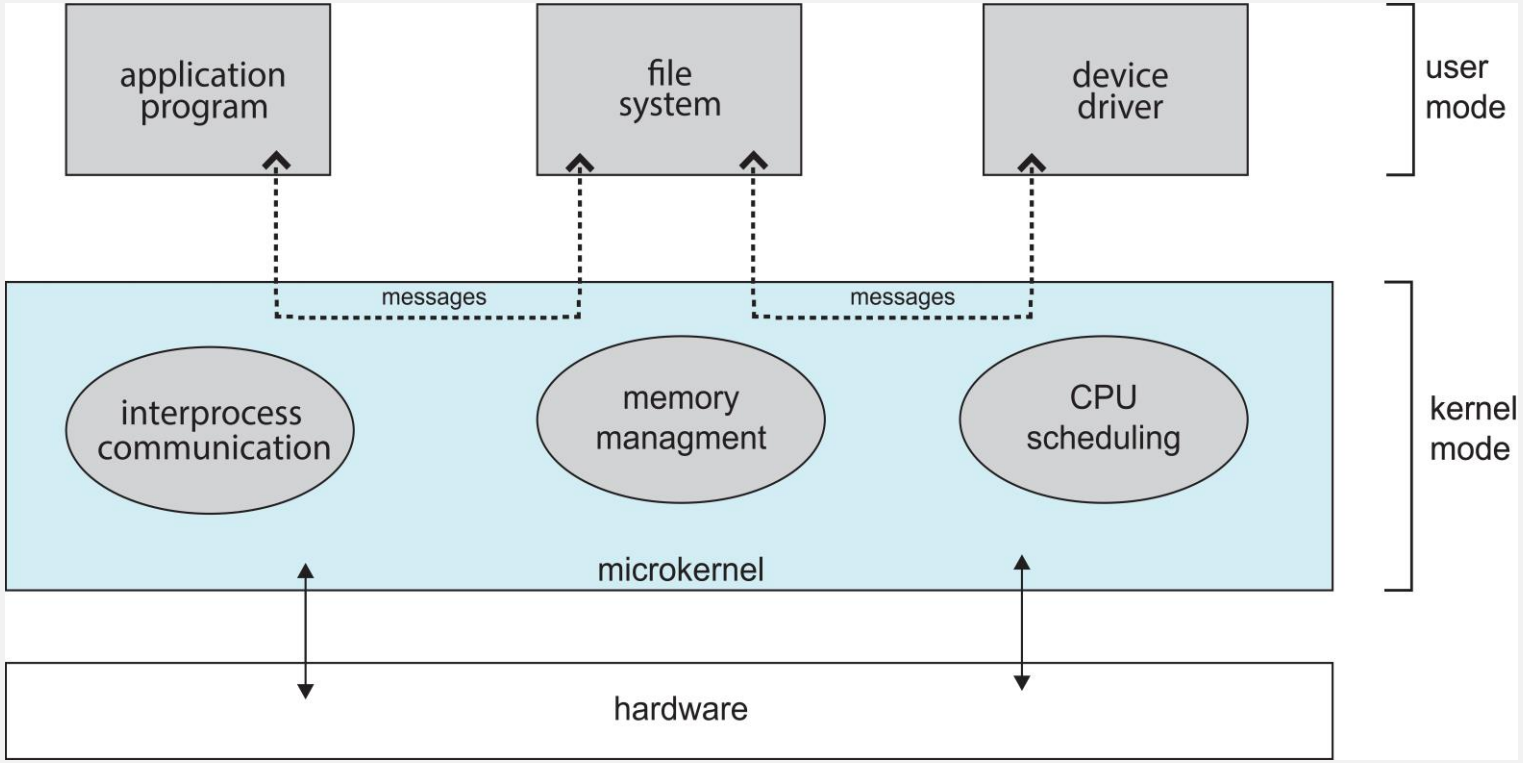
- **Detriments:**
  - Performance **overhead of user space to kernel space** communication

- Outline
- Why Applications are Operating-System Specific?
- Operating-System Design and Implementation
- ✓Operating-System Structure
- Building and Booting an Operating System
- Operating-System Debugging

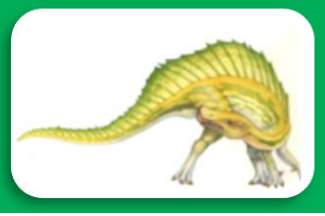


OPERATING-SYSTEM STRUCTURE

Microkernel System Structure



- Outline
- Why Applications are Operating-System Specific?
- Operating-System Design and Implementation
- ✓Operating-System Structure
- Building and Booting an Operating System
- Operating-System Debugging



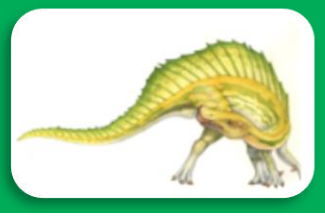
OPERATING-SYSTEM STRUCTURE

Modules

- Many modern operating systems implement **Loadable Kernel Modules (LKMs)**
  - Uses object-oriented approach
  - Each core component is separate
  - Each talks to the others over known interfaces
  - Each is loadable as needed within the kernel

- Overall, **similar to layered approach** but **more flexible** because **any module can call any other module**
  - Linux, Solaris, etc.

- Outline
- Why Applications are Operating-System Specific?
- Operating-System Design and Implementation
- ✓Operating-System Structure
- Building and Booting an Operating System
- Operating-System Debugging



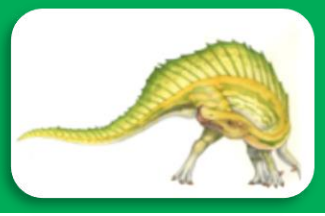
OPERATING-SYSTEM STRUCTURE

Hybrid Systems

- Most modern operating systems are actually not one pure model
  - Hybrid combines multiple approaches to address performance, security, usability needs
  - Linux and Solaris kernels in kernel address space, so monolithic, plus modular for dynamic loading of functionality
  - Windows mostly monolithic, plus microkernel for different subsystem ***personalities***

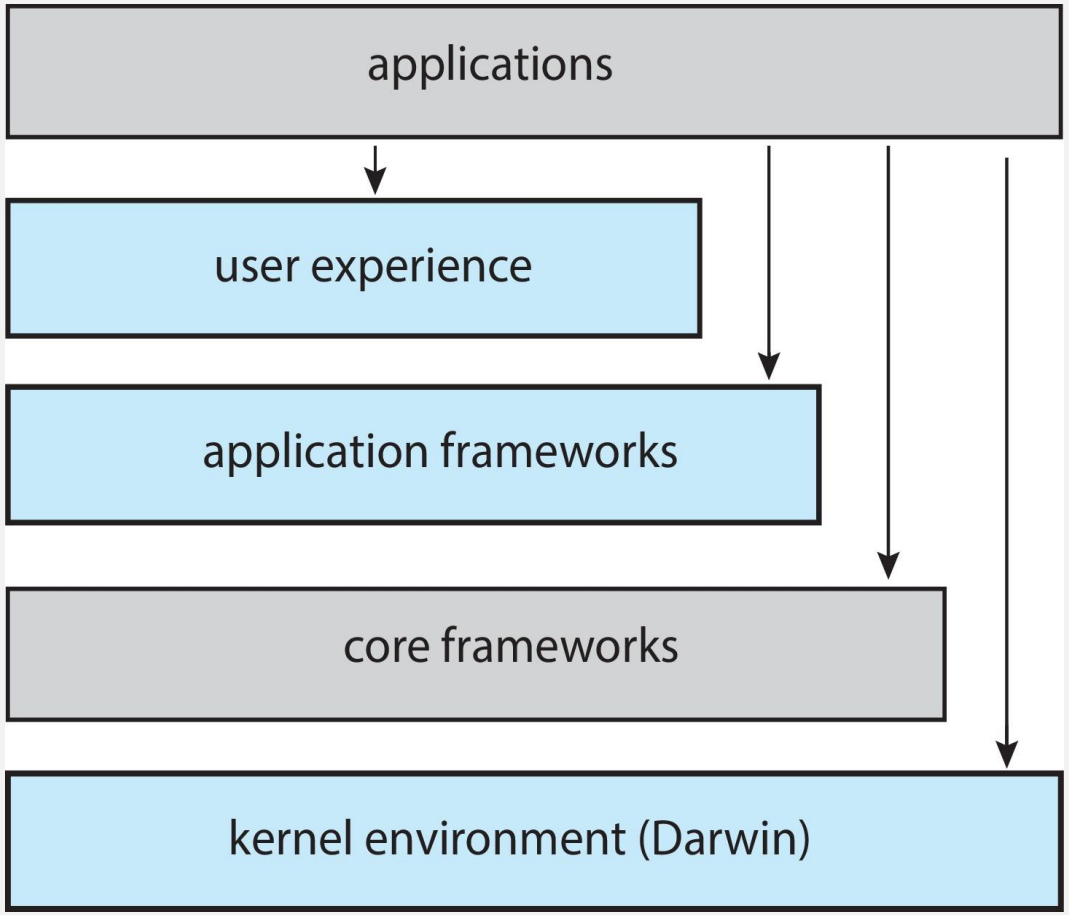
- Apple Mac OS X hybrid, layered, **Aqua** UI plus **Cocoa** programming environment
  - Below is kernel consisting of Mach microkernel and BSD Unix parts, plus I/O kit and dynamically loadable modules (called **kernel extensions**)

- Outline
- Why Applications are Operating-System Specific?
- Operating-System Design and Implementation
- ✓Operating-System Structure
- Building and Booting an Operating System
- Operating-System Debugging

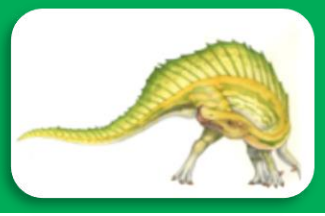


OPERATING-SYSTEM STRUCTURE

macOS and iOS Structure

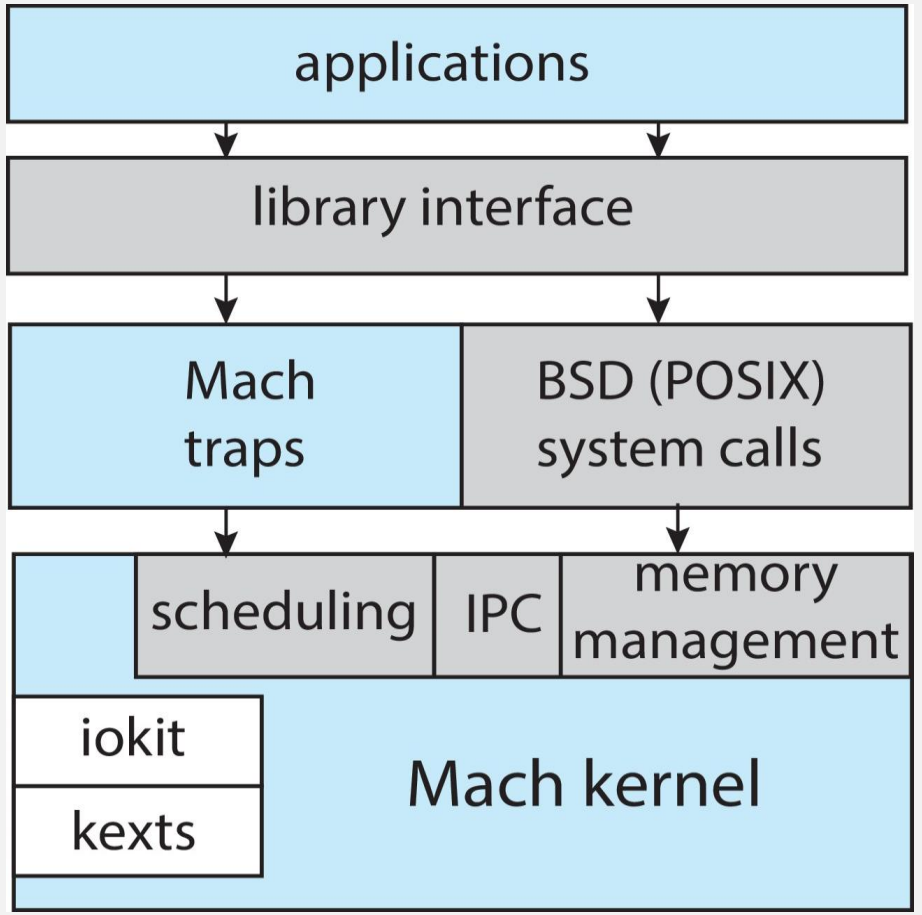


- Outline
- Why Applications are Operating-System Specific?
- Operating-System Design and Implementation
- ✓Operating-System Structure
- Building and Booting an Operating System
- Operating-System Debugging

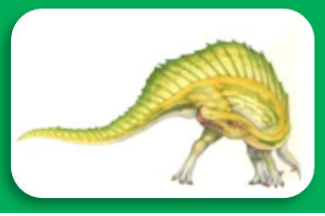


OPERATING-SYSTEM STRUCTURE

Darwin



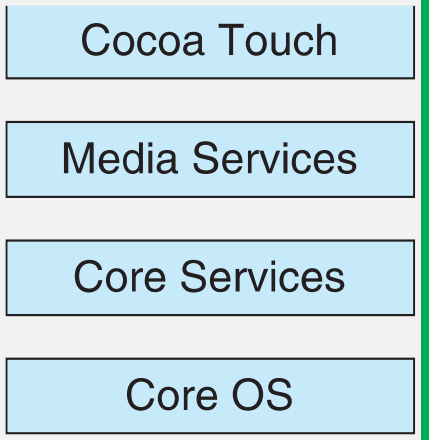
- Outline
- Why Applications are Operating-System Specific?
- Operating-System Design and Implementation
- ✓Operating-System Structure
- Building and Booting an Operating System
- Operating-System Debugging



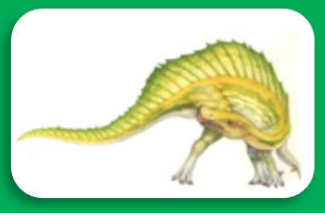
OPERATING-SYSTEM STRUCTURE

iOS

- Apple mobile OS for ***iPhone***, ***iPad***
  - Structured on Mac OS X, added functionality
  - Does not run OS X applications natively
    - Also runs on different CPU architecture (ARM vs. Intel)
  - **Cocoa Touch** Objective-C API for developing apps
  - **Media services** layer for graphics, audio, video
  - **Core services** provides cloud computing, databases
  - **Core O**perating **S**ystem, based on Mac OS X kernel



- Outline
- Why Applications are Operating-System Specific?
- Operating-System Design and Implementation
- ✓Operating-System Structure
- Building and Booting an Operating System
- Operating-System Debugging

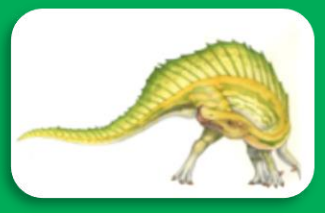


OPERATING-SYSTEM STRUCTURE

Android

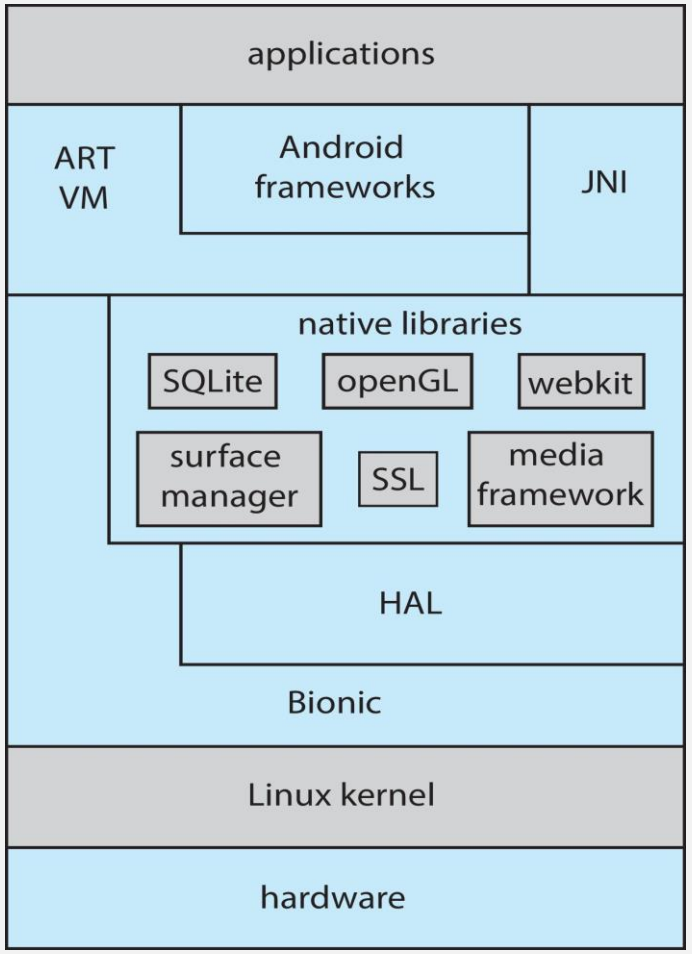
- Developed by Open Handset Alliance (mostly Google)
  - Open Source
- Similar stack to IOS
- Based on Linux kernel but modified
  - Provides process, memory, device-driver management
  - Adds power management
- **Runtime environment** includes core set of libraries and Dalvik virtual machine
  - Apps developed in Java plus Android API
    - ✓ Java class files compiled to Java bytecode then translated to executable than runs in Dalvik VM
- Libraries include frameworks for web browser (webkit), database (SQLite), multimedia, smaller libc

- Outline
- Why Applications are Operating-System Specific?
- Operating-System Design and Implementation
- ✓Operating-System Structure
- Building and Booting an Operating System
- Operating-System Debugging

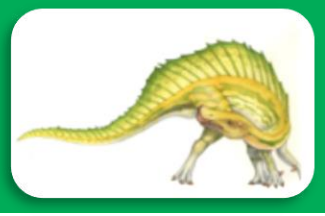


OPERATING-SYSTEM STRUCTURE

Android Architecture



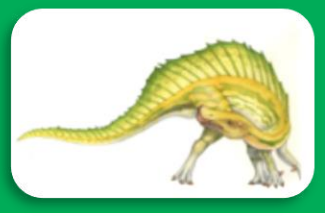
- Outline
- Why Applications are Operating-System Specific?
- Operating-System Design and Implementation
- Operating-System Structure
- ✓Building and Booting an Operating System
- Operating-System Debugging



BUILDING AND BOOTING AN OPERATING SYSTEM

- Operating Systems are generally designed to run on a class of systems with variety of peripherals
- Commonly, operating system is already installed on a purchased computer
  - But one can build and install some other operating systems
- If generating an operating system from scratch:
  - Write the operating system source code
  - Configure the OS for the system on which it will run
  - Compile the operating system
  - Install the operating system
  - Boot the computer and its new operating system

- Outline
- Why Applications are Operating-System Specific?
- Operating-System Design and Implementation
- Operating-System Structure
- ✓Building and Booting an Operating System
- Operating-System Debugging

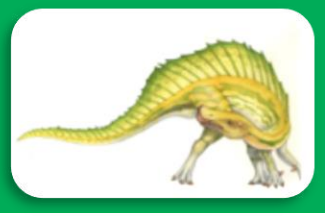


BUILDING AND BOOTING AN OPERATING SYSTEM

Building and Booting Linux

- Download Linux source code (https://www.kernel.org)
- Configure kernel via "`make menuconfig`"
- Compile the kernel using "`make`"
  - Produces `vmlinuz`, the kernel image
  - Compile kernel modules via "`make modules`"
  - Install kernel modules into `vmlinuz` via "`make modules_install`"
  - Install new kernel on the system via "`make install`"

- Outline
- Why Applications are Operating-System Specific?
- Operating-System Design and Implementation
- Operating-System Structure
- ✓Building and Booting an Operating System
- Operating-System Debugging

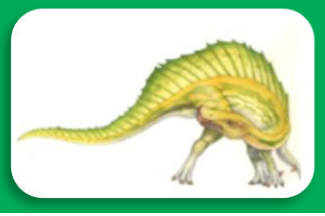


BUILDING AND BOOTING AN OPERATING SYSTEM

System Boot

- When power is initialized on system, execution starts at a fixed memory location
- Operating system must be made available to hardware so hardware can start it
  - Small piece of code: **bootstrap loader**, **BIOS**, stored in **ROM** or **EEPROM** locates the kernel, loads it into memory, and starts it
  - Sometimes a two-step process where **boot block** at fixed location loaded by **ROM** code, **which loads bootstrap loader from disk**
  - Modern systems replace BIOS with **Unified Extensible Firmware Interface (UEFI)**
- Common bootstrap loader, **GRUB**, allows selection of kernel from multiple disks, versions, kernel options
- Kernel loads and system is then **running**
- Boot loaders frequently allow various boot states, such as single user mode

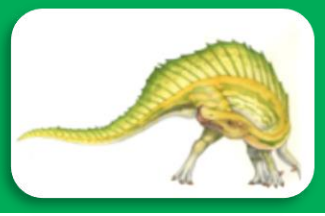
- Outline
- Why Applications are Operating-System Specific?
- Operating-System Design and Implementation
- Operating-System Structure
- Building and Booting an Operating System
- ✓Operating-System Debugging



OPERATING-SYSTEM DEBUGGING

- **Debugging** is finding and fixing **errors**, or **bugs**
- Can also include **performance tuning**
- OS generate **log files** containing error information
- Failure of an application can generate **core dump** file capturing memory of the process
- Operating system failure can generate **crash dump** file containing kernel memory

- Outline
- Why Applications are Operating-System Specific?
- Operating-System Design and Implementation
- Operating-System Structure
- Building and Booting an Operating System
- ✓Operating-System Debugging

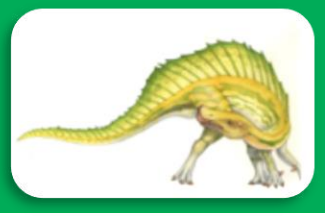


OPERATING-SYSTEM DEBUGGING

- Beyond crashes, **performance tuning** can optimize system performance
  - Sometimes using ***trace listings*** of activities, recorded for analysis
  - **Profiling** is the periodic sampling of instruction pointer to look for statistical trends

- **Kernighan's Law:**
  - "Debugging is twice as hard as writing the code in the first place.
  - Therefore, if you write the code as cleverly as possible, you are, by definition, not smart enough to debug it."

- Outline
- Why Applications are Operating-System Specific?
- Operating-System Design and Implementation
- Operating-System Structure
- Building and Booting an Operating System
- ✓Operating-System Debugging

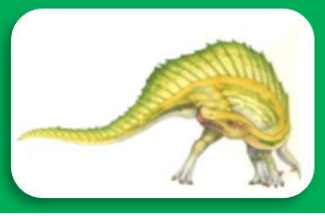


OPERATING-SYSTEM DEBUGGING

Performance Tuning

- Improve performance by removing bottlenecks
- OS must provide means of computing and displaying measures of system behavior
- **Example:** "top" program or Windows Task Manager

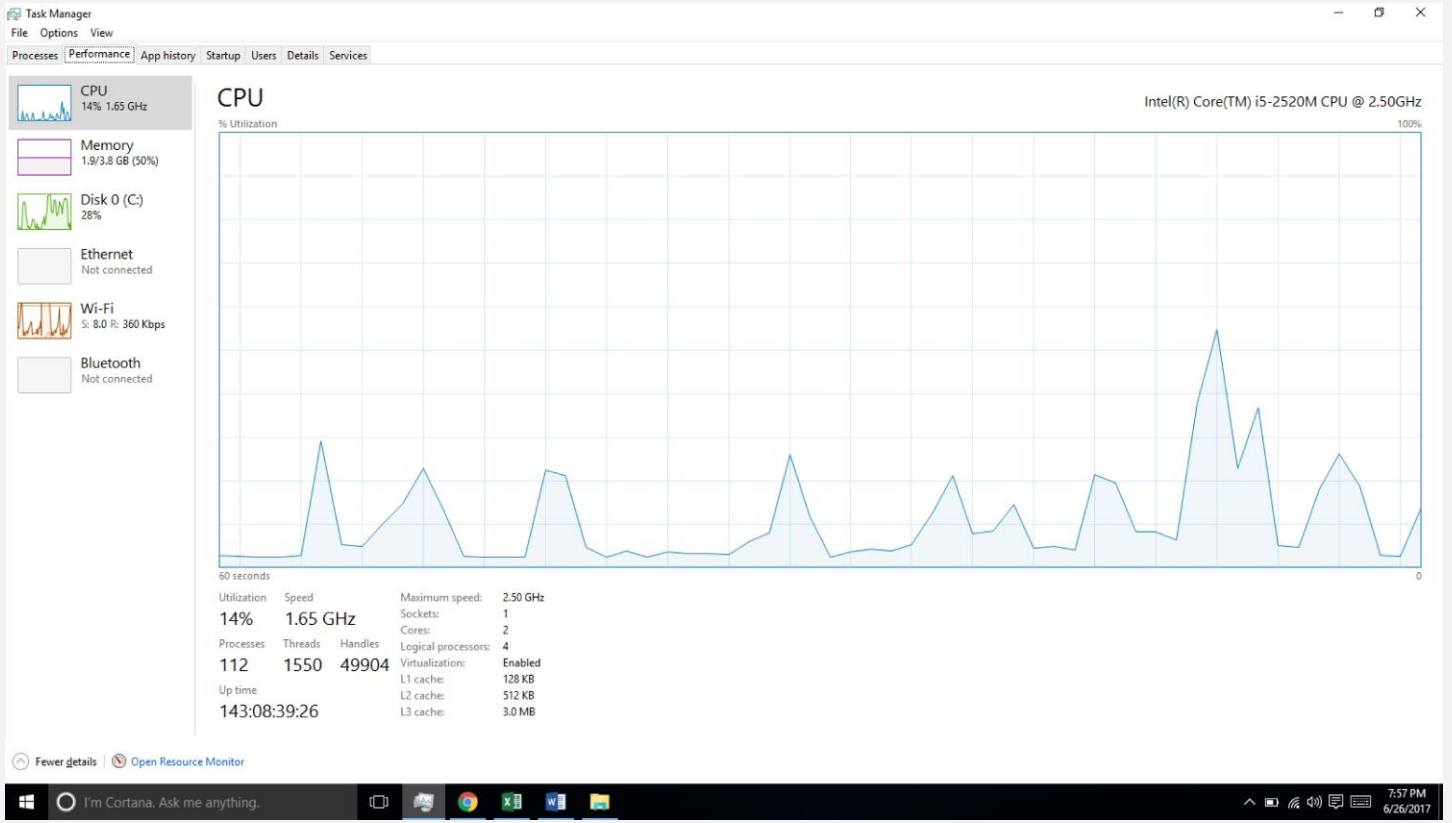
- Outline
- Why Applications are Operating-System Specific?
- Operating-System Design and Implementation
- Operating-System Structure
- Building and Booting an Operating System
- ✓Operating-System Debugging



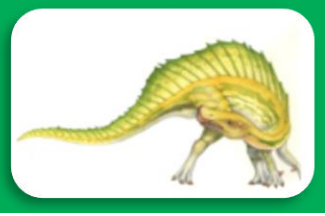
OPERATING-SYSTEM DEBUGGING

Performance Tuning

The Windows 10 Task Manager



- Outline
- Why Applications are Operating-System Specific?
- Operating-System Design and Implementation
- Operating-System Structure
- Building and Booting an Operating System
- ✓Operating-System Debugging



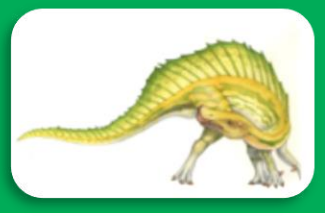
OPERATING-SYSTEM DEBUGGING

Counters

- Operating systems **keep track of system activity** through a series of **counters**
  - the number of system calls made, or
  - the number of operations performed to a network device or disk

- **Counter-based tools** are tools that simply **inquire on the current value of certain statistics that are maintained by the kernel**

- Outline
- Why Applications are Operating-System Specific?
- Operating-System Design and Implementation
- Operating-System Structure
- Building and Booting an Operating System
- ✓Operating-System Debugging

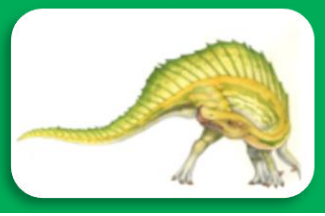


OPERATING-SYSTEM DEBUGGING

Counters

- **Tools include:**
  - **ps** — reports information for a single process or selection of processes
  - **top** — reports real-time statistics for current processes
  - **vmstat** — reports memory-usage statistics
  - **netstat** — reports statistics for network interfaces
  - **iostat** — reports I/O usage for disks

- Outline
- Why Applications are Operating-System Specific?
- Operating-System Design and Implementation
- Operating-System Structure
- Building and Booting an Operating System
- ✓Operating-System Debugging



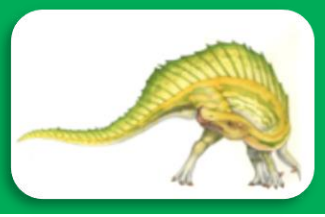
OPERATING-SYSTEM DEBUGGING

Tracing

- **Collects data for a specific event**, such as **steps** involved in a **system call invocation**

- **Tools include:**
  - **strace** – trace system calls invoked by a process
  - **gdb** – source-level debugger
  - **perf** – collection of Linux performance tools
  - **tcpdump** – collects network packets

- Outline
- Why Applications are Operating-System Specific?
- Operating-System Design and Implementation
- Operating-System Structure
- Building and Booting an Operating System
- ✓Operating-System Debugging



OPERATING-SYSTEM DEBUGGING

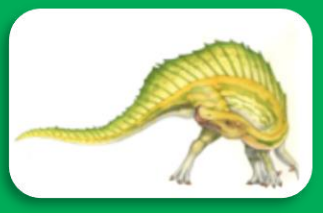
BCC

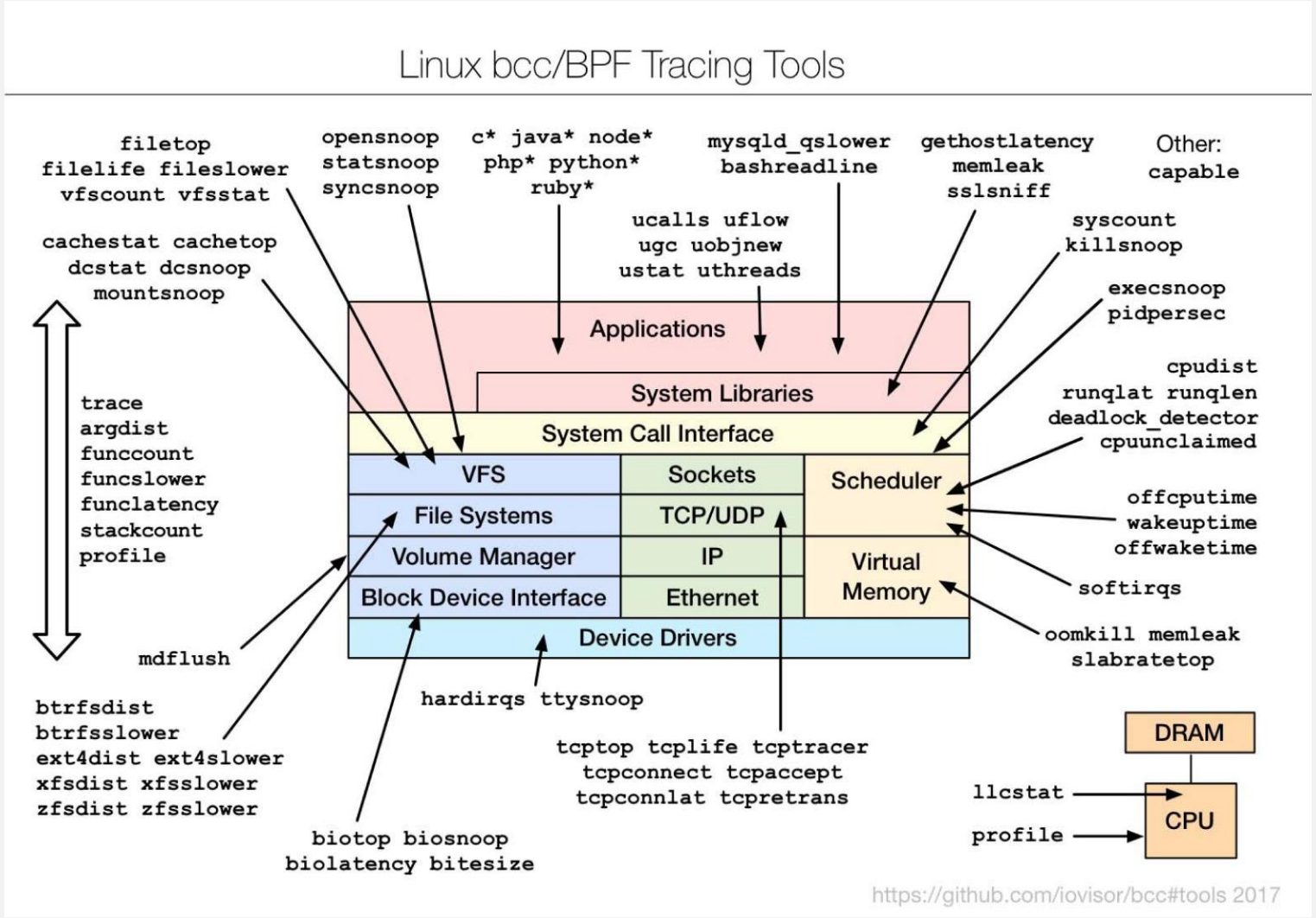
- **Debugging interactions** between **user-level and kernel code** nearly impossible without **toolset** that understands both and an instrument their actions
- **BCC (BPF Compiler Collection)** is a **rich toolkit** providing **tracing features for Linux**
- **Example: disksnoop.py** traces disk I/O activity

```
TIME(s)          T    BYTES    LAT(ms)
1946.29186700    R    8            0.27
1946.33965000    R    8            0.26
1948.34585000    W    8192         0.96
1950.43251000    R    4096         0.56
1951.74121000    R    4096         0.35
```

OPERATING-SYSTEM DEBUGGING

- Outline
- Why Applications are Operating-System Specific?
- Operating-System Design and Implementation
- Operating-System Structure
- Building and Booting an Operating System
- ✓Operating-System Debugging





UNIVERSITY of SAN CARLOS
SCIENTIA • VIRTUS • DEVOTIO

DCISM
DEPARTMENT OF COMPUTER, INFORMATION SCIENCES AND MATHEMATICS

CS 3104
OPERATING SYSTEMS

2

*** END ***

THANK YOU!

UNIVERSITY *of* SAN CARLOS

SCIENTIA • VIRTUS • DEVOTIO

USC FLEX

Flexible Learning Plan Primer

A.Y. 2022-2023

CONTENTS

Message from the President

I

Dear Carolinian Students,

Blessings of peace and good health to all of you!

As I welcome you (back) to the University, I also welcome your aspirations and the future that you represent. It is a future that holds great promise for those who keep their nose to the grindstone and stay committed to their dreams amidst pitfalls and setbacks. The educational experience you will acquire this academic year will be based on how you apply the lessons you've learned from the past and on how hard you will try to overcome the challenges of the present.

The pandemic's disruptive impact continues and is undeniable. It has altered our best laid plans and exposed our hidden vulnerabilities. But a new academic year is always a chance to make a fresh start and come out stronger. Recall therefore where you faltered and where you did well. Try to be wiser this time and be on the lookout out for the things that can dampen your spirit. The pandemic has given you unforgettable memories, many of which made you better. In all of them, learn to be grateful. Please know as well that we have people in USC who can journey with you and guide you in your time of need. Our team of counsellors and academic staff are more than willing to help you.

Certainly, many of you are no longer newbies in online or blended learning, for it has been more than two years already since you began attending classes from home. Even then, nothing beats face-to-face schooling, and we certainly hope and pray for the safety and wellbeing of everyone as we gradually go back to the "normal" academic life.

We cannot give in to fears and doubts, especially with COVID-19 still bearing down on us. Despite the numerous downsides of the current educational crisis, silver linings of hope and faith can be found. May this Primer, therefore, be a help to you as it guides you in navigating your way through flexible learning modalities in USC.

Palihug amping kanunay!

With you in keeping the Word alive,

Fr. Siso

Fr. Narciso A. Cellan, Jr., SVD
USC President

II Transforming the Educational Scene

The educational landscape is continuously being transformed to adapt to the changing times. Universities around the world are transforming the educational scene by introducing teaching innovation and multiple channels of learning.

The University of San Carlos continuously strives to deliver quality education in the Philippines and adopts its own methods to achieve this goal.

Guided by its core values and the missionary charism of the Society of the Divine Word and the life of San Carlos de Borromeo, USC continues to strive to provide timely, relevant, and transformative educational programs in the midst of these difficult times.

1. The USC Flexible Learning Plan (USC-FLEX)

The COVID-19 pandemic continues to pose challenges to the educational scene. Fortunately, the global vaccination drive has led to the reopening of schools around the world, albeit in limited capacity and with social distancing measures in place.

Cognizant of the risks that have to be mitigated and the progress that has been made possible, the University of San Carlos will adopt a flexible learning plan through our core values: ***Scientia***, ***Virtus***, and ***Devotio***.

Scientia translates to the quality of the flexible learning plan: modality, infrastructure, and human resources. ***Virtus*** translates to the integrity of the flexible learning plan. And ***Devotio*** translates to access to the flexible learning plan.

Through USC-FLEX, the University will continue to enlighten minds, hone skills, and transform characters of its students and the entire Carolinian community.

2. Delivery of Instruction in Three Modalities

The teaching and learning processes this semester will be delivered in three modalities: fully online, blended learning, and limited face-to-face. Each School will implement a default mode of instruction depending on the needs and objectives of their respective program courses.

a. Fully Online

Guidelines

A fully online mode of delivery will be implemented for courses and programs that can be delivered remotely i.e. those that do not have laboratory courses. The Schools will determine which programs and/or students can avail of this modality. Fully online classes will be delivered through different teleconferencing tools and via Canvas, USC's chosen Learning Management System (LMS).

Learning Management System (LMS)

Canvas is a web-based LMS where online courses, complete with learning materials and assessments, have been developed by the faculty. Canvas allows students to learn and receive feedback about progress throughout the semester. It helps students with:

- Keeping track of assigned works, grades, and feedback from instructors
- Organizing all learning materials that instructors have created
- Staying in touch with other students in the class

Additionally, while Canvas is primarily a web-based software, any user can access the Canvas Student app on any iOS and Android- based mobile device.

Teleconferencing Tools

Google Meet will be the primary teleconferencing tool in the conduct of online classes and other virtual meetings. Other teleconferencing platforms are also available such as MS Teams and Zoom for select units of the University. The USC webmail accounts are to be consistently used to ensure access to the different functions and features of the University's Google Education Plus subscription.

b. Blended Learning

Guidelines

Blended Learning combines face-to-face learning experiences with online learning. In the context of USC, students under this modality will undergo a mixed-mode learning where they can attend virtual classes for subjects that can be delivered remotely. It can also mean learning from the traditional face-to-face teaching with some technology-based inclusions.

For laboratory courses, in-person learning will be available; students shall also have access to laboratories and other campus resources, depending on their needs.

Functional on-site and online technical support

Since students will be learning online and on-site, technical support will be provided regardless of their learning environment. Academic and support services staff will be available in all campuses to provide assistance to students who are in face-to-face classes.

Meanwhile, all formal channels of communication including USC's official website, email, Facebook page, and other social media accounts will be active and available at all times.

c. Limited Face-to-Face

Guidelines

The most advanced learners in terms of maturity and level, particularly graduate students, will be prioritized for limited face-to-face classes. In this mode of instruction, students will be required to attend their classes on campus. The frequency of classes will depend on the schedule provided by the respective Schools.

Learning Spaces

The University will provide learning spaces for students under the limited face-to-face classes. Classrooms and study areas will be available to accommodate individualized or small-group activities. All learning spaces will require students to observe safety protocols and maintain social distancing at all times.

3. USC-FLP Alert Level-Based Activation Framework

a. Alert Level Classification Scheme (Levels 1 to 5)

USC will adopt an Alert Level Classification Scheme in line with the current classification of community quarantine. The activation of the different modalities of instruction and delivery will depend on the movement restrictions ranging from most restrictive full lockdown (Level 5) to the least restrictive modified general community quarantine (Level 1).

The University will implement a trigger-based response which is dependent on the announced quarantine classification at a given time.

In this model, Alert Level 1 means that all course types may be in a limited face-to-face setting. If Levels 2 and 3 are activated, all courses will be done online except for laboratory courses that will require limited face-to-face instruction. As for licensure-oriented courses, a case-to-case consideration will be applied. In the event that Levels 4 and 5 are announced, the classes will return to fully online modality for all courses including laboratory and other performance-based offerings.

b. Default Modality of Program Delivery

The default program delivery is fully online. In this model, face-to-face interaction will only be approved under highly exceptional circumstances. Students are advised to contact their respective schools and departments for instructions and information regarding program delivery.

c. Sequence of Prioritization

The alert level consideration originates from the most critical level (Level 5) where the default modality is fully online. In the event that the alert levels will be lowered down to the least restrictive setting, a sequence of prioritization will be followed.

From fully online, a limited face-to-face setup will be implemented. When the community restrictions are further lifted and Alert Level 1 prevails, a full face-to-face program delivery will be adopted by the University.

The alert level changes can take place in the middle of the semester, trimester, or school year. However, in consideration of the realities of USC students and personnel, a unidirectional change of modality will be applied.

If the mode of instruction is delivered through limited face-to-face at the start of the semester, in the event that the alert levels increase, the classes will gradually shift to blended learning and eventually, to a fully online setup.

Meanwhile, courses that are initially conducted fully online, will not be arbitrarily shifted towards a limited face-to-face modality in the middle of the semester.

4. Admission Options

The University of San Carlos is open for all. Keeping in mind the safety and security of everyone, vaccinated students are encouraged to enroll in courses and programs that will adopt limited face-to-face and blended modalities. Meanwhile, students who are yet to be vaccinated may apply in courses and programs that will adopt a fully online modality.

The Schools will announce the default modality for each program and course. Students may refer to their respective Schools and departments for more information and assistance.

5. Online and Onsite Behavior and Conduct

All students must observe at all times and especially on online platforms all the fundamental tenets of integrity, decency, fairness, cyber responsibility and due regard for human dignity.

Students must honor various legislations embodying intellectual property and data privacy and integrity such as Republic Act 10173 or the Data Privacy Act of 2012, and Republic Act 10175 or the Cybercrime Prevention Act of 2012, among others.

a. Online Learning

A guide for good online behaviour and appropriate use of the internet is provided herein.

General Guidelines:

- Come to an online meeting, forum, webinar or class properly groomed.
- Wear decent attire. It may be true that you are in the comforts of your home but dressing in your sleepwear does not give an impression of professionalism expected of students. Dressing decently prepares the mind and body that what is about to transpire is important.
- Comb/fix your hair and be neat in appearance.
- Be ready to flash your clean pearly whites online.
- Be on time. It is highly suggested to come 10 minutes before the time so that you can check if your equipment works, e.g. headphones, video.
- Be respectful. When typing your messages, avoid using all caps, exclamation points or enlarging fonts for emphasis. These are all equivalent to shouting.
- Be mindful of your grammar and spelling.
- Be polite. Address people properly. Wait until it is your turn to speak. Do not interrupt when another person is talking. Respect the opinions/ answers of others.
- Be considerate when making remarks. Use appropriate language.
- Be mindful of the data privacy requirements. For instance, taking of photos or screenshots without consent, and unauthorized sharing of personal information may have legal consequences.

Attending Online Classes

- Choose a place in your house where you can attend your online classes away from distraction such as noise from the TV.
- Be prepared. Be ready with your notebook and pen or anything that your teacher has asked you to prepare. Have these things within your reach.
- Once your online class has started, listen to your teacher and remain seated for the entire class period.

Attending Fora / Webinars

- Be prepared for the forum / webinar. Have a notebook and pen ready for taking down notes.
- Be present and be recognized. Officially sign on and off, and use your video so that everyone knows you are present.
- Engage and ask questions. Prepare your questions or remarks ahead of time to be able to provide engaging and concise messages.
- Use the chat box. Keep your statements brief and concise.
- Take time to answer the evaluation form that may be provided by the speaker/organizers of the forum/ webinar. This can encourage them and help them improve future activities.

b. In-campus Learning

Subject to strict government regulations and USC health protocols, all five campuses are open to the whole USC academic community.

Classrooms and laboratories will be opened subject to the approval of the University. Students who are in campus may avail of the internet connection inside the University's premises. Everyone must strictly observe all health and safety protocols found here under **Section IV. Health & Safety Protocols.**

III Academic Policies

Flexibility is the keyword that defines academic life in the University amid this global pandemic. At the outset, this required redesigning the curricula and rewriting syllabi while embracing constructivism and transactional distance education.

Teachers are expected to partner with students as knowledge builders who shall use their experiences, interactions, and observations as sources of information and understanding. Teachers have to share the responsibility of learning with their students. For education to be authentic and life-impactful, it must be holistic.

1. Schedule of Classes (Online, Blended, Face-to-Face)

a. Undergraduate Programs

Guided by the Flexible Learning Plan adopted by the University, classes will follow a fully-online learning method with a combination of synchronous and asynchronous teaching-learning modes, as well as the blended learning option which allows for limited face-to-face activities.

In the fully online learning mode, online classes will be held in real-time based on a set of schedules per class. Asynchronous learning happens offline enabling students to carry out independent activities. For blended learning, the details regarding online classes and limited face-to-face interaction will be announced by the respective departments. Schedule of limited face-to-face instruction will also be determined by each School and/or departments.

b. Graduate Programs

As is the nature of graduate studies, independent learning will continue to be the norm. However, there will be instances where online classes will be held, depending on the different policies adopted by the various schools in the University.

The respective School Deans shall disseminate a separate detailed advisory specific to their programs regarding this matter.

c. International Students

New international students interested in availing of online learning arrangements are advised to immediately access https://ismis.usc.edu.ph (International Student Applicants) for admission-related information.

2. In-Person Lab Classes

Physical entry into laboratories will be allowed, provided that all health protocols are strictly observed.

3. Thesis/Dissertation Proposal and Oral Defense

Proposal and oral defense hearings in graduate and undergraduate programs shall be carried out either via different online platforms with the assurance that students and panelists have stable internet connectivity or through a face-to-face panel defense.

4. Graduation Ceremonies

Graduation ceremonies will be scheduled this semester. The Office of the Registrar shall issue an advisory on the deadline of applications for graduation.

5. Academic Calendar

2022

AUGUST

S	M	T	W	T	F	S
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	~~15~~	~~16~~	~~17~~	~~18~~	~~19~~	~~20~~
21	~~22~~	~~23~~	~~24~~	~~25~~	~~26~~	~~27~~
28	29	~~30~~	~~31~~			

SEPTEMBER

S	M	T	W	T	F	S
				~~1~~	~~2~~	~~3~~
4	~~5~~	~~6~~	~~7~~	~~8~~	9	~~10~~
11	~~12~~	~~13~~	~~14~~	~~15~~	~~16~~	~~17~~
18	~~19~~	~~20~~	~~21~~	~~22~~	~~23~~	~~24~~
25	~~26~~	~~27~~	~~28~~	~~29~~	~~30~~	

OCTOBER

S	M	T	W	T	F	S
						~~1~~
2	~~3~~	~~4~~	~~5~~	~~6~~	~~7~~	~~8~~
9	~~10~~	~~11~~	~~12~~	~~13~~	~~14~~	~~15~~
16	~~17~~	~~18~~	~~19~~	~~20~~	~~21~~	~~22~~
23	~~24~~	~~25~~	~~26~~	~~27~~	~~28~~	~~29~~
30	~~31~~					

NOVEMBER

S	M	T	W	T	F	S
		1	2	~~3~~	~~4~~	~~5~~
6	~~7~~	~~8~~	~~9~~	~~10~~	~~11~~	~~12~~
13	~~14~~	~~15~~	~~16~~	~~17~~	~~18~~	~~19~~
20	~~21~~	~~22~~	~~23~~	~~24~~	~~25~~	~~26~~
27	~~28~~	~~29~~	30			

DECEMBER

S	M	T	W	T	F	S
				~~1~~	~~2~~	~~3~~
4	~~5~~	~~6~~	~~7~~	8	~~9~~	~~10~~
11	~~12~~	~~13~~	~~14~~	~~15~~	~~16~~	~~17~~
18	19	20	21	22	23	24
25	26	27	28	29	30	31

Legend:

☑ School Day ☐ Holiday

FIRST SEMESTER 2022-2023

Aug 2-14 - Online Enrollment 1st Semester
Aug 6 - Cebu Province Charter Day
Aug 15 - **CLASSES START**
Aug 21 - Ninoy Aquino Day
Aug 29 - National Heroes Day
Sept 8 - SVD Foundation Day
Sept 9 - Osmeña Day
Sept 15-16 - **PRE-MIDTERM EXAMS**
Oct 5 - World Teachers' Day
Oct 17-22 - **MID-TERM EXAMS**
Oct 24 -29 - University Days
Nov 1-2 - All Saints Day / All Souls Day
Nov 4 - Feast day of San Carlos Borromeo
University President's Message
Nov 17-18 - **PRE-FINAL EXAMS**
Nov 30 - Andres Bonifacio Day
Dec 8 - Feast of Immaculate Conception
Dec 12-17 - **FINAL EXAMS of Graduating and Non-Graduating students**
Dec 17 - CLASSES END
Dec 25 / Dec 30 - Christmas Day / Jose Rizal Day
Jan 21, 2023 - Commencement Rites

SECOND SEMESTER 2022-2023

Jan 1 - New Year's Day
Jan 2-15 - Online Enrollment 2nd Semester
Jan 15 - Feast of St. Arnold Janssen
- Feast of Santo Niño
Jan 16 - **CLASSES START**
Jan 29 - Feast of St. Joseph Freinademetz
Feb 9-10 - **PRE-MIDTERM EXAMS**
Feb 24 - Cebu City Charter Day
Feb 25 - People Power Anniversary
Mar 13-18 - **MID-TERM EXAMS**
Apr 6-7 - Maundy Thursday / Good Friday
Apr 9 - Araw Ng Kagitingan (Day of Valor)
Apr 13-14 - **PRE-FINAL EXAMS**
Apr 20-21 - University Research Conference
Apr 22 – Eid al Fitr
May 1 - Labor Day
May 15-20 - **FINAL EXAMS of Graduating and Non-Graduating students**
May 20 - **CLASSES END**
Jun 2-3 - Commencement Rites

SUMMER 2023

Jun 1-11 - Online Enrollment Summer Term
Jun 12 - Independence Day
Jun 13 - **CLASSES START**
Jun 29-30 - **MID-TERM EXAMS**
Jul 20-21 - **FINAL EXAMS**
Jul 21 - **CLASSES END**

2023

JANUARY

S	M	T	W	T	F	S
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	~~16~~	~~17~~	~~18~~	~~19~~	~~20~~	~~21~~
22	~~23~~	~~24~~	~~25~~	~~26~~	~~27~~	~~28~~
29	~~30~~	~~31~~				

FEBRUARY

S	M	T	W	T	F	S
			~~1~~	~~2~~	~~3~~	~~4~~
5	~~6~~	~~7~~	~~8~~	~~9~~	~~10~~	~~11~~
12	~~13~~	~~14~~	~~15~~	~~16~~	~~17~~	~~18~~
19	~~20~~	~~21~~	~~22~~	~~23~~	24	25
26	~~27~~	~~28~~				

MARCH

S	M	T	W	T	F	S
			~~1~~	~~2~~	~~3~~	~~4~~
5	~~6~~	~~7~~	~~8~~	~~9~~	~~10~~	~~11~~
12	~~13~~	~~14~~	~~15~~	~~16~~	~~17~~	~~18~~
19	~~20~~	~~21~~	~~22~~	~~23~~	~~24~~	~~25~~
26	~~27~~	~~28~~	~~29~~	~~30~~	~~31~~	

APRIL

S	M	T	W	T	F	S
						~~1~~
2	~~3~~	~~4~~	~~5~~	6	7	8
9	~~10~~	~~11~~	~~12~~	~~13~~	~~14~~	~~15~~
16	~~17~~	~~18~~	~~19~~	~~20~~	~~21~~	22
23	~~24~~	~~25~~	~~26~~	~~27~~	~~28~~	~~29~~
30						

MAY

S	M	T	W	T	F	S
	1	~~2~~	~~3~~	~~4~~	~~5~~	~~6~~
7	~~8~~	~~9~~	~~10~~	~~11~~	~~12~~	~~13~~
14	~~15~~	~~16~~	~~17~~	~~18~~	~~19~~	~~20~~
21	22	23	24	25	26	27
28	29	30	31			

JUNE

S	M	T	W	T	F	S
				1	2	3
4	5	6	7	8	9	10
11	12	~~13~~	~~14~~	~~15~~	~~16~~	17
18	~~19~~	~~20~~	~~21~~	~~22~~	~~23~~	24
25	~~26~~	~~27~~	~~28~~	~~29~~	~~30~~	

JULY

S	M	T	W	T	F	S
						1
2	~~3~~	~~4~~	~~5~~	~~6~~	~~7~~	8
9	~~10~~	~~11~~	~~12~~	~~13~~	~~14~~	15
16	~~17~~	~~18~~	~~19~~	~~20~~	~~21~~	22
23	24	25	26	27	28	29
30	31					

SUMMARY OF CLASS DAYS

First Semester			Second Semester			Summer Term		
Months	Weeks	Days	Months	Weeks	Days	Months	Weeks	Days
Aug	3	14	Jan	3	14	June	3	14
Sept	4	25	Feb	4	22	July	3	15
Oct	4	26	Mar	4	27			
Nov	4	23	Apr	4	21			
Dec	3	14	May	3	17			
Total	**18**	**102**	**Total**	**18**	**101**	**Total**	**6**	**29**

IV Health and Safety Protocols

1. On-site Safety Guidelines

Students, faculty and administrative personnel must strictly observe proper protection measures at all times including mandatory wearing of face mask, hand washing, and physical distancing when entering any of the five campuses.

Anyone entering any of the University campuses will undergo a temperature check and will be asked to fill up a health declaration form. The University reserves the right to disallow entry to anyone who does not wish to undergo this procedure. If visitors or students have a recorded temperature of 37.5 degrees Celsius or higher, they shall be advised to leave the premises, stay home and/or seek medical advice. Further details are stipulated in Health and Safety Guidelines of the University.




2. Counseling and Mental Health Support Services (Online and Onsite)

Students can avail of counseling services spearheaded by the USC Mental Health Online Support through:

https://www.facebook.com/USCConselingandDevelopmentCenter.

Students may also visit their school's onsite counseling center located at their respective campus and school building.

V Student Support Services

1. Connectivity Support (Digital and On-site)

a. Digital support systems

The University provides an array of support for both online and on-site classes. Offices are open to efficiently address different student concerns during face-to-face activities. Moreover, Caloy, the student support and services avatar of the University can be accessed through ISMIS.

b. Online counterparts of all essential onsite support units

Students availing of the fully online mode of learning can continue to avail online counterparts of all essential support units. Mental health support services, e-library services, and other academic-related inquiries can be accessed through different social media platforms and online channels.

c. Formal Channels of Communication

Regardless of learning modality, students can reach out to USC's different formal channels of communication. For the latest news and updates, the USC website and official Facebook page are good sources of information. Email <info@usc.edu.ph> and telephone assistance <+63 32 230 0100> are also available for urgent inquiries and concerns.

2. Library Services

The Library System has designed both online and onsite services to support the academic community's curricular and research needs in the midst of the pandemic.

This semester, the following libraries will open their doors for a limited number of users per day subject to IATF, LGU and the University's own health and safety protocols:

Libraries	Time	Number of Users Allowed per Day
Talamban Campus Josef Baumgartner Learning Resource Center	8:00 AM - 12:00 Noon	50 (Only the Ground floor reading area with be open)
Downtown Campus **Bonk Library (SBE Library)** **American Corner** **Law Library**	8:00 AM - 12:00 Noon 8:00 AM - 12:00 Noon 1:00 PM – 5:00 PM	15 (Allowed at one time) 5 (Allowed at one time) 25 (Allowed at one time)
South Campus Education Library	8:00 AM - 12:00 Noon	15

1. On-site/In-Person:

References

Reach out to the librarian in-person regarding research and information needs. The librarians will be glad to offer assistance.

Book Drop and Curbside Services

The process of returning borrowed books (Book Drop) or borrowing books from the Library will continue to be safe in the new normal through the use of Book Drop/Curb- side stations outside of your particular Library or in strategic places within the campus. These stations are open from 8:00 AM to 4:00 PM, Mondays to Fridays. The safety protocol for pick-up and return of books will be sent by email.

Self-check Station (LRC only)

Allows users to self-issue items without assistance from the librarian. Complete with touch screen and receipt printer. This is a DIY service.

Stack service

The libraries have currently adopted closed stack service to ensure the safety of users and librarians. However, upon request, the librarians will readily retrieve books from the stack area for the students and other users.

WIFI zone

For internet access, all libraries are WIFI zones.

3. Online Services:

These online resources are your gateway to the Library's online resources: https://ezproxy.usc.edu.ph or https://library.usc.edu.ph

For assistance, please email, chat, SMS, text, call the respective librarians:

Name of Librarian	Location	Contact Information
Ambos, Gesyl	Education Library 2nd Floor, Anthony Buchcik Bldg. South Campus	geambos@usc.edu.ph 2300-100 loc 742
Caro, Mary Gwyn	American Corner/ Caro, Mary Gwyn Political Science Library 2nd Floor, Dingman Bldg,USC Downtown Campus Social Science Library 4th Floor, LRC, Talamban Campus	mlcaro@usc.edu.ph 2300-100 loc 568
Concepcion, Irish	EducUSA Center Upper 2nd Floor, LRC, Talamban Campus	igconcepcion@usc.edu.ph 2300-1 00 loc 199
Gabat, Jacquelyn	Humanities Library 2nd Floor, LRC, Talamban Campus	jagabat@usc.edu.ph 2300-100 loc 197
Javier, Rave Ann	Filipiniana Library Upper 3rd Floor, LRC, Talamban Campus	rejavier@usc.edu.ph 2300-100 loc 309
Lapad, Mari-Rose	Cebuano Studies Center Library Upper 2nd Floor, LRC, Talamban Campus	mylapad@usc.edu.ph 2300-100 loc 308
Litonjua, Moreta	KNC/VTR Ground Floor, LRC, Talamban Campus	mclitonjua@usc.edu.ph 2300-100 loc 192
Sumalinog, Sharyn Keith	Science and Technology Library 3rd Floor, LRC, Talamban Campus	stsumalinog@usc.edu.ph 2300-100 loc 193
Mission, Narcisa	Director of Libraries Office LRC, Talamban Campus	nmmission@usc.edu.ph 2300-100 loc 194
Monguez, Jesel	Circulation/ Information Desk General Reference/SHS Libraries Ground Floor, LRC, Talamban Campus	jimonguez@usc.edu.ph 2300-100 loc 198
Onde, Mark Francis	CALM Unit Upper Ground Floor, LRC, Talamban Campus	mfgonde@usc.edu.ph 2300-100 loc 307
Pastera, Edgar	Law Library 5th Floor, Ernest Hoerdemann Bldg, Downtown Campus	ebpastera@usc.edu.ph 2300-100 loc 569
Ramos, Sharlene	Serials Library 3rd Floor, LRC, Talamban Campus	sg ramos@usc.edu.ph 2300-100 loc 310
Tenorio, Divina	Bonk Library (SBE) 2nd Floor, Arthur Dingman Bldg., Downtown Campus	dctenorio@usc.edu.ph 2300-100 loc 570

Article Request via DigDDs

Email us and we will assist you through the Digital Document Delivery Service (DigDDs). We can scan and deliver via email a page(s), chapter(s) or an article(s) that you need.

Webliography/OER your web reading list guide

You can discover sites/articles on topics of interest to you. At the same time, you can connect with experts in the field.

Online Instruction

We conduct assistance on accessing and navigating the online databases, OPAC, and other electronic resources.

Online Events

You may join the any of Webinars and MOOCs organized by the Library.

Tutorials and Vlogs

For asynchronous learning to discover and explore the University's libraries, you may subscribe to our YouTube Channel: USC Library

Social Media Channels

Facebook: **University of San Carlos Library System**
Instagram: **@usclibrarysystem**
Youtube: **USC Library**

The University of San Carlos

The Word Alive.

www.usc.edu.ph

Prepared by **USC Corporate Communications Office**
communications@usc.edu.ph