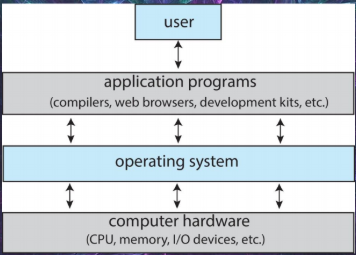
**Guía 1era Evaluación**

**Componentes de un sistema Operativo**

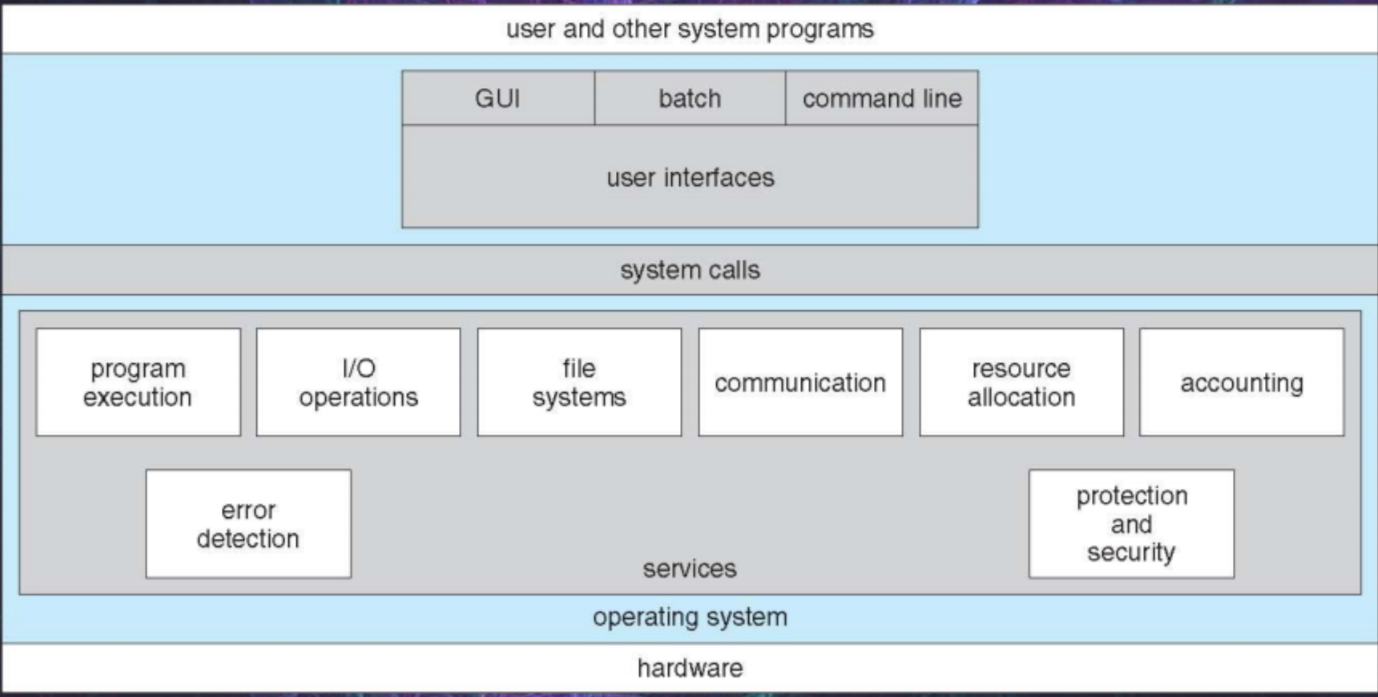


Driver: traduce el funcionamiento del hardware

**Sistema Operativo**

1. Es un asignador de recursos
   1. Gestiona los recursos de: CPU, memoria, dispositivos I/O
   2. Decide entre peticiones contradictorias para el uso eficiente y justo del recurso.
2. Programa de control
3. Controla la ejecución de programas para prevenir errores y evitar el uso indebido de la computadora
4. Intermediario entre el usuario y el hardware
5. El kernel es el programa que está corriendo todo el tiempo, los demás son un programa de sistema(Task manager) o aplicación

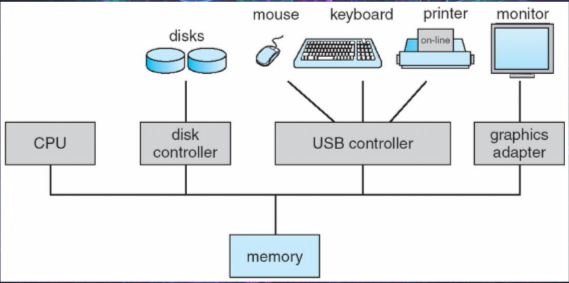
**Servicios de un SO**



**BootStrap**: programa cargado cuando la computadora inicia.

Es el encargado de iniciar todos los aspectos del SO, incluyendo cargar el kernel e iniciar su ejecución

**Organización de un SO**



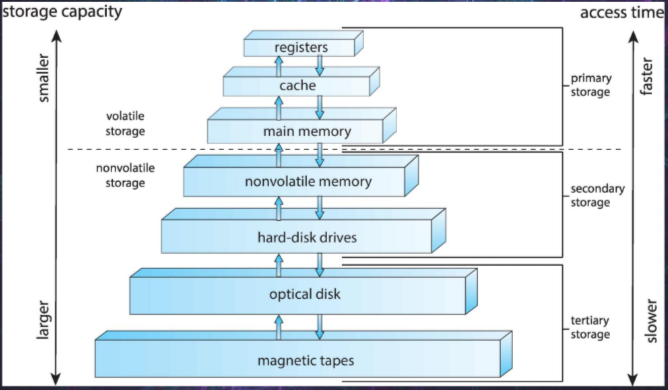
**Interrupciones**

Transfieren el control a una función de servicio. El vector de interrupción contiene las direcciones de éstas funciones.

**Trap o excepción**

Interrupción generada vía software (error o una petición).

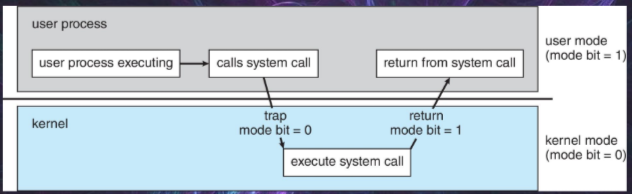
**Jerarquía de Almacenamiento**



**Modo Dual**

El SO cuenta con dos modos de operación como mecanismo de protección: usuario y kernel

Las llamadas se sistema permiten al usuario obtener servicios del SO



**Administración de procesos**

* Un proceso es un programa en ejecución
* Un programa es una entidad pasiva, un proceso es una entidad activa. Los procesos necesitan recursos (CPU, memoria, I/O)
* El SO se encarga de:
* Crear y eliminar procesos
* Suspender y continuar procesos

**Administración de memoria**

* Saber que partes de la memoria están siendo utilizadas y por quién
* Decidir qué proceso y datos entran y salen de la memoria
* Asignar o quitar espacio de memoria

**Administración de almacenamiento**

* Archivo: unidad lógica de almacenamiento
* Puede:
* Crear y borrar archivos y directorios
* Manipular archivos y directorios
* Mapear archivos a almacenamiento secundario
* Determinar quién tiene acceso a un archivo

**Seguridad**

* Protección: cualquier mecanismo que controle el acceso de procesos y usuarios a los recursos definidos por el SO
* Seguridad: defensa del sistema frente a ataques internos y externos

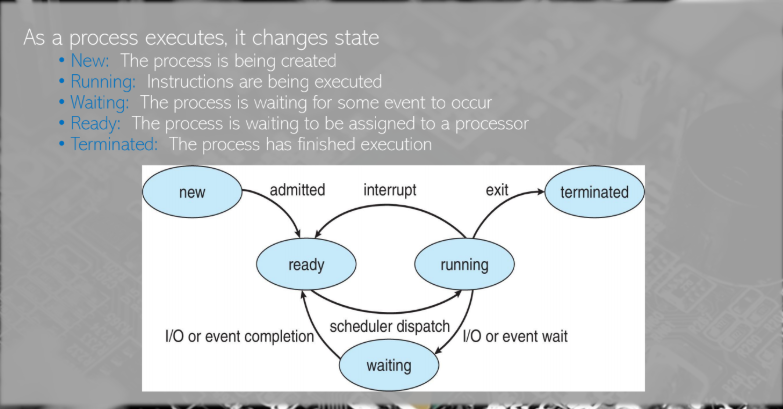
**Procesos**

* Un programa es un conjunto de instrucciones que le dicen a la computadora como hacer algo, usualmente en un contenedor secundario. Es una entidad pasiva almacenada en un disco(archivo ejecutable).
* Cuando un programa es activo y cargado en memoria, se le conoce como un proceso.
* Este programa se convierte en proceso cuando un archivo ejecutable es cargado en memoria
* La ejecución de un programa empieza vía GUI, mouse click, line de comando. Un program puede ser varios procesos

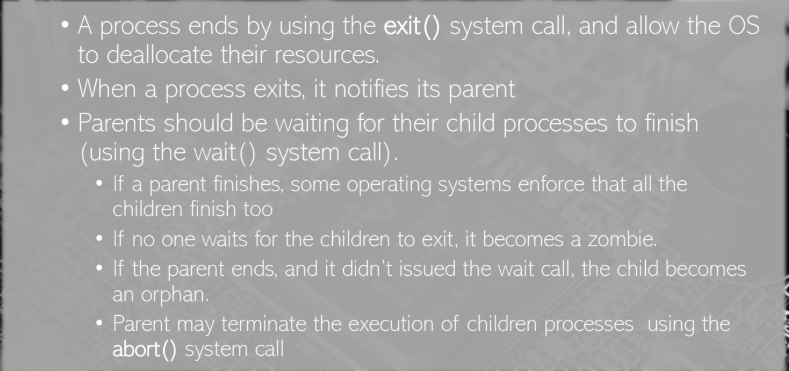
**Un proceso en memoria**

* Text: el código del programa
* Data: Información global o inicializada
* Stack: Información temporal(function parameters, return address, local variables)
* Heap: Asignación de memoria dinámica(Memory allocated dynamically)

**Diagrama de estados de procesos**



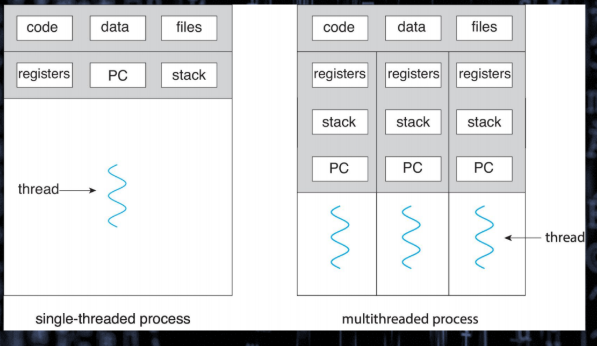
**Terminación de procesos**



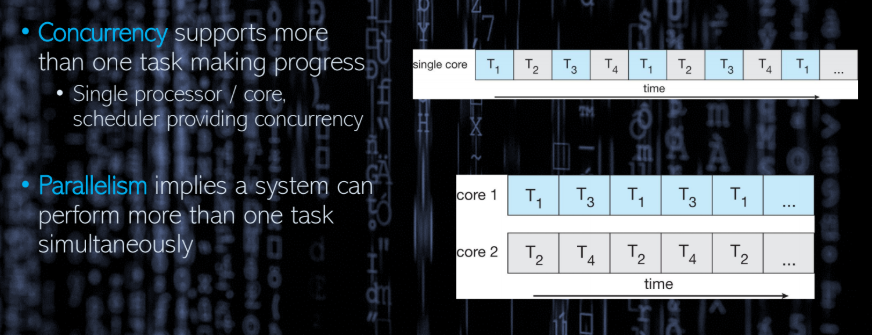
**Threads**

* Casi todas las aplicaciones modernas utilizan multithreaded
* Múltiples tareas con la aplicación pueden ser implementadas por threads separados
* Actualizar el display
* Responder una solicitud de internet
* La creación de threads es ligera

**Single and Multithreaded Processes**

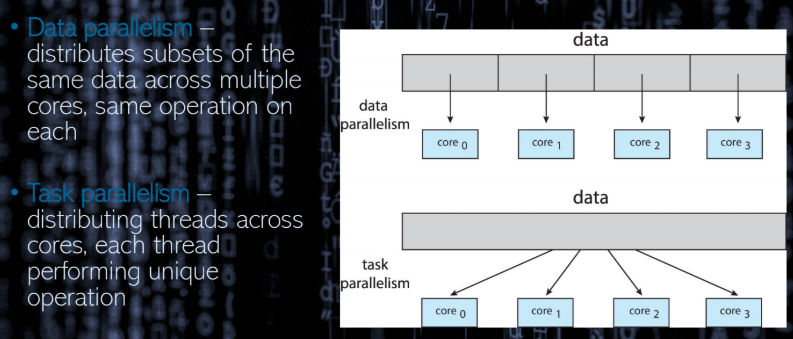


**Concurrencia y Paralelismo**

****

**Tipos de paralelismo**

* Paralelismo de datos
* Paralelismo de tareas



**Tipos de threads**

**User thread:** administración hecha por nivel de usuario en la librería de threads

**Tres librerías de thread primarias:**

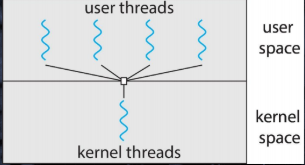
* POSIX Pthread
* Windows threads
* Java threads

**Kernel Thread:** Soportado por Kernel

**Modelos Multithreading**

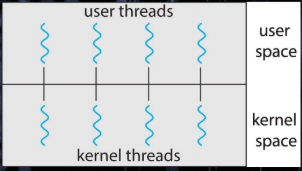
**Muchos a uno:**

* Un thread de kernel a muchos threads de usuario
* Crea un cuello de botella si la llamada de sistema tiene que esperar
* No es utilizado



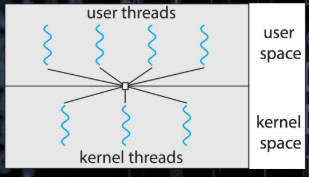
**Uno a muchos:**

* Por cada thread de usuario, un thread de kernel es creado
* Permite gran concurrencia a todos los modelos thread
* Número de thread puede estar limitado por el sistema operativo

Así es como Linux y Windows trabajan

**Muchos a muchos:**

* Pool(Estanque) de threads de kernel disponibles que son compartidos
* Si un proceso necesita un nuevo thread the sistema y no hay uno disponible, entra en una fila de espera
* No es utilizado



**Estrategias Thread**

* **Asynchronous threading**
  + Una vez que el padre crea al thread hijo, el padre resume su ejecución, para que el padre y el hijo ejecuten concurrencia e independientemente uno del otro.
  + Comparten pocos datos
  + Comúnmente utilizado para diseñar interfaces de usuario
* **Synchronous threading**
  + El padre crea uno o más hijos y luego debe esperar por todos sus hijos a terminar antes de que reanude. Los threads hijos hacen trabajo concurrente
  + Comparten muchos datos
  + Threads padres pueden combinar los resultados calculados por sus hijos.

**Cancelación de threads**

Finalizar un thread antes de que termine

Dos aproximaciones generales:

* **Asynchronous cancellation** termina el target thread inmediatamente
* **Deferred cancellation** permite al target thread que periódicamente cheque si debe ser cancelado

Pablo García Morales A01745530