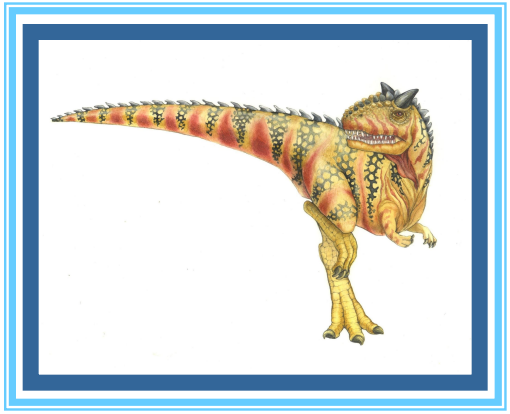
Unidad 1: Introducción



Capítulo 1: Introducción

- Que hacen los Sistemas Operativos?
- Estructura de un Sistema Operativo
- Operaciones de un Sistema Operativo
- Administración de Procesos
- Administración de Memoria
- Estructuras de datos del Kernel
- Ejemplos de Sistemas Operativos

Objectivos

- Describir la organización básica de una computadora
- Proveer una visión general de los componentes más importantes de un Sistema Operativo

Ejemplos de Sistemas Operativos

- Sistemas que vienen instalados en la compu:
  Microsoft Windows (83%), **Mac OS (11.2%)**, GNU/**Linux (1.55%)**




Mac OS X

- Sistemas operativos para dispositivos móviles:
  Google Android, Apple iOS, **Windows Phone**

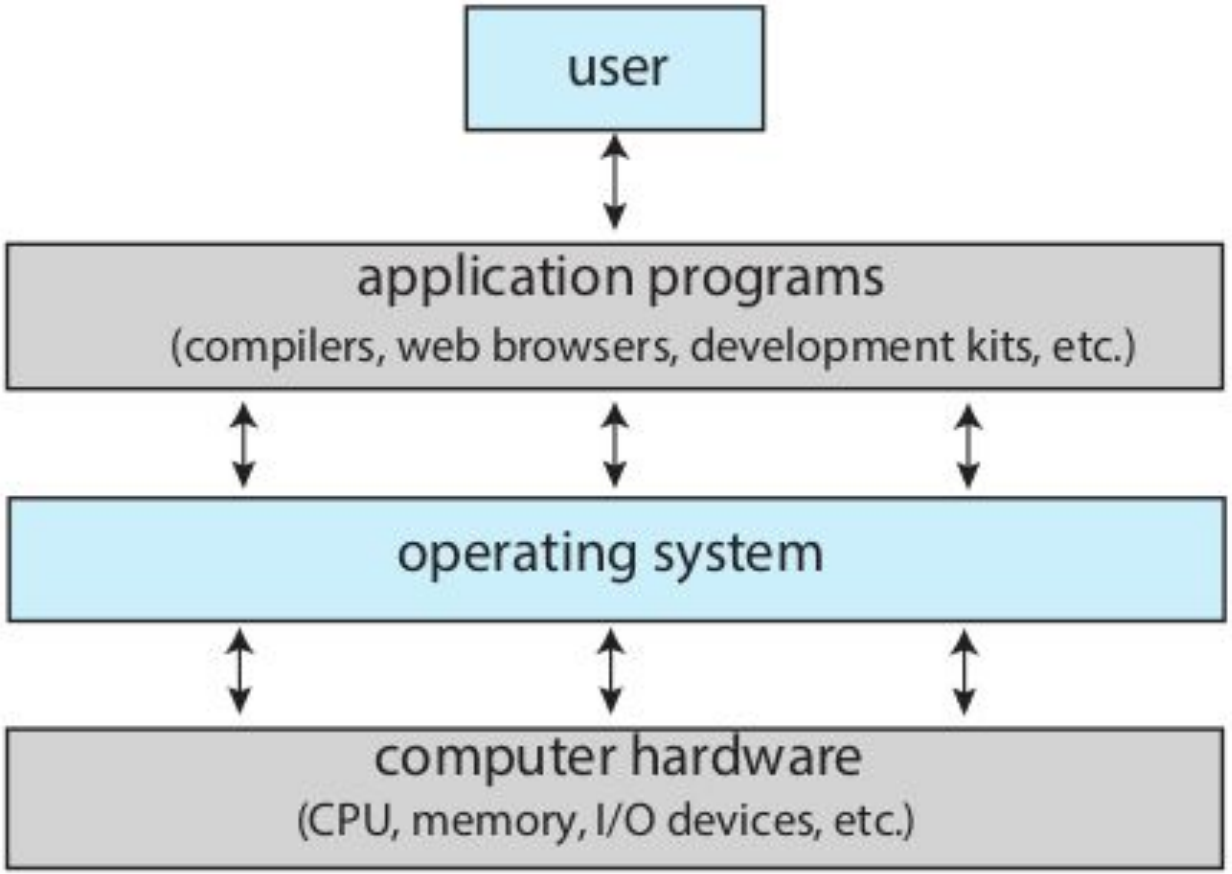


ANDROID

- Sistemas operativos en **servidores y grandes centros de cómputo**:
  GNU/**Linux Debian**


Qué es un Sistema Operativo?

- Un programa que actúa como intermediario entre un usuario de computadora y el hardware de la misma.



Qué es un Sistema Operativo?

- Objetivos:
  - Ejecutar programas de usuario y facilitar la resolución de problemas para el.
  - Hacer que el sistema sea cómodo de usar.
  - Utilizar el hardware de la computadora de manera eficiente.

Definición de Sistema Operativo

- SO es un **administrador de recursos**
  - Maneja todos los recursos
  - Decide entre solicitudes en conflicto para un uso eficiente y justo de los recursos
- SO como **programa de control**
  - Controla la ejecución de programas para prevenir errores y el uso incorrecto de la computadora
- SO ofrece una interfaz a los usuarios que buscan comodidad, **facilidad de uso** y **buen rendimiento** y no se preocupan por la **utilización de recursos.**

Definición de Sistema Operativo (Cont.)

- No hay una definición universal
- "El único programa que se ejecuta en todo momento en la computadora" el **kernel**.

En Linux, para ver los procesos ordenados por PID ejecutar: **ps -eaf**

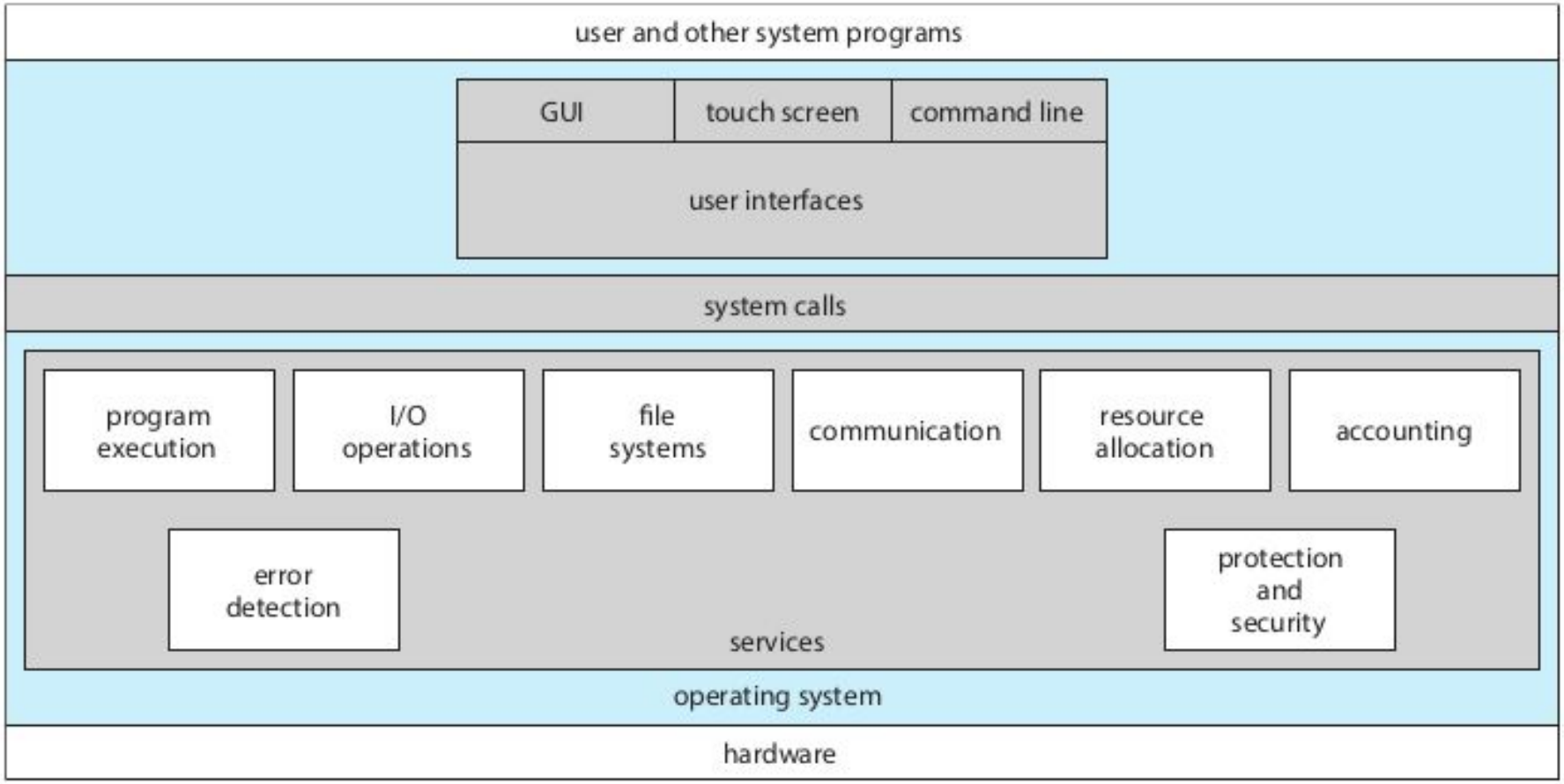
El proceso PID 1 (init) es responsable de iniciar y apagar el sistema
El proceso PID 2 (kthreadd) es responsable de lanzar los procesos o tareas del kernel. Para ver los procesos hijos del proceso 2 ejecutar:
ps --ppid 2 -p 2 -o uname,pid,ppid,cmd,cls

El proceso PID 0 no se muestra pero es el primer proceso que se ejecuta en tiempo de booteo. Tiene la función idle process, ser el proceso que se ejecuta cuando no hay nada para ejecutar.

Estructura de un SO

- Una computadora puede dividirse en cuatro componentes:
  - Hardware – provee los recursos computacionales básicos
    - CPU, memoria, dispositivos de I/O
  - Sistema Operativo
    - Controla y coordina el uso del hardware entre varias aplicaciones y usuarios
  - Programas de aplicación o usuario – define la manera en que se utilizan los recursos para resolver problemas de usuario
    - Compiladores, browsers, juegos, etc.
  - Usuarios
    - Personas, máquinas, otras computadores

Estructura de un SO

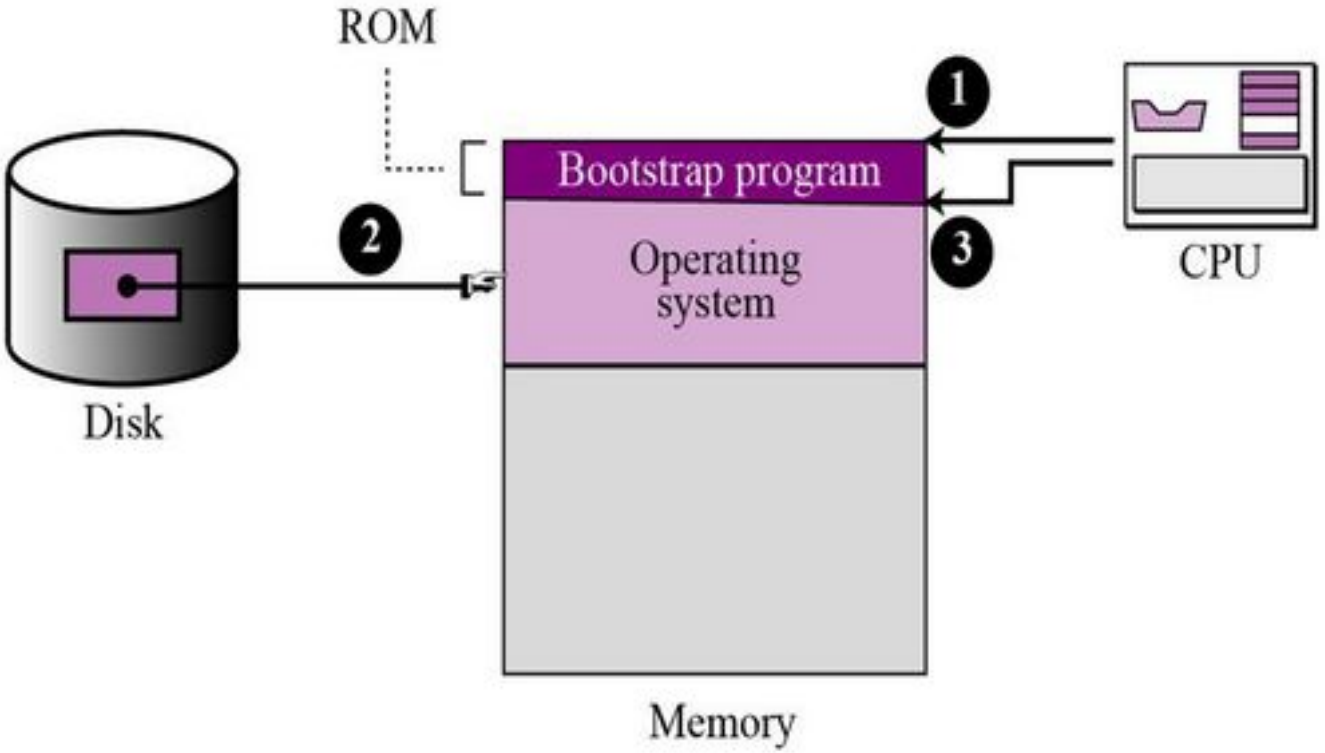


Inicialización

- **bootstrap program** es cargado en el encendido o el reinicio de la computadora
  - Típicamente alojado en ROM o EPROM, generalmente conocido como **firmware**
  - Inicializa todos los aspectos del sistema
  - Carga el kernel del sistema operativo y comienza su ejecución

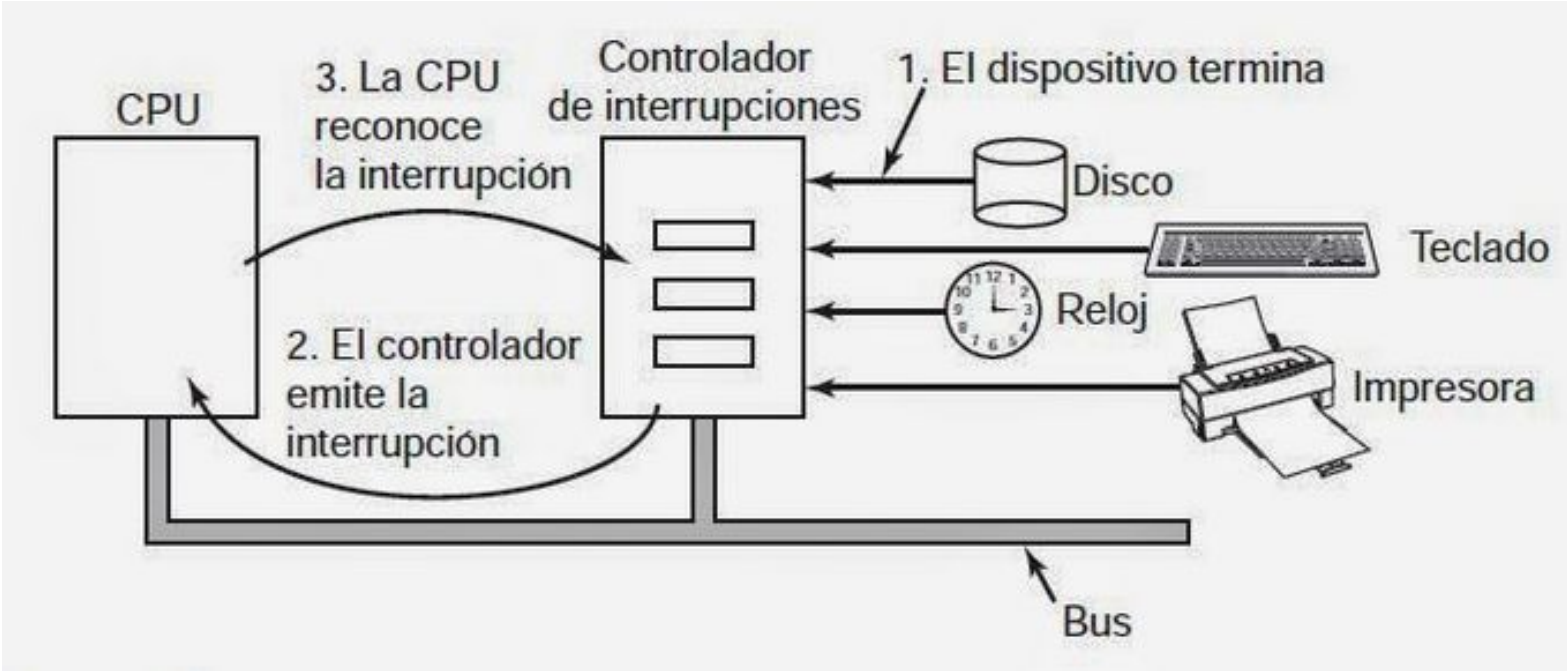
Inicialización

- **bootstrap program** está usualmente en la BIOS (Read Only Memory - ROM) de la motherboard.



Funciones Comunes e Interrupciones

- Un sistema operativo es accionado por interrupciones



Funciones Comunes e Interrupciones

- Las interrupciones transfieren el control a la rutina de servicio de interrupción generalmente, a través del **vector de interrupciones**, que contiene las direcciones de todas las rutinas de servicio
- La arquitectura de interrupción debe guardar la dirección de la instrucción interrumpida
- Una **trap** o **exception** es una interrupción generada por software causada por un error o una solicitud del usuario.

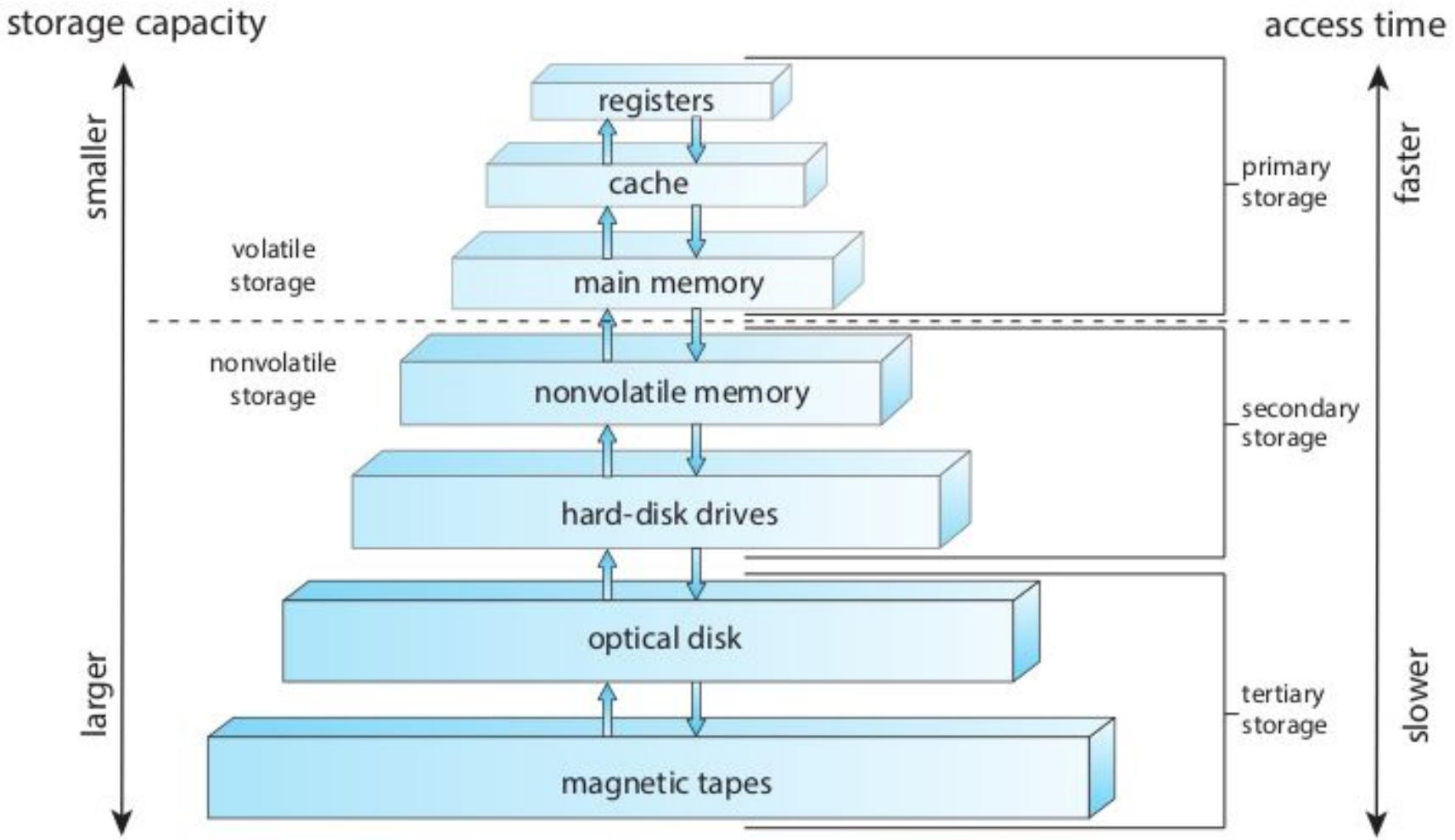
Manejo de Interrupciones

- El sistema operativo conserva el estado de la CPU al almacenar los registros y el contador del programa.
- Determina qué tipo de interrupción ocurrió:
  - **polling** (La CPU comprueba constantemente el estado del dispositivo si necesita atención de la CPU)
  - **vector de interrupciones** (El dispositivo notifica a la CPU que necesita atención de la CPU)
- Segmentos separados de código determina qué acción se debe tomar para cada tipo de interrupción

Administración de Memoria

- Para ejecutar un programa, todas ó parte de las instrucciones debe estar en la memoria
- El administrador de memoria determina que y cuando se carga en memoria principal
- Actividades del administrador de memoria
  - Guardar registro de que partes de la memoria están siendo utilizadas y por quien
  - Decidir qué procesos o datos deben ser cargados en la memoria
  - Alojar y desalojar memoria en la medida que sea necesario

Jerarquía de almacenamiento



Definiciones de almacenamiento

La unidad básica de almacenamiento informático es el **bit**. Un bit puede contener uno de dos valores, 0 y 1. Todo el almacenamiento en una computadora se basa en colecciones de bits. Dados suficientes bits, es sorprendente la cantidad de cosas que una computadora puede representar: números, letras, imágenes, películas, sonidos, documentos y programas, por nombrar algunos.

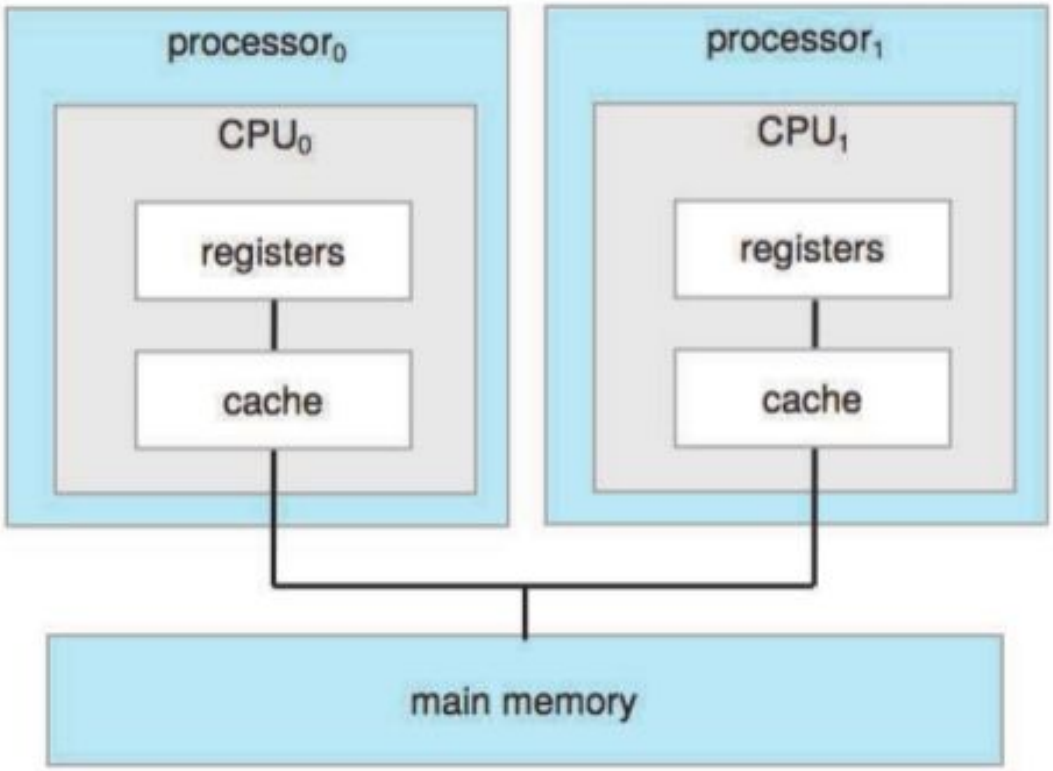
Un **byte** son 8 bits, y en la mayoría de las computadoras es la porción de almacenamiento más pequeña y conveniente. Por ejemplo, la mayoría de las computadoras no tienen una instrucción para mover un **bit**, pero sí una para mover un byte. Un término menos común es **word** o **palabra**, que es la unidad de datos nativa de una arquitectura de computadora dada. Una palabra se compone de uno o más bytes. Por ejemplo, una computadora que tiene registros de 64 bits y direccionamiento de memoria de 64 bits generalmente tiene palabras de 64 bits (8 bytes). Una computadora ejecuta muchas operaciones en su tamaño de palabra nativo en lugar de un byte a la vez.

Unidades de medida:
A **kilobyte**, or **KB**, is 1,024 bytes
a **megabyte**, or **MB**, is $1{,}024^2$ bytes
a **gigabyte**, or **GB**, is $1{,}024^3$ bytes
a **terabyte**, or **TB**, is $1{,}024^4$ bytes
a **petabyte**, or **PB**, is $1{,}024^5$ bytes

Arquitecturas

- Los sistemas solían utilizar un único procesador de propósito general
- **Multiprocesador** sistemas que crecen en uso e importancia.
  - También conocidos como **sistemas paralelos**
  - Ventajas:
    1. **Incrementa el throughput (tasa de procesamiento de trabajo)**
    2. **Mayor confiabilidad - tolerancia a fallas**

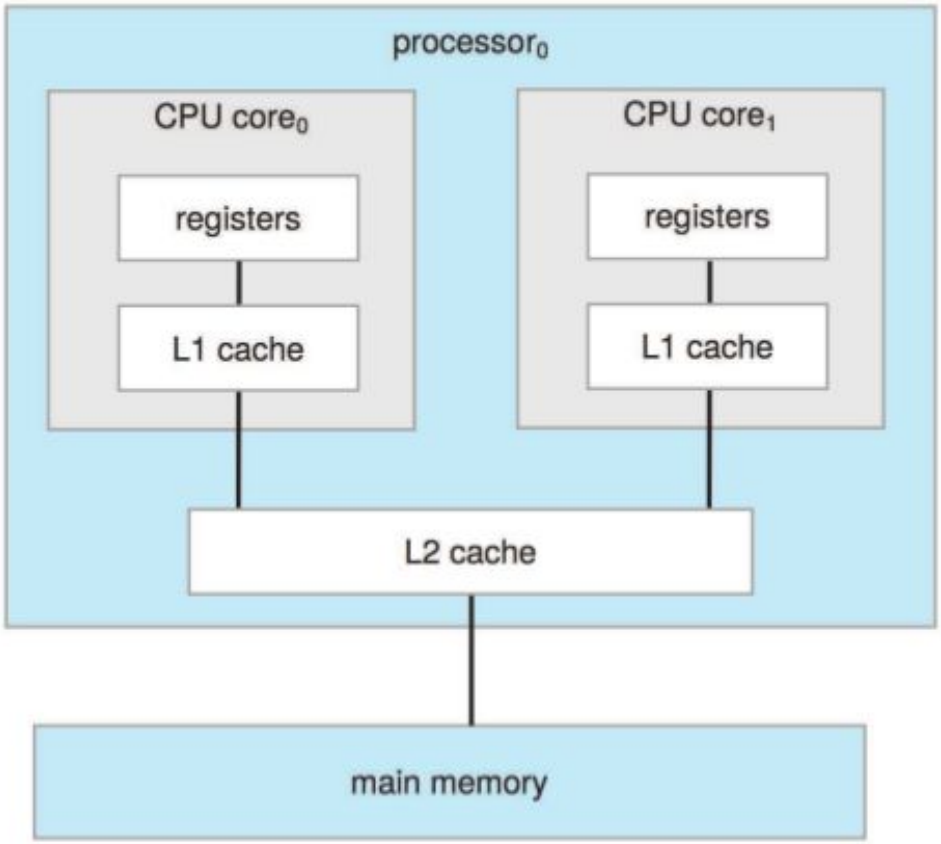
Arquitectura de Multiprocesamiento Simétrico



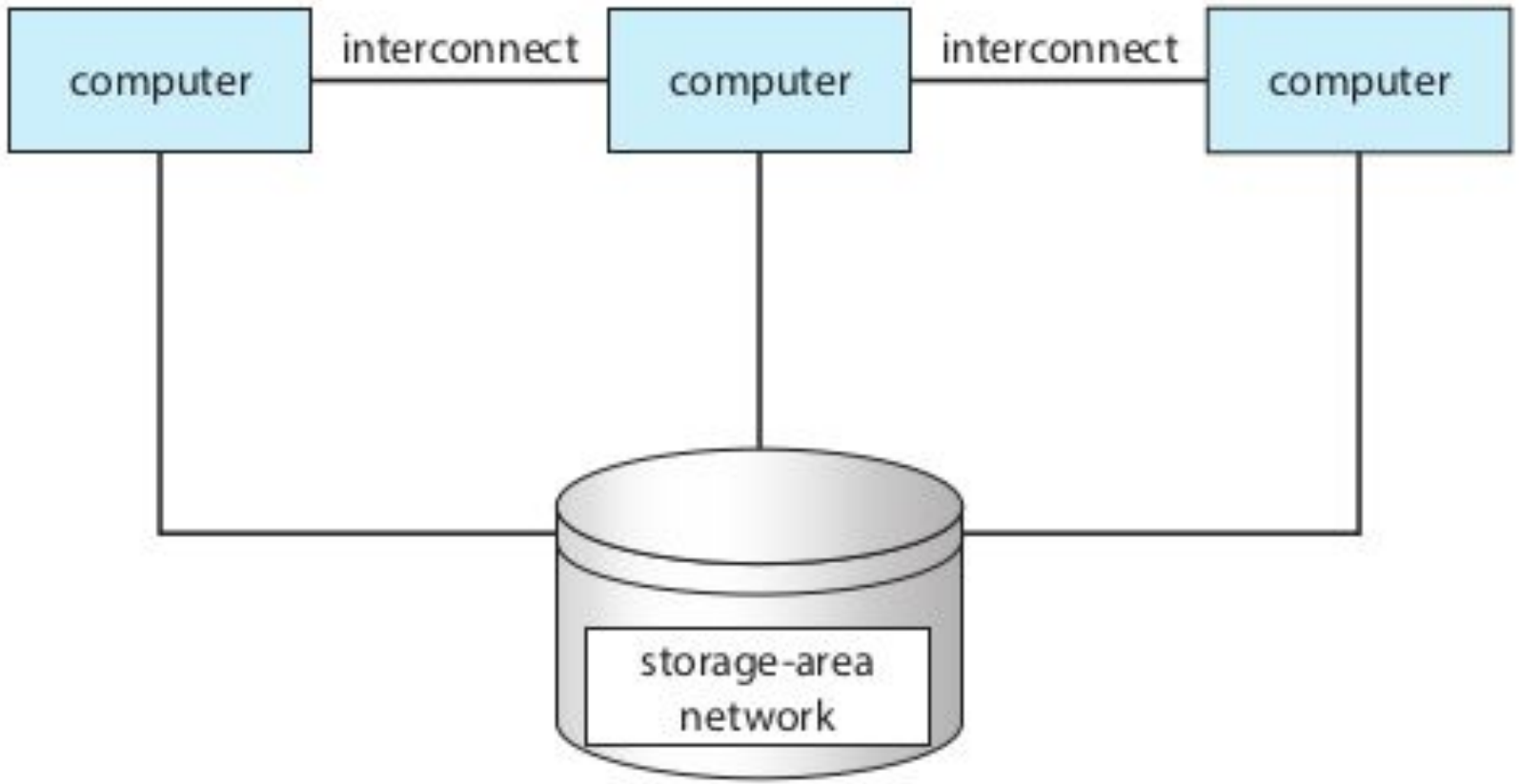
Es la arquitectura más utilizada. Cada cpu realiza todas las tareas incluyendo las del sistema operativo y de los usuarios

Diseño Dual-Core

- Sistemas multiprocesamiento evolucionaron a **multicore**
- La comunicación on-chip es mucho más rápida



Sistemas Cluster



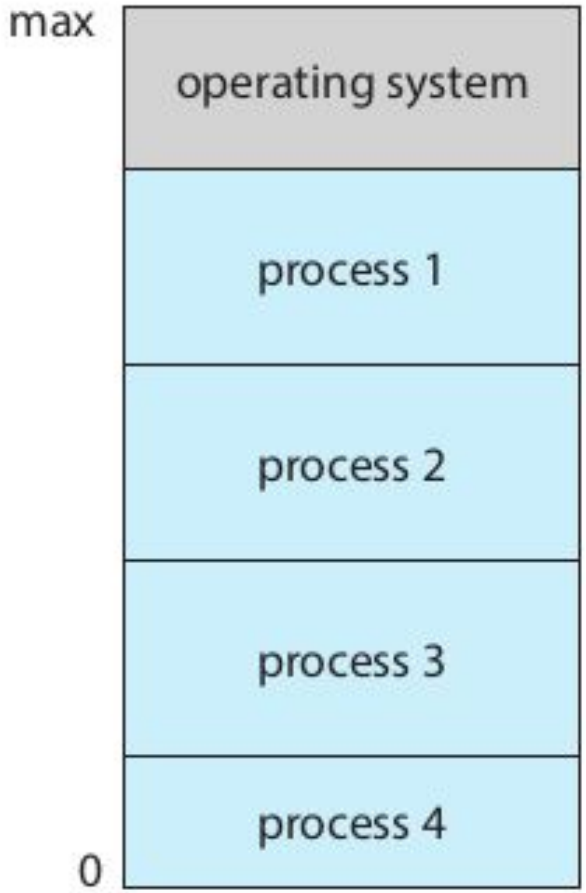
Sistemas Cluster

- Como sistemas multiprocesador, pero múltiples sistemas trabajando juntos
  - Usualmente se comparte el almacenamiento via **storage-area network (SAN)**
  - Provee servicio de **alta-disponibilidad** resistente a fallas
    - **Clustering Asimétrico** tiene una máquina en modo hot-standby
    - **Clustering Simétrico** tiene múltiples nodos corriendo aplicaciones, se monitorean mutuamente
  - Algunos clusters son para **high-performance computing (HPC)**
    - Las aplicaciones hacen uso del **paralelismo**
  - Algunos tienen **administrador de locks distribuido** (**DLM**) para evitar operaciones conflictivas

Multiprogramación

- **Multiprogramación**
  - La **multiprogramación** organiza trabajos (código y datos) para que la **CPU siempre tenga uno para ejecutar**
  - Un conjunto de tareas se mantienen en memoria
  - Se selecciona un trabajo , mediante **scheduling**
  - Cuando tiene que esperar (por ejemplo, para E/S), el sistema operativo cambia a otro trabajo

Multiprogramación



- Memoria en un sistema con **Multiprogramación**

Multitarea

- **Multitarea** (**multitasking**) es una extensión del concepto de multiprogramación en donde la CPU, cambia los trabajos con **tanta frecuencia** que los usuarios pueden interactuar con cada trabajo mientras se ejecuta, creando una computación interactiva
  - **Response time** , tiempo de respuesta < 1 seg
  - Cada usuario tiene al menos un programa cargado en la memoria ⇨**proceso**
  - Si existen varios trabajos listos para ejecutarse ⇨ **CPU scheduling**
  - Si los procesos no caben en la memoria , se realiza **swapping/paging**

Administración de Procesos

- Un proceso es un programa en ejecución. Es una unidad de trabajo dentro del sistema. El programa es una entidad pasiva, el proceso es una entidad activa.

- Los procesos necesitan recursos:
  - CPU, memory, I/O, files
  - Datos de inicialización

- Normalmente, el sistema tiene muchos procesos, algunos usuarios, algunos sistemas operativos que se ejecutan simultáneamente en una o más CPU
  - La Concurrencia se da al multiplexar las CPUs entre los procesos / hilos

Administración de Procesos

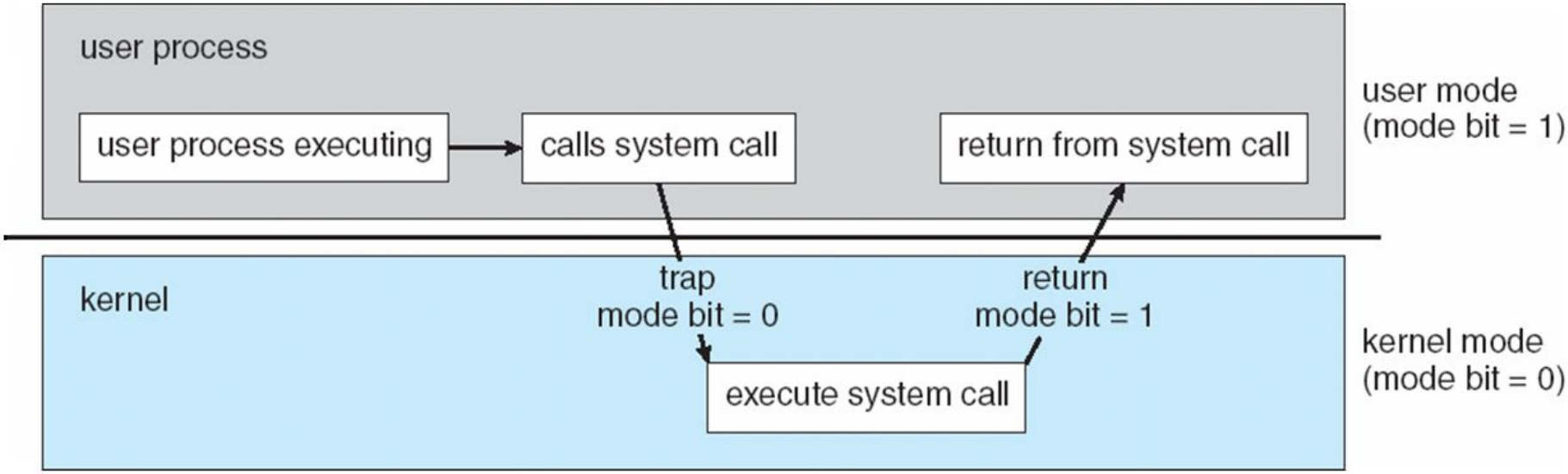
El sistema operativo es responsable de las siguientes actividades relacionadas con la gestión de procesos:

- Creación y eliminación de procesos de usuario y sistema.
- Suspender y reanudar procesos.
- Proporcionar mecanismos para la sincronización de procesos.
- Proporcionar mecanismos para la comunicación de procesos.
- Proporcionar mecanismos para el manejo de deadlock

Modo Kernel vs Modo Usuario

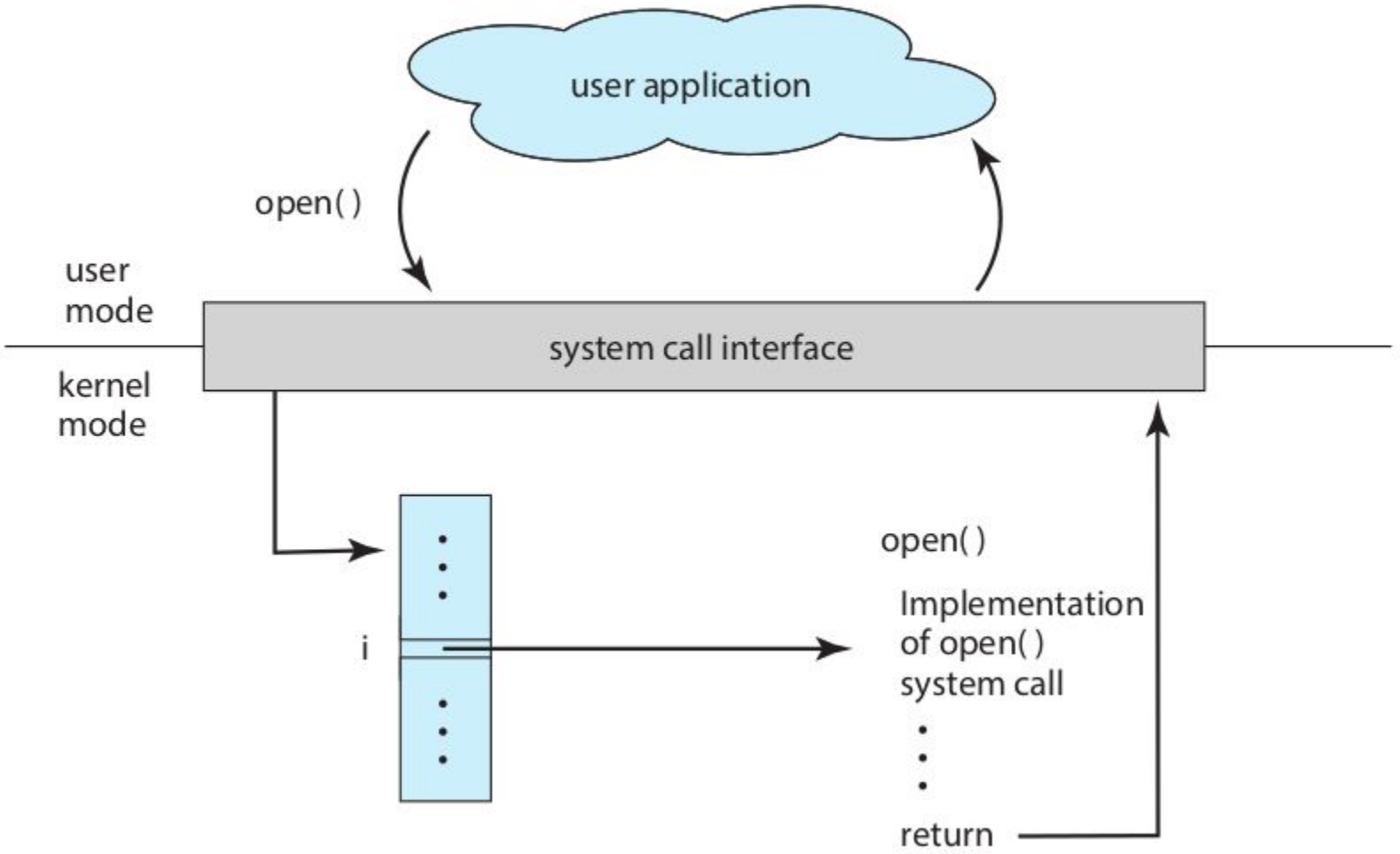
- **Dual-mode** o el SO se protege a sí mismo y a otros componentes del sistema
  - **User mode** y **kernel mode**
  - **Mode bit** provisto por hardware
    - Permite distinguir cuando el sistema está ejecutando código de usuario o de kernel
    - Algunas instrucciones designadas como **privileged**, solo se ejecutan en modo Kernel
    - Las llamadas al sistema (**System call**) se ejecutan en modo Kernel y luego retornan a modo usuario

Transición de modo Usuario a modo Kernel



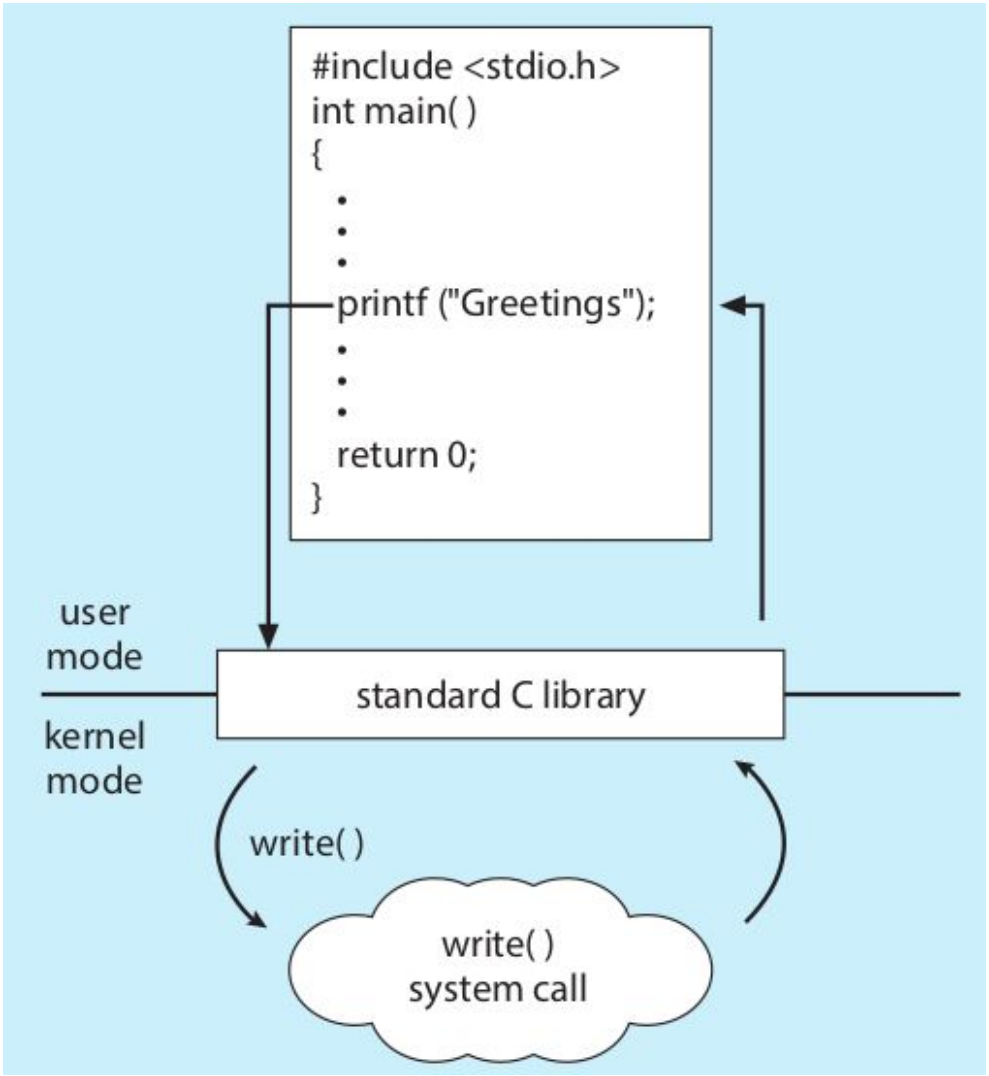
Transición de modo Usuario a modo Kernel

Ejecutando la llamada al sistema open()

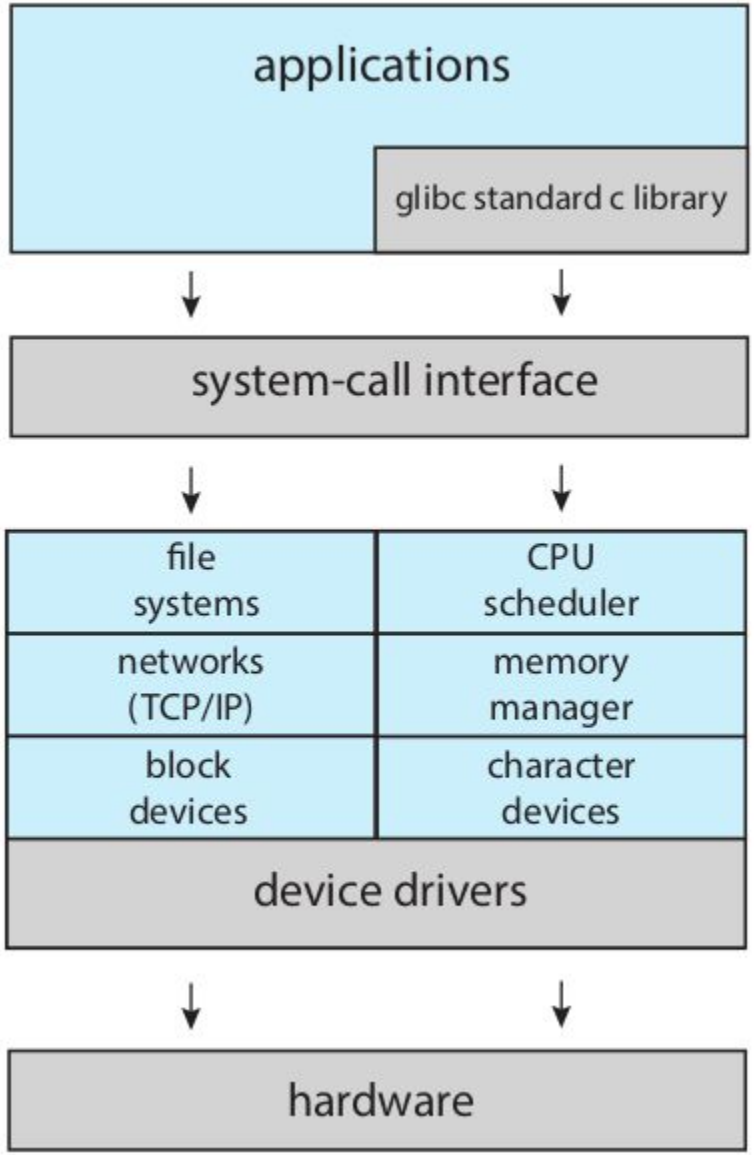


Transición de modo Usuario a modo Kernel

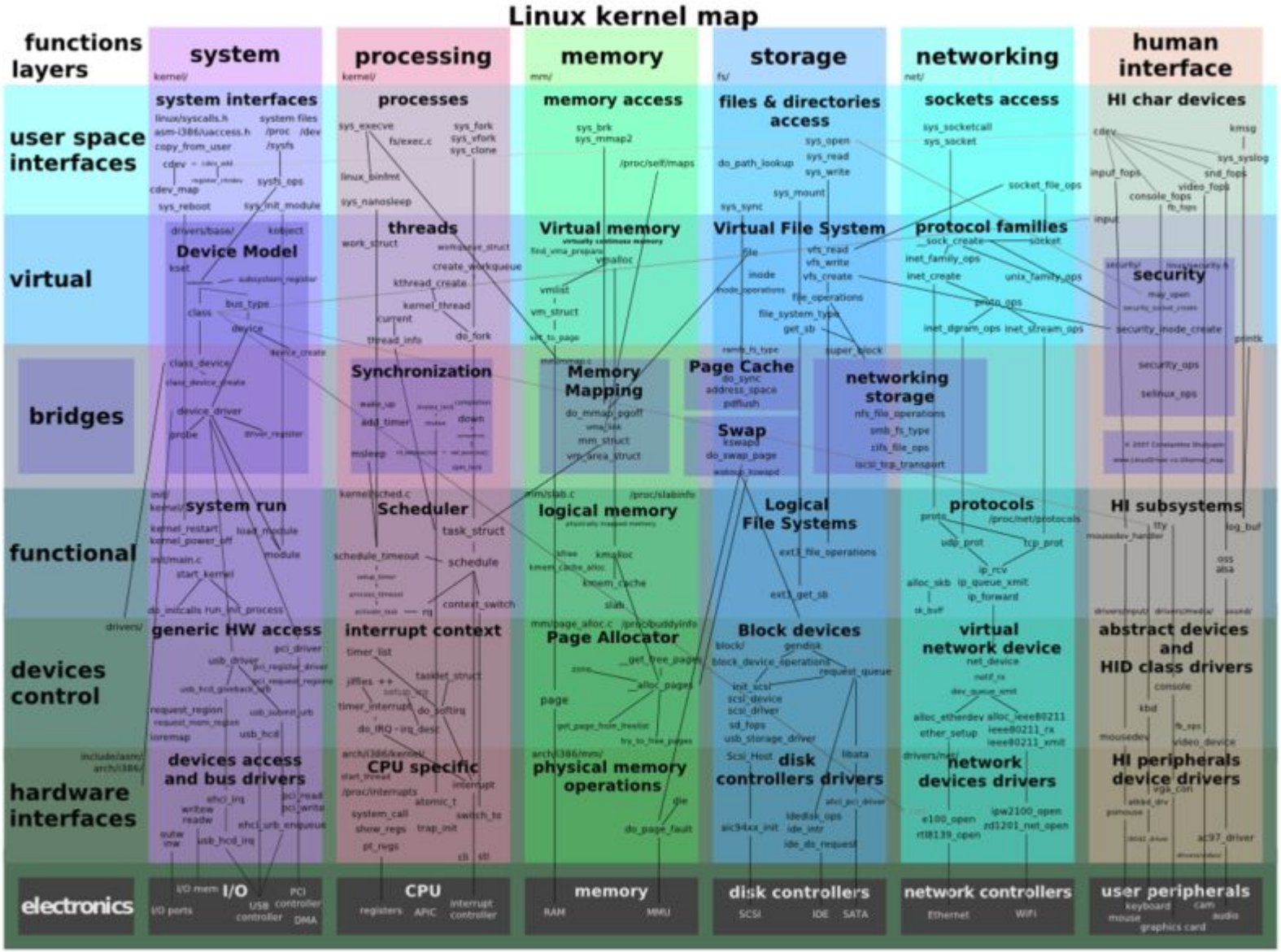
Ejecutando la llamada al sistema write()



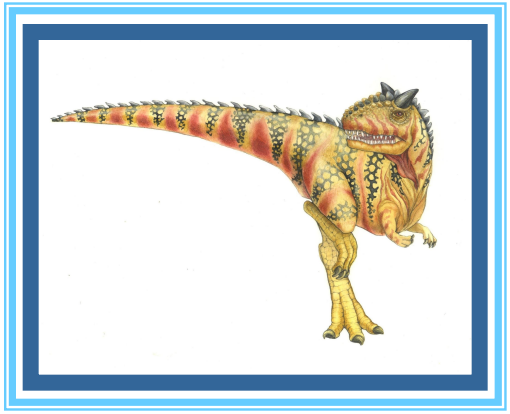
Estructura de GNU/Linux



Estructura de GNU/Linux

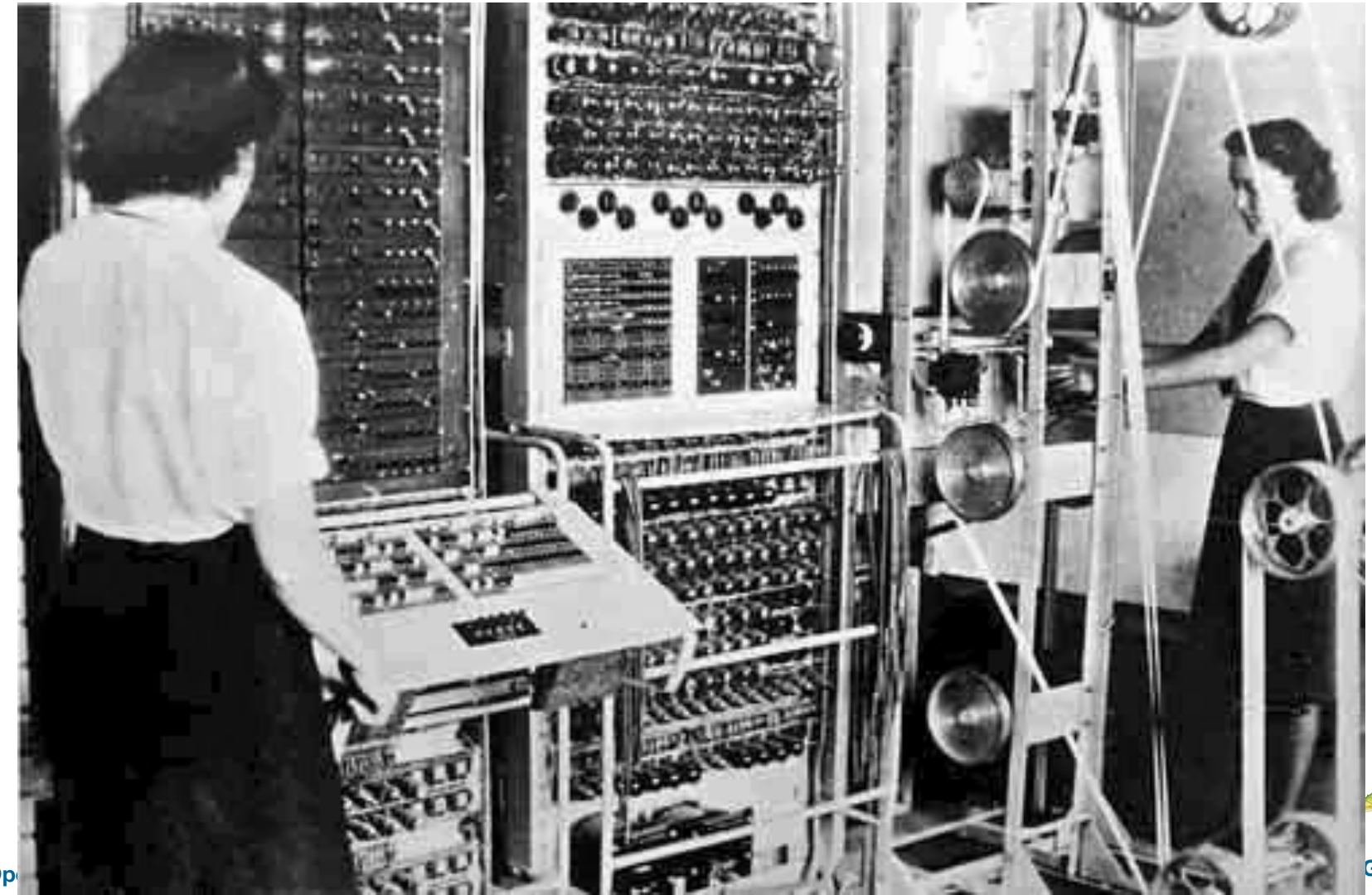


Fin de Unidad 1

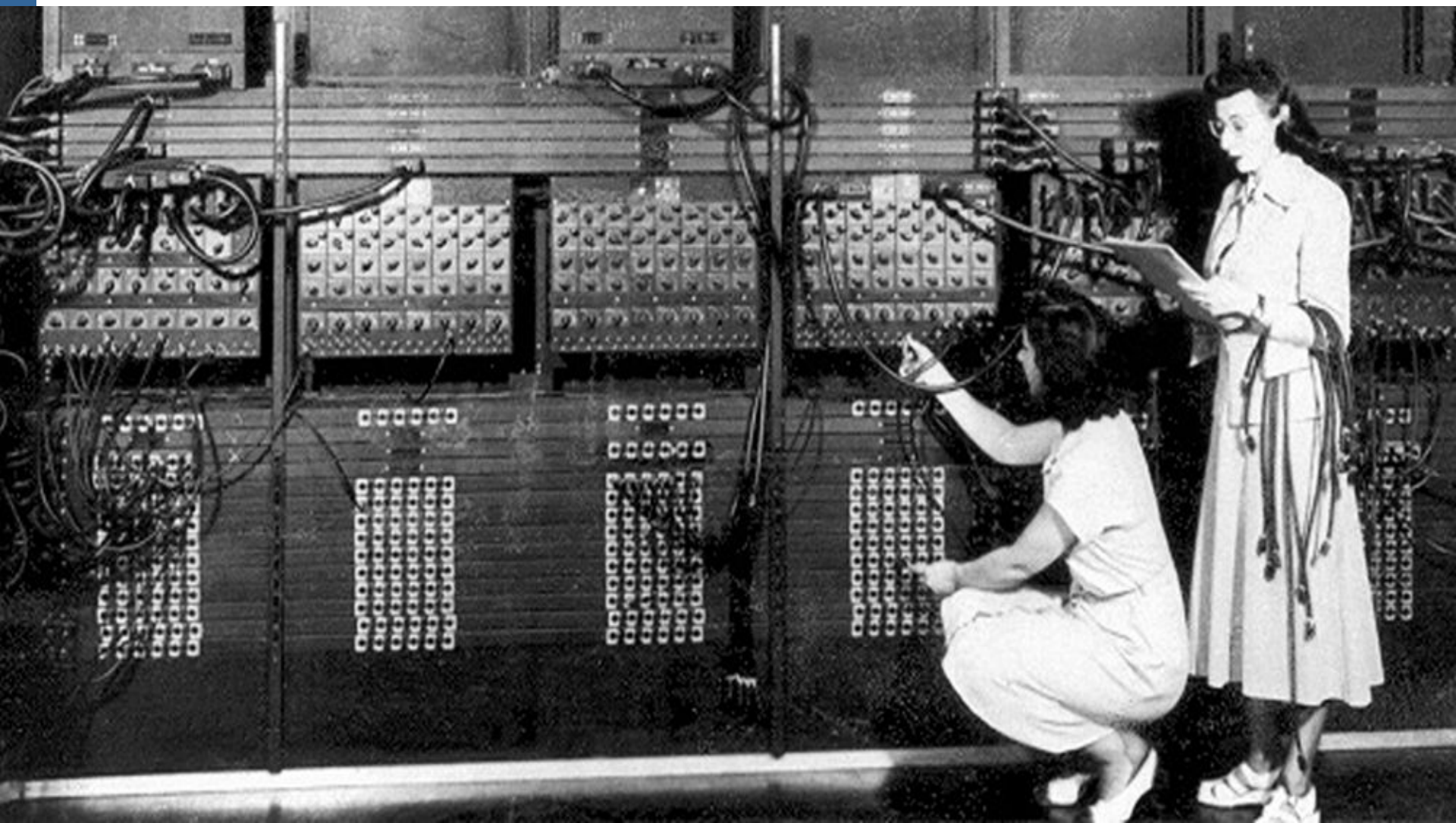


Unidad 1: Anexo - Historia de los Sistemas Operativos

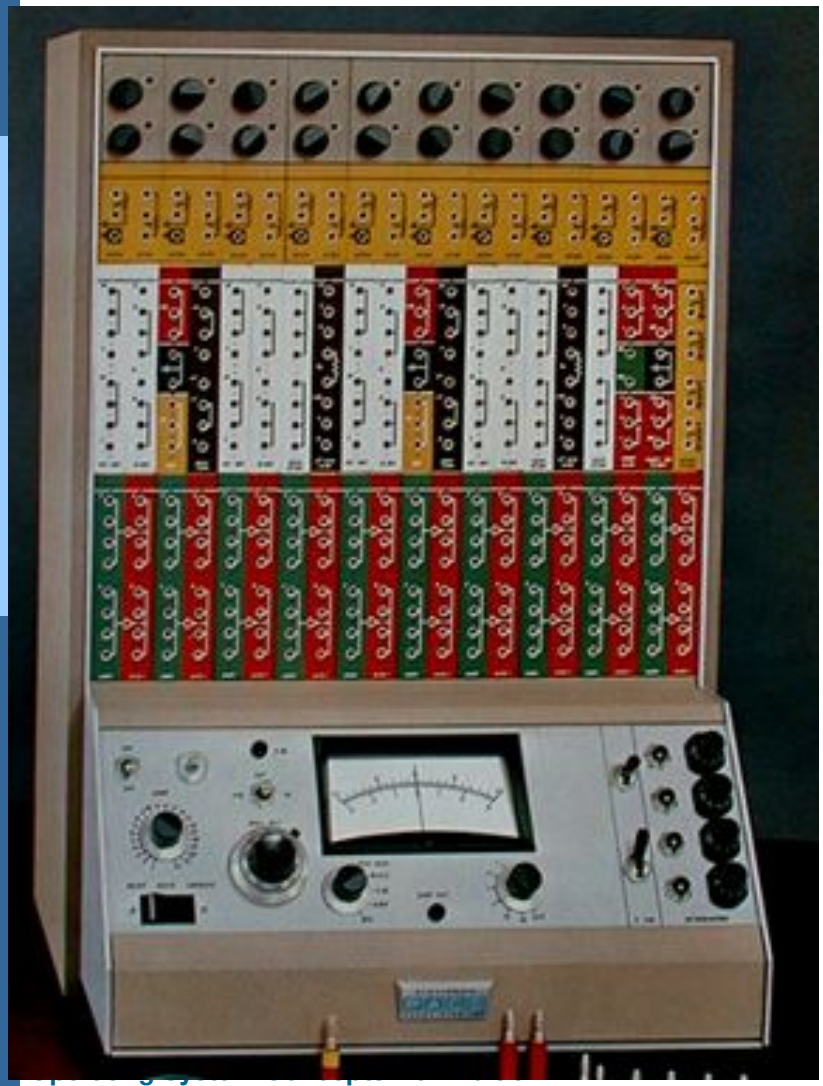
(1944) Colossus, programación por hardware, usada para Criptografía, no hay SO



(1946) ENIAC, primera computadora de propósito general, programación con cables! no hay SO

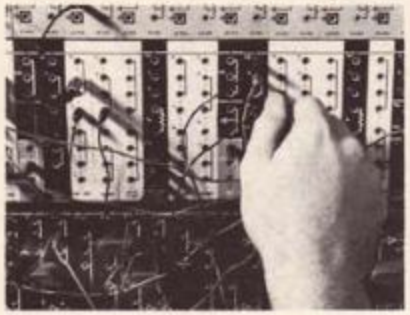


(1959) Computadora analógica TR-10 continúa la programación con cables! no hay SO



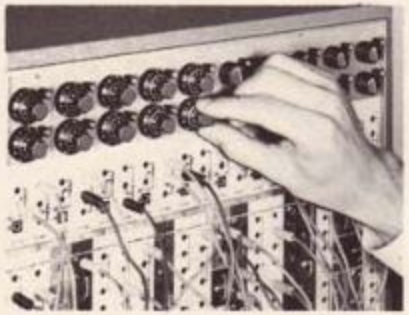
STEP 2 — PROGRAMMING

An information flow sheet is prepared using a block diagram to represent the various computer elements and their interconnections.



STEP 3 — PATCHING

Following this diagram, the patch cord connections are made between the various computer elements.



STEP 4 — INSERTION OF PROBLEM PARAMETERS

Coefficient potentiometers are adjusted to provide design parameter inputs.

STEP 5 — SOLUTION

The computer solution is performed in the exact manner prescribed by the mathematical equations. Solutions are presented on an XY plotter, strip chart recorder or on an oscillograph.

(1957) Fortran, primer lenguaje de alto nivel
(1958) Lisp, segundo lenguaje de alto nivel

(1961) Burrough MCP, primer SO escrito en lenguaje de alto nivel (una variante de ALGOL-60)

(1964) IBM OS/360, ocupaba 64KB

ahora una captura de pantalla ocupa 509 KB!

(1969) SO Multics, desarrollado por MIT, General Electric y Laboratorios Bell, escrito en PL/1

- incorpora multiprogramación

multiprogramación = múltiples procesos

Historia - la era Unix

(1971) Unix, escrito por Thompson, Ritchie y Mcllroy en Laboratorios Bell

hecho en lenguaje C (creado por Ritchie)

- estándar POSIX (interfaz mínima de llamadas al sistema)

Historia - la guerra de los SO

(1973) primera GUI (ventanas), por Xerox-Alto

(1976-1977) Apple I y II

Jobs y Wozniak los fundadores de Apple, se
les ocurrió la entonces extraña idea de vender computadoras para uso doméstico, como si fueran licuadoras o radios.

(1977) BSD, Berkeley Software Distribution (unix-like)

(1981) MS-DOS (escrito en ensamblador Intel 8088)
de Bill Gates

Bill Gates y Paul Allen, se les ocurrió una idea todavía más extraña y fantasiosa: vender sistemas operativos. Esto era mucho más extraño que la idea de Jobs y Wozniak. Un ordenador por lo menos tenía cierta realidad física. Un sistema operativo no tenía ninguna encarnación tangible

(1983) Proyecto GNU
de Richard Stallman

Software Libre

Puede ser estudiado, modificado y distribuido con libertad

vs

Software Privativo

Priva de la libertad a sus usuarios quienes no disponen del código fuente, por lo tanto no pueden cambiarlo ni investigar lo que realmente el software está haciendo

Software Libre

Un software es libre si satisface las 4 libertades esenciales:

Libertad 0

De ejecutar el programa con cualquier propósito

Libertad 1

De estudiarlo y modificarlo.

Libertad 2

De copiar el programa y distribuirlo.

Libertad 3

De mejorar el programa y hacer públicas las mejoras.

https://www.gnu.org/philosophy/free-sw.es.html

¿Por qué GNU?

GNU = "**G**NU is **N**ot **U**nix"

También sucede que GNU es la palabra más cargada de humor en el idioma inglés:

¿Por qué GNU?

GNU = "**G**NU is **N**ot **U**nix"

GNU

En el siguiente video el mismo Stallman cuenta cómo inició el proyecto GNU, en español!

Mientras tanto... la guerra de los SO continuaba

(1984) Mac OS (populariza GUI) por Steve Jobs

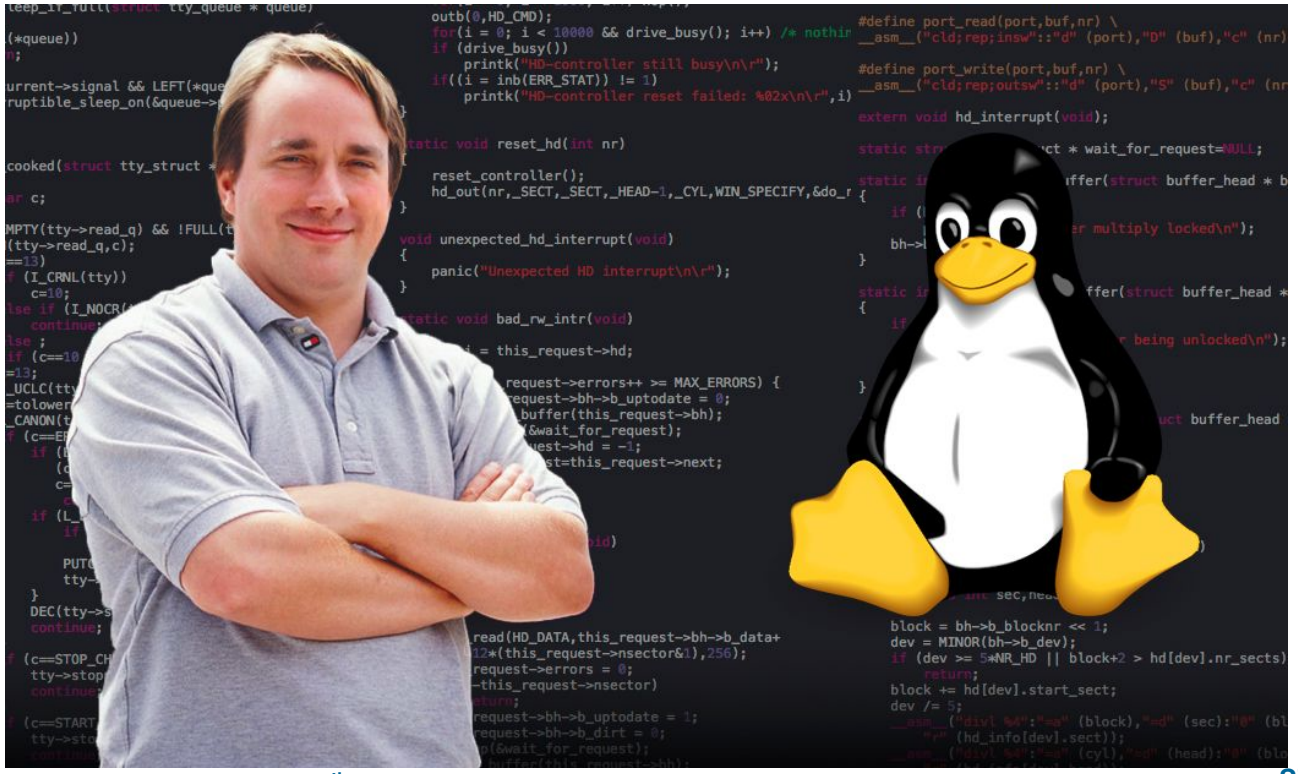
(1985) Microsoft Windows 1.0

(1987) Minix por Andrew Tanenbaum

- Unix modularizado y apto para la enseñanza

(1991) kernel Linux, por Linus Torvalds

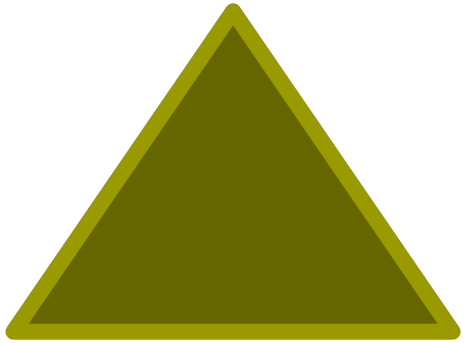
(basado en minix y en las herramientas del GNU)



ver descendientes de unix

El fenómeno GNU/Linux

Gates: produjo el mercado de hardware barato, muchas pcs en muchos hogares.



Stallman: Inició el movimiento de software libre y programó las bases del proyecto GNU

Torvalds: basado en la herramientas desarrolladas en el proyecto GNU creó el kernel Linux

Quite cualquiera de los 3 y GNU/Linux nunca hubiera existido!

(1993) FreeBSD (Berkeley Software Distribution)

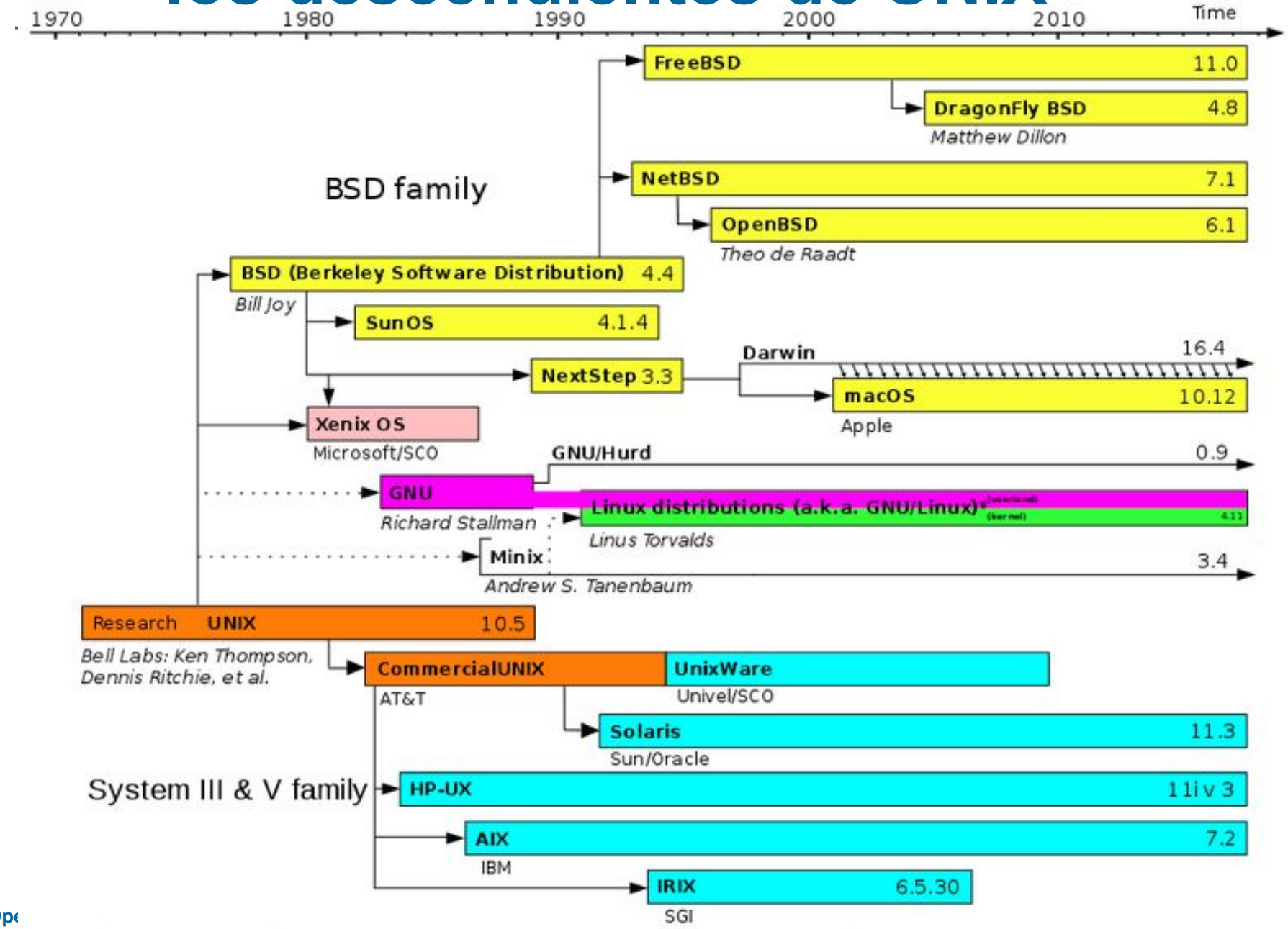
(1993) Debian (distribución GNU/Linux)

(2004) Ubuntu (distribución GNU/Linux)

(2005) Alpine Linux (usado en docker)

(2008) Android, por Google

los descendientes de UNIX



Bibliografía

- *Fundamentos de Sistemas Operativos (7ma Edicion)*
  A Silberschatz, PB Galvin, G Gagne, A Silberschatz
  - Capítulo 1

- *Página de la FSF (Free Software Foundation)*
  http://www.fsf.org/campaigns/priority-projects/

- *In the beginning was the command line*
  Neal Stephenson
  http://cristal.inria.fr/~weis/info/commandline.html