Tema 1: Introducción a los sistemas operativos

-Definiciones

-**Un S.O. es un conjunto de programas que controla la ejecución y actúa como interfaz entre el usuario y el hardware del computador**. Es un programa que tiene como objetivos simplificar el manejo y el uso de la computadora, haciéndolo seguro y eficiente.

-**Los programas de usuario:** utilizan los servicios proporcionados por el S.O. para tener acceso a los recursos.

-Objetivos:

- -**Comodidad:** hace que sea más fácil de usar el computador.
- **-Eficiencia:** El S.O. utiliza los recursos de la forma más eficiente posible.
- **-Seguridad:** Asignación controlada de dispositivos E/S.
- **-Capacidad de evolución:** Un S.O. debe poder ampliarse fácilmente.

-El SO puede ser visto como: el interfaz entre usuario y la máquina o el gestor de recursos.

-Usuarios y Grupos

- Un usuario es una persona autorizada a usar el sistema. Se autentica con **usuario y password.**
- El S.O. no asocia el usuario con persona física, si no con **Cuenta.**
- Cada usuario tiene asociado un **identificador (UID) y un perfil de usuario.**
- Cada usuario y perfil tienen **asociados unos permisos.**
- **Existe un "User (root)" sin restricción alguna.**
- **Los usuarios se organizan en grupos.**
    - **Todo usuario debe pertenecer a un grupo.**
        - **Los grupos tienen privilegios.**

Estructura de los S.O.

-Estructura de un sistema:

-Estructura de un sistema de información:

- **-Hardware (CPU, memoria, dispositivos E/S).**
- **-Sistema Operativo.**
- **-Utilidades y Librerías.**
- **-Aplicaciones de usuario.**
- **-Usuarios (personas y otros computadores).**

-Núcleo (Kernel):

- **-Parte fundamental del SO que está ejecutando permanente y lo hace en modo root.**
- -Responsable de **facilitar el acceso seguro al hardware.**
- -**Gestiona los recursos**
- -**Multiplexa el acceso al hardware y recursos.**
- -Implementa capas de abstracción del hardware.

-Componentes del so:

-El núcleo (Kernel) interactúa directamente con el hardware:

- **-Gestión del procesador:**
  - Tratamiento de interrupciones.
  - Funciones básicas de memoria.
  - DMA.
- **-Gestión de procesos:**
  - Creación.
  - Planificación.
  - Destrucción de procesos.
- **-Gestión de memoria:**
  - Mantenimiento del mapa de memoria libre
  - Asignación
  - Liberación.
- **-Gestión de E/S:**
  - Acceso a los registros y protocolos de comunicación con periféricos
- **-Gestión de archivos y directorios:**
  - Manejar archivos y directorios.
  - Administrar el sistema secundario
- **-Seguridad y protección:**
  - Identidad de los “Users”.
  - Privilegios.
  - Violaciones.
- **-Comunicación y sincronización de procesos:**
  - Ofrece mecanismos para que los procesos puedan comunicarse y sincronizarse
- **-Interfaz de llamadas al sistema:**
  - Son capas de abstracción
  - Incluyen programas que encapsulan las llamadas
  - Utilidades del sistema
- **-Intérpretes de comandos**

-Estructura de los S.O.:

-Pueden ser monolíticos, estructurados (por capas o cliente-servidor) o máquinas virtuales.

-Sistemas monolíticos:

- -**No tienen estructura definida**.
- **-Un único programa.**
- -Todas sus funciones son en modo Kernel
- -Procedimientos y funciones con interfaces bien definidas.
- **-Libertad de llamada entre ellos.**
- **-No hay ocultación de información**, se ven entre ellos y sus datos.
- -Dentro de la no estructura existen procedimientos de servicio y de utilidades:
    - **-Programa principal llama a una rutina de servicio.**
    - **-Conjunto de rutinas de servicio.**
    - **-Conjunto de rutinas de utilidades que apoyan a las de servicio.**
- -Para cada llamada hay una rutina de servicio.
- -Problemas con la evolución y la programación.

-Sistemas estructurados por capas:

- -**Cada capa ofrece servicios y utiliza servicios de otras capas.** Solo conocen la interfaz.
- -No se conocen detalles de implementación.
- **-Modularidad y ocultación**
- **-Desarrollo y mantenimiento separado por capas**
- **-Fácil depuración.**

-Sistemas estructurados cliente/servidor:

- **-Solo una pequeña parte es el microKernel.**
- **-El resto de los servicios se implementa en modo usuario.**
- -Son los servidores
- -MicroKernel, básicamente contiene:
    - **Gestión de interrupciones**
    - **Gestión de memoria básica**
    - **Gestión básica de procesos**
    - **Gestión de servicios de comunicación entre procesos**

- -Los procesos solicitan servicio a los servidores, y estos entre ellos si se requiere.
- -Los servidores utilizan servicios del microKernel.
- **-Ventajas: flexibilidad, funcionalidades manejables, facilita desarrollo y depuración, si falla un servicio no se detiene todo el sistema, adaptabilidad a sistemas distribuidos.**
- **-Desventajas: mayor sobrecargas y tiempo de servicio. Competencia por recursos.**

-Sistemas de máquinas virtuales:

- **-El núcleo del sistema es el monitor de máquina virtual**
- -Sobre el monitor se implementan las máquinas virtuales. Replicas lógicas del hardware.
- **-En cada máquina virtual se puede ejecutar un SO con diferentes características**

Subsistemas

-Subsistema: Procesos:

- -Es el subsistema principal. Todo gira alrededor del concepto de **proceso.**
- **-El proceso es:**
  - **-Un programa en ejecución**
  - **-Una instancia de un programa funcionando en un computador.**
  - **-La entidad que puede ser asignada al procesador y ejecutada por él.**
  - **-Una unidad de actividad caracterizada por un tratamiento de ejecución secuencial, un estado y recursos asociados a la misma.**
- -El subsistema **se encarga de generar y gestionar los procesos** y atender sus peticiones.
- -**Para ejecutar un programa tiene que estar en memoria**
  - -Imagen en memoria-Código-Datos
  - -Estado del procesador-Parte del PCB (*el PCB es un registro que guarda descripción básica del proceso*)
- **-Gestión de las estructuras y servicios que conforman el proceso:**
  - **-PCB, Servicio de gestión de procesos y resolución de conflictos.**
- **-El proceso se trata como una estructura de datos que está en un estado determinado**

- -**Servicios de procesos:**
  - **-Creación de procesos:**
    - **-Proceso padre crea procesos hijos.**
    - -Creación:
      - **-A partir del proceso padre (UNIX)**
      - **-A partir de un archivo ejecutable (Windows)**
  - **-Ejecución de procesos:**
    - **-Ejecución Batch ("background"):**
      - -El proceso no tiene asociado terminal.
      - -E/S de y hacia archivos.
    - **-Ejecución interactiva:**
      - **-El proceso está asociado a un terminal.**
    - **-Terminación de procesos:**
      - -Termina la ejecución del programa.
      - **-Se produce una condición de error.**
      - **-Otro proceso, el SO o el usuario lo terminan.**
    - **-Cambio de programa del proceso:**
      - **-Cambia el código que está ejecutando por el de otro programa almacenado en disco, normalmente a petición propia.**
      - **-Sustituye el código que el proceso ejecuta.**

- -Un proceso puede dividir su trabajo (**relaciones entre padre-hijo**)
- **-El subsistema proporciona servicios de comunicación y sincronización locales y remotos**
- **-Comunicación:**
  - ***-Síncrona:*** Espera en el servicio
  - ***-Asíncrona*****:** Fuerza al SO a gestionar el almacenamiento temporal
- **-Los mecanismos de comunicación tienen entidad propia:**
  - -Se crean, se utilizan y se liberan.

-Planificación de Procesos y Asignación de Recursos:

- **-Las colas de procesos permite la planificación y asignación**
- *-Corto plazo:*
  - **-Mantiene procesos en memoria principal. Listos para ejecución.**
  - **-El planificador a corto plazo y el distribuidor se encargarán de llevar el proceso a ejecución.**
  - **-Round-Robin** o **Niveles de prioridad.**

- *-Largo plazo:*
    - -Nuevos trabajos en espera de usar el procesador
    - -**Se encarga de balancear la carga del sistema y pasarlos a corto plazo**
    - -Al entrar en corto plazo **se asigna memoria al proceso.**

- *-Colas de E/S:*
    - **-Hay una para cada dispositivo gestionadas por el S.O.**
    - **-Los procesos en estas colas están bloqueados o suspendidos.**

- **-El S.O. toma el control cuando se produce una interrupción.**

-Activación del S.O.:

- *-Secuencia normal:*
    - -Un proceso A solicita servicio del S.O. -> Se salva el estado del proceso A -> Se realiza la función pedida -> Se ejecuta el planificador -> Se ejecuta el Activador -> Se ejecuta el proceso B.

- *-Solicitud por una de estas vías:*
    - -**Llamadas al sistema. Interrupciones producidas por los periféricos o hardware. Condiciones de excepción o error del hardware.**

- *-Una función de librería que llama al sistema suele componerse de:*
    - -Una parte inicial de preparación de parámetros del servicio. Una instrucción TRAP. Una parte final que recupera los parámetros de contestación del S.O. para devolverlos al programa que la llamó.

-Subsistema: Memoria:

- -**Asignación de memoria al gestor de procesos para crear la imagen del proceso.**
- -Garantizar el **acceso correcto** a las zonas de memoria:
    - -**Garantizar que ningún otro proceso viole la memoria de otro**
    - **-Permitir acceder a distintas zonas de memoria en distintos modos**

- -Proporcionar memoria a los procesos que lo soliciten y gestionar su liberación, tanto para datos como para código:
    - -**Aumentar el tamaño de la asignación del proceso, datos o código**
    - -Siempre que haya memoria disponible devuelve un puntero a la nueva zona asignada
    - -**Disminuir el tamaño de la asignación del proceso y marcar la zona como disponible**

- -Tratamiento de los errores de acceso a memoria
- *-Gestión de la compartición de memoria:*
    - **-Ofrece servicios para asignar segmentos de memoria compartida a varios procesos cooperantes**

- *-Gestionar la memoria virtual:*
    - -**Permitir direccionar más memoria de la que realmente hay disponible de manera transparente para el programador**
    - -Permite a los programas direccionar la memoria desde un punto de vista lógico
    - **-Permite la coexistencia en memoria de varios procesos.** Así no existirá un tiempo muerto entre la ejecución de procesos sucesivos, puesto que el siguiente proceso planificado **ya reside en memoria**.

- *-Sistema de paginación:*
    - -Permite que los procesos estén formados por varios bloques de tamaño fijo, denominados páginas.
    - -La dirección virtual está formada por un número de página y un desplazamiento dentro de la página.
    - -**Cada página puede estar ubicada en un lugar cualquiera de la memoria principal.**
    - -La dirección real o dirección física se utiliza en la memoria principal.

-Subsistema: E/S:
- -Controlar el funcionamiento de todos los dispositivos de E/S.
- -Facilitar su manejo.
- -Ofrece servicios independientes del dispositivo.

-Subsistema: Archivos y Directorios:
- -Servidor de archivos:
    - *-Es la parte del S.O. encargada de:*
        - **-Facilitar el manejo de dispositivos periféricos.**
        - **-Proteger y limitar los archivos que cada usuario es capaz de manipular.**
    - -Los servicios son de dos tipos:
        - **-Orientados al manejo de datos o archivos.**
        - **-Orientados al manejo de nombres o directorios.**
    - -Ofrece una Visión lógica (*archivos y directorios identificados por nombre lógico*) y una física (*detalles de cómo están almacenados los objetos en los periféricos*).
    - -**Unidad de almacenamiento lógico no volátil.**
    - -El servidor tiene información asociada al archivo para el uso propio y de los usuarios.
    - **-Operaciones sobre la visión lógica del archivo**
    - **-Operaciones sobre la visión física del archivo**

- -Gestionar la información lógica y física

- -Servicio de Directorios:
  - **-Relaciona un nombre con un archivo.**
  - **-Identifica y ubica los archivos.**
  - **-Un mismo nombre no puede referenciar a varios archivos.**
  - **-Un archivo no puede tener varios nombres.**

- **-Visión lógica: esquema jerárquico de directorios y archivos.**
- **-Visión física: tablas de i-nodes, tabla FAT o NTFS, relaciones nombres-descriptores**

- -Sistema de archivos:
  - **-Conjunto de archivos incluidos en una unidad de almacenamiento**
  - **-Compuesta por los datos de los archivos y la información auxiliar para gestionarlos.**
  - -La metainformación del sistema de archivos está compuesta por:
    - **-Estructura física de los archivos**
    - **-Directorios**
    - **-Estructura física del sistema de archivos**
    - **-estructura de información de bloques e i-nodos libres**
  - -Cada S.O. organiza las particiones de disco de una determinada manera para repartir el espacio entre:
    - **-El programa de carga.**
    - **-la metainformación.**
    - **-los datos.**
    - **-el fichero de intercambio**.

Interfaz del programador

-POSIX:

- -**Es el estándar para sistemas operativos portables basado en UNIX**
- -No define la implementación solo la definición del estándar
- -**Los distintos S.O. pueden ofrecer lo mismo programado de distinta forma**
- -Es una familia de estándares en evolución
- **-Se encuentra en todas las versiones actuales de UNIX y LINUX. También disponible para Windows.**
- -Algunos estándares:
  - *-1003.1: Servicios básicos del S.O.*
  - *-1003.1a: Extensiones a los servicios básicos*

- *-1003.1b: Extensiones de tiempo real*
- *-1003.1c: Extensiones de procesos ligeros(threads)*
- *-1003.1d: Extensiones adicionales de tiempo real*
- *-1003.1e: Seguridad*
- *-1003.2: Shell y utilidades*
- *-1003.2b: Utilidades adicionales*

- -Algunas características:
  - **-Hay tipos de datos definidos en el estándar, otros se dejan al implementador, por ejemplo, en <sys/types.h> Estos acaban con _t.**
  - **-Nombres de funciones cortos**

- **-Win32:**
  - **-Define los servicios ofrecidos por Windows 95/98, Windows NT y Windows 2000.**
  - **-No es un estándar. Es una definición comercial de Microsoft**
  - -El API es totalmente diferente al de POSIX.
    - **-prácticamente todos los recursos se tratan como objetos que se referencian mediante handlers.**
    - **-Los nombres son largos y descriptivos.**
    - **-Tiene una serie de datos predefinidos.**
    - **-Los nombres de las variables llevan el prefijo del tipo de datos.**
    - **-Las funciones devuelven true si la llamada se ejecuta con éxito y false si no**

Clasificación y evolución de los S.O.

- **-Clasificaciones:**
  - **-La evolución de los S.O. nos ha dejado una variedad de tipos de sistemas.**
    - **-De Mainframe. Sistemas Operativos propietarios:**
      - -Grandes máquinas en capacidad y tamaño.
      - Procesamiento por lotes, procesamiento de transacciones y tiempo compartido.
    - **-De Servidor. Unix, Linux, Windows Server**
      - -Servidores.
      - **-Computadoras personales muy grandes, incluso mainframes dedicados.**
      - Múltiples usuarios con diversos perfiles, compartición de recursos.
    - **-Multiprocesador y Distribuidos:**
      - -Computadoras paralelas.

- Multicomputadoras o multiprocesadores, son SO para servidor con variaciones en conectividad y comunicaciones.

- **-Computadora Personal.**
- **-De tiempo Real.**
- **-Integrados.**
- **-De tarjeta inteligente.**

- **-Evolución-1. En serie**
  - -Procesamiento en serie:
    - -**Usuario != operador.**
    - -El usuario escribía los programas en código máquina -> controla el HW directamente
    - **-Programas orientados a cálculo matemático**
    - -El operador era el encargado de insertar los trabajos de cada usuario de forma manual.
    - **-Cintas papel o tarjetas perforadas**
    - **-Problemas: planificación y tiempo de preparación**

- **-Evolución 2. Por lotes**
  - -**Usuario != operador**
  - **-Cintas magnéticas o tarjetas perforadas**
  - **-Programas escritos en ensamblador o en lenguajes de alto nivel**
  - -Se basaba en el uso de un monitor que gestionaba el problema de la planificación
  - -El tiempo de inicio se reducía para trabajos similares
  - *-El monitor se encarga de:*
    - **-Leer e interpretar tarjetas perforadas**
    - **-Cargar programas de sistema y de usuario en la memoria**
    - **-Comunicarse con los dispositivos de E/S**
  - *-Problemas:*
    - **-Depuración difícil**
    - **-Bajo rendimiento**
  - -*Solución*: **sistemas ejecutivos totalmente residentes.**
  - *-Los sistemas ejecutivos evolucionaron:*
    - **-Contabilización del uso del sistema**
    - **-Límites temporales de procesamiento**
    - **-Protección**
    - *-Inconvenientes:*
      - **-Secuencialidad estricta**
      - **-Mucho uso de CPU.**
    - -*Solución*: **procesadores auxiliares**

- **-Evolución-3. Multiprogramados:**
  - **-Multiprogramación: tener varios procesos en marcha para que el S.O. cambie a otro cuando el que ejecuta espere una operación de E/S**
  - -*Se comparte*: **Espacio de memoria y tiempo de procesador**
  - -**Todos los procesos están en memoria principal en diferentes estados de ejecución.**
  - -*Grado o factor de multiprogramación:*
    - **Pes = Probabilidad de un proceso en E/S**
  - -*Utilización procesador con 1 solo proceso:*
    - **Utilización = 1-Pes**
  - -*Utilización procesador con "n" procesos*
    - **Utilización= 1-(Pes)n**
  - -*Monoprogramación:*
    - -Antes de continuar, **el procesador debe esperar hasta que la instrucción de E/S termine**
  - -*Multiprogramados:*
    - -Se **comparte tiempo de procesador**
    - -*Usado para:*
      - **-Incrementar uso de CPU y recursos**
      - **-Soportar múltiples usuarios activos**
      - **-Permitir programas interactivos**

    - -*Se requiere:*
      - **-Gestión de recursos**
      - **-Planificación de procesos**

      -*Características Hardware que lo favorecen*:
      - **-Interrupciones E/S.**
      - **-Canales E/S autónomos.**

    - -*Tipos:*
      - -*Lotes multiprogramados*: mientras quede memoria se aceptan trabajos por lotes**. El S.O. decide cuál de los procesos en cola ejecuta. Se requiere mecanismos de protección**
      - -*Tiempo compartido:*
        - -Ejecución de varias tareas interactivas(shell)
        - **-Multitarea y multiusuario**
        - -El S.O. cede el control a otro proceso cuando:
          - -**Un proceso termina su ejecución.**
          - **-Un proceso necesita una operación E/S.**
          - **-El quantum se agota.**

- -*SO combinados:* no encaja en un solo molde
- -*Sistemas paralelos:*
  - -**Son sistemas fuertemente acoplados basados en computadores con varios procesadores comunicados**
  - -*Se comparten ciertos recursos:*
    - -Ventaja: **se aumenta el rendimiento, la fiabilidad y son económicos**
  - -*Multiproceso simétrico (SMP):*
    - -**Cada procesador ejecuta una copia idéntica del SO**
    - **-Ejecución de muchos procesos sin bajar rendimiento**
    - **-La mayoría de los SO actuales lo soportan**
  - -*Multiproceso asimétrico:*
    - **-Uno de los procesadores es el maestro y controla al resto**

- **-Evolución-4. Personales y red**
  - -*Sistemas de ordenador personal y red:*
    - -PC: **son sistemas de computadores dedicados a un único usuario, de bajo coste**
    - -Énfasis para facilitar su uso por el usuario y mejorar la interactividad
    - **-Incorporan interfaces gráficas y dispositivos E/S amigables**
  - -*Estaciones de trabajo en red:*
    - -Desarrollo de LAN.
    - -Servicios de comunicación integrados en el SO.
    - **-Computación cliente/servidor.**
    - -Estaciones sin disco.
  - -*Convergencia PC y estaciones en red.*
  - -*Los users son conscientes de la existencia de otras máquinas y recursos.*

Tema 2: Introducción a las llamadas al sistema

- **-La llamada al sistema es la forma en la cual un proceso requiere un servicio específico de núcleo**
- -El código de la llamada al sistema se puede ejecutar en:
  - -Modo kernel:
    - **-Tiene acceso completo a los recursos hardware.**
    - **-Gracias a esto el Kernel mantiene el control del sistema**
  - -Modo Usuario:
    - **-Acceso limitado a los recursos**

        - -Si el código intenta ejecutar una instrucción privilegiada el microprocesador avisa al Kernel para matar al proceso.
        - -Tiene que pedirle al Kernel que ejecute la operación por el:
            - -**Acceso a ficheros y a la red.**
            - **-Crear y destruir procesos.**
            - **-Apropiarse de más memoria.**

- **-Llamadas al sistema:**

    - -Casi todos los SO Unix-Like tienen las mismas llamadas al sistema.
    - -El mecanismo real de la llamada al sistema depende de la máquina y se realiza en ensamblador.
    - -Se ofrecen funciones c que encapsulan los detalles de la implementación de la llamada al sistema
    - -Antes de poder escribir Software para un sistema se necesitan conocer las APIs que ofrece. En el núcleo de todos ellos reside el API de las llamadas al sistema.

    - **-Tipos de llamadas al sistema:**
        - **-Control y gestión de procesos.**
        - **-Manipulación de archivos.**
        - **-Manipulación de dispositivos.**
        - **-Comunicaciones.**

- **-Llamadas al sistema:**

    - **-Permiten a los usuarios pedir servicios del kernel.**
    - -En C parecen funciones normales, no solo se transfiere el control de ejecución a la función, sino que se pasa al procesador en modo Kernel.
    - -Cualquier proceso linux se puede ver como un bucle que:
        - **-Realiza algún cálculo**
        - **-Hace una llamada al sistema**
        - **-Vuelve al paso 1**
    - -La mayoría de las llamadas tienen que ver con la carga del programa en si y de la inicialización de la Librería de C
    - **-Cuando una llamada falla, el sistema devuelve -1, que es una forma estándar del kernel de indicar ERROR.**
    - **-Los programas deben comprobar después de una llamada si todo es correcto.**

Tema 3: Procesos Estructuras y Estados

-Concepto de proceso:

- **-Es un programa en ejecución.**
- -**Es la entidad que se puede asignar al procesador.**
- -La entidad a la que se pueden asignar los recursos
- **-Se caracteriza por la ejecución de una secuencia de instrucciones, un estado y un conjunto de recursos asociados.**
- **-Es una entidad formada por: *código, datos, memoria, pila y PCB*.**
- -*Se caracteriza en un instante por*: **identificador (PID), Estado, Contador de programa, Punteros a memoria, Datos de contexto, Información de E/S, Información de auditoria y Punteros de enlace**
- -Toda esta información se almacena en su **PCB** (Process Control Block)
- -El S.O. lo crea y gestiona.
- **-Permite restaurar el proceso tras una interrupción.**
- -Da soporte a múltiples procesos

-Estructuras de control

- ***-Del S.O.:***
  - -El SO: controla los eventos dentro del computador, planifica y activa procesos, reserva recursos y responde a las solicitudes de servicio.
  - -El SO construye y mantiene tablas con información para cada entidad que gestiona. Estas tablas son:
    - ***-Tablas de Memoria:***
      - -Parte de la memoria reservada para el SO
      - -Con ellas, el SO mantiene un registro de:
        - **-Memoria principal**
        - **-Memoria virtual**
      - -*Incluye info relativa a:*
        - -Reservas de memoria principal por parte de qué procesos
        - -Reservas de memoria virtual por parte de qué procesos (Swap)
        - -Todos los atributos de protección, tanto principal como virtual
        - -Todos los atributos de compartición
        - -Información de ubicación y control de bloques de swap

        - -Reservas y asignación de memoria para los dispositivos.

- **-Tablas de Dispositivos:**
    - *-Con ellas el SO gestiona:*
        - **-Los dispositivos E/S.**
        - **-Canales de comunicación.**
        - -Canales **DMA** (Direct Memory Access).
    - *-Incluyen informativa relativa a:*
        - **-Dispositivos/canales libres.**
        - **-Asignados a qué proceso.**
        - -¿Operación de E/S en curso? ¿Estado?
        - -Direcciones de memoria involucradas.
        - **-Tasa y volumen de las transferencias.**

- **-Tablas de Ficheros:**
    - *-Incluyen información relativa a:*
        - -Existencia de ficheros
        - -Posición en almacenamiento secundario
        - -Estado actual
        - -Permisos

- **-Sobre las tablas:**
    - -Todas están entrelazadas y referenciadas mediante referencias cruzadas
    - -Las tablas residen en Memoria y son propiedad del S.O.
    - -El S.O. necesita información del hardware y recursos disponibles para crearlas
    - -El arranque del S.O. recopila esta información

-Representación física de un proceso:

- *-Un proceso debe incluir:*
    - -Un programa o conjunto de programas a ejecutar -> Código -> Reserva de memoria -> Punteros.
    - -Áreas de memoria donde ubicar sus variables y constantes -> Datos -> Reserva de memoria -> Punteros.
    - -Área de memoria donde ubicar la pila de llamadas a procedimientos -> Pila -> Reserva de memoria -> Punteros
    - -Área de memoria donde ubicar los atributos del proceso -> PCB -> Reserva de memoria -> Punteros

- *-Imagen del proceso:*
    - **-Stack:** pila de llamada a procedimientos LIFO gestiona parámetros, retornos y almacenamiento para variables locales a procedimientos.
    - **-Heap:** zona alojamiento dinámico.
    - **-Bss:** zona para variables estáticas no inicializadas.
    - **-Data**: zona para variables globales o estáticas inicializadas.
    - **-Text:** código ejecutable del proceso.

-Ubicación del proceso:

- -Un proceso no necesariamente debe estar completo en memoria.
- -Una vez creado:
    - -Caso más simple->Completo y contiguo en memoria->Se mantiene en memoria secundaria(Swap)
    - -Para que el SO lo pueda gestionar-> se carga en memoria principal, al menos una parte
    - -Para que se pueda ejecutar->Tiene que estar completo en memoria
- -El S.O. debe conocer las ubicaciones en disco, memoria virtual y memoria principal.
- -S.O. modernos tienen HW de Paginación.

-Atributos del proceso->PCB:

- -Toda la meta-información del proceso queda recogida en el PCB
- -Cada SO organiza el PCB a su modo.
- -Elementos:
- *-Identificación del proceso:*
    - **-Identificadores del proceso, del padre y del usuario**
- *-Estado del procesador:*
    - **-Registros visibles por el usuario**
    - **-Registros de estado y control**
    - **-Punteros de Pila**
- *-Información de Control de Proceso:*
    - **-Información de estado y planificación**
    - **-Estructuras de datos**
    - **-Comunicación entre procesos**
    - **-Privilegios del proceso**
    - **-Gestión de memoria**
    - **-Recursos**

-Rol del PCB:

- **-Es la estructura más importante del SO.**
- -**El conjunto de PCB definen el estado del sistema**

- **-Cada PCB contiene toda la info que el S.O. necesita**
- **-La info es leída y modificada en cualquier parte del SO**
- -Es necesaria la protección del PCB**. Un fallo en la rutina del S.O. puede dañar un PCB** y los afectados. Un cambio en la estructura o semántica del PCB obliga a rediseñar
- **-Es obligado el acceso al PCB a través de una única rutina o servicio manejador**
    - **-Sirve los datos y sincroniza su acceso**
    - **-Diseñada para un gran rendimiento**

-Estados de los Procesos:

- -Los procesos están en una cola para su ejecución -> en la de listos.
- -El planificador selecciona a uno de la cola -> varias políticas de selección.
- -El proceso es puesto en ejecución por el Activador del SO.
- *-El proceso se ejecuta hasta que:*
    - **-Termina** -> El proceso sale del sistema.
    - **-Necesita una operación E/S** -> Se bloquea hasta que termine la operación.
    - **-Necesita un recurso y no está disponible** -> se bloquea hasta que esté disponible.
    - -**Se le acaba el tiempo(quantum)** -> se pone a la cola de los que están Listos.
    - **-Ocurre algo inesperado, una interrupción** -> El SO atiende ese algo y le deja continuar o lo pone en la cola.

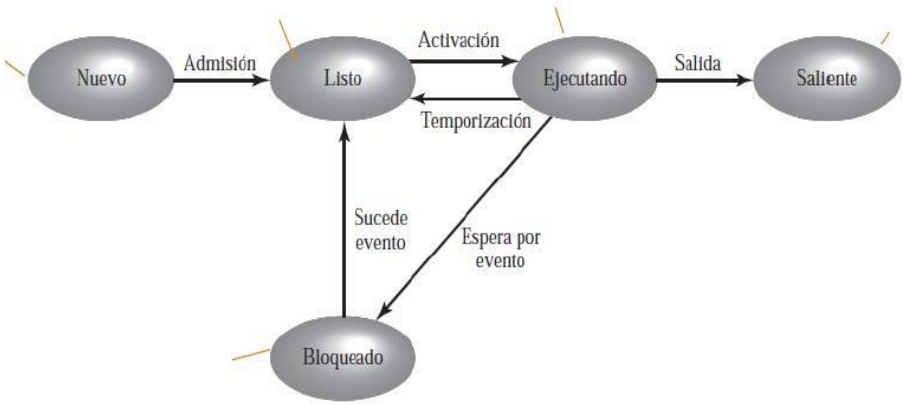
- **-Siempre que pasa algo el SO toma el control.**
- -Creación/Terminación:
    - *-Se va a añadir un nuevo proceso:*
        - -El S.O. crea y actualiza todas las estructuras de datos
        - -Crea la imagen del proceso
        - -Añade el proceso en la cola correspondiente

    - *-Eventos que provocan la creación de un proceso:*
        - **-Emisión de un nuevo proceso por lotes:**
            - -Existe un sistema de carga de trabajos y decide que puede comenzar con uno nuevo
        - -Sesión interactiva**: un usuario desde terminal o GUI quiere ejecutar un programa**
        - -**Para ofrecer un servicio**
        - **-Por solicitud del proceso en ejecución**

- *-Eventos que provocan la terminación de un proceso:*
  - *-Finalización normal:* **el proceso le indica al SO que ha terminado**
  - **-Límite de tiempo excedido.**
  - -Memoria no disponible.
  - -Violaciones de frontera.
  - **-Error de protección.**
  - **-Error aritmético.**
  - -Límite de tiempo.
  - **-Fallo de E/S.**
  - **-Instrucción no válida.**
  - -Instrucción privilegiada.
  - **-Uso inapropiado de datos.**
  - -Intervención del operador.
  - **-Terminación del padre.**
  - **-Solicitud del proceso padre.**

-Modelo de los 5 estados:

- -



Nuevo: entra en el sistema, se crean e inicializan estructuras. A espera de los recursos necesarios

- *-Listo:* **tiene todos los recursos asignados y está listo, solo falta la CPU.**
- *-Ejecutando:* **el proceso tiene la CPU, está en ejecución. Suponemos un único procesador.**
- *-Bloqueado:* **no puede ejecutar hasta que ocurra el evento que lo bloqueó**
- *-Saliente:* **ya no podrá ejecutar. Su información no está disponible**

- -**Si no existe memoria virtual** -> el proceso debe estar completo en memoria. Como las operaciones E/S son más lentas que el procesador, la mayoría de sus procesos estarán bloqueados.

-Para evitar esto se inventa el intercambio:

- -*Un proceso está bloqueado*: lo llevo a disco (swap file) y lo suspendo.
- -*Queda sitio para otro:* Nuevo o Suspendido

-Un proceso suspendido cumple:

- -**No está disponible inmediatamente para su ejecución**
- **-Puede estar o no a la espera de un evento de desbloqueo.**
- -**El proceso fue suspendido por un agente**
- **-No puede ser recuperado hasta que el agente lo indique**

Tema 4: Control Modos y ejecución

-Modos de ejecución:

- **-Las aplicaciones no deben poder usar todas las instrucciones de la CPU.**
- -El **SO tiene que poder utilizar todo el juego de instrucciones** de la CPU.
- -*Modo Kernel:*
  - **-Tiene acceso completo a los recursos hardware.**
  - **-Se puede ejecutar cualquier instrucción del repertorio de la CPU.**
  - -Normalmente está reservado sólo para las capas más internas del SO.
  - **-Un error en ejecución en modo kernel es catastrófico y provoca el paro completo del sistema.**
  - -Gracias a esto el kernel mantiene control del sistema.

- -*Modo usuario:*
  - **-Acceso limitado a los recursos.**
  - **-SI intenta ejecutar una instrucción privilegiada el microprocesador avisa al Kernel**, quien lo mata.
  - **-Tiene que pedirle al Kernel**, vía APIs de llamadas al sistema**, que ejecute la operación por él.**

- -El modo de ejecución lo indica un bit del PSW.
- -Este modo se cambia a Kernel en base a ciertos eventos.
- **-Tras la llamada al sistema o el fin de la rutina de interrupción, vuelve a "User Mode"**
- -*Para cambiar a Kernel Mode:*
  - -Ejemplo, *sucede una interrupción*:
    - **-El hardware guarda el estado**
    - **-Establece el bit indicando "Kernel Mode"**

        - **-Coloca en el PC la dirección de la rutina de tratamiento de interrupciones**
        - **-El controlador de interrupciones ha colocado en el bus de datos la entrada de la tabla de interrupciones**

- *-Cómo volver a "User Mode":*
    - **-El/los bits de modo se establecen de nuevo a User Mode con una instrucción de cambio de modo**
    - **-O simplemente se cambiará al restaurar el PSW del programa en ejecución**

-Creación de procesos:

- -Una vez que S.O. decide crear un proceso, *lo crea siguiendo estos pasos:*
    - **-Asigna un ID único**
    - **-Reserva espacio para el proceso**
    - **-Inicializa el PCB**
    - **-Establece los enlaces apropiados**
    - **-Crea o amplía otras estructuras**

-Cambio de proceso:

- -El S.O. realiza el cambio de proceso para mantener la multiprogramación
- -El cambio se realiza en cualquier momento que el S.O. tome el control.
- -Para interrumpir la ejecución se usa:
    - -Interrupción: causa externa a la ejecución del proceso. Reacción ante evento externo asíncrono
    - -Trap: causa asociada a la ejecución de la instrucción actual. Manejo de una condición de error o de excepción
    - -Llamada al sistema. Causa: solicitud explícita. Uso: llamada a una función del S.O.
- -Interrupción de un proceso:
    - -Ordinarias (Interrupciones HW):
        - -El control pasa a la rutina de gestión de interrupciones
    - -Luego salta a la rutina de tratamiento específico en función del tipo de interrupción.
- -Traps:
    - -El SO determina si el error es fatal:
        - -Lo es->Termina el proceso
        - -No lo es->Depende del diseño del SO
- -Llamadas al sistema:

- -El sistema pasa a modo kernel-> rutina de tratamiento->puede bloquearse o no

- -Cambio de modo:
  - -El ciclo de instrucción es "*ininterrumpible*"
  - -Tras el ciclo de instrucción tenemos la fase de interrupción -> *Comprueba si ha habido errores:*
    - -No hay interrupciones pendientes -> **Siguiente ciclo de instrucción.**
    - -Hay interrupción, el hardware:
      - PC <- Dirección de la rutina de tratamiento de interrupción
      - Cambio de modo usuario a Kernel.
      - Salvaguarda el contexto del proceso.
      - La siguiente instrucción será de la rutina de tratamiento de interrupciones y en modo kernel.
      - -Borra el flag de interrupciones.
      - -Copia el contexto del procesador en el PCB del proceso interrumpido
  - -La existencia de una interrupción -> Provoca cambio de modo.
- -No necesariamente un cambio de proceso.

-Ubicación del SO en ejecución:

- -Núcleo fuera de todo proceso (S.O. antiguos):
  - -Núcleo del S.O. como entidad separada que opera en modo Kernel, con su memoria protegida y pila propia
  - -Los procesos son solo programas de usuario.

- -Ejecución dentro de los procesos de usuario (PCs y workstations):
  - -Software del S.O. en el contexto de un proceso de usuario
  - -El SO es una colección de rutinas a las que el user llama
  - -Se mantiene una pila del kernel en su imagen para cada proceso
  - -Se enlazan(linkan) las rutinas del kernel necesarias para el proceso
  - -Se accede al conjunto de rutinas situadas en memoria compartida
  - -Estas rutinas se ejecutan en modo kernel, al que pasa por interrupción y se produce cambio de modo pero no de proceso
  - -Antes de retornar a user mode se comprueba si puede/debe continuar el mismo proceso, si es asi, se cambia de modo.

- -SO basado en procesos:
  - -Software del SO como procesos separados
  - -Las funciones más importantes se ejecutan como procesos
  - -El kernel se reduce a lo mínimo
  - -Favorece la programación modular
  - -Permite enviar procesos del SO a procesadores dedicados

Tema 5: Threads

-Dos visiones de un proceso:

- *-El proceso es visto por el SO como:*
  - **-La unidad propietaria de los recursos.**
  - **-La unidad expedición.**

- *-Un proceso:*
  - **-Incluye un espacio de direcciones virtuales para mantener la imagen del proceso.**
  - **-Mantiene una ejecución secuencial de sus instrucciones que puede ser intercalada con la de otros.**

- -Estas dos características son tratadas por el SO *de manera independiente:*
  - **-La unidad propietaria se conoce como Proceso (Process).**
  - **-La unidad de asignación se conoce como Hilo (Thread).**

- *-Cada hilo:*
  - **-Es un estado de ejecución.**
  - **-El contexto del procesador que el SO guarda para el proceso se asocia al hijo en ejecución**
  - **-Tiene una pila de ejecución propia**
  - **-Tiene almacenamiento estático para las variables locales**
  - **-Tiene acceso a memoria, variables y recursos asignados al proceso**
  - **-Las variables y recursos globales del proceso deben ser protegidas en los hilos con mecanismos de sincronización y exclusión mutua.**

-Un sistema multihilo:

- *-Multihilo:* **capacidad del SO de soportar múltiples hilos de ejecución en un solo proceso**
- -Todos los hilos de un proceso **comparten su estado y sus recursos**. Si un hilo:
  - *-Modifica una variable* -> **El resto la ven**
  - *-Abre para lectura un fichero* -> **El resto pueden leer**

-Estado de los hilos:

- -Operaciones básicas:
  - *-Creación:*
    - **-Al crear un proceso se crea un hilo principal.**
    - **-Se crea un nuevo hilo desde el principal.**
    - **-Se pasa puntero a su código y argumentos, tendrá espacio de contexto y espacio de pila.**
    - **-El hilo pasa a listo.**

- *-Bloqueo:*
  - **-A la espera de un suceso se guardan registros, PC y punteros de pila**
  - **-El hilo pasa a la cola de bloqueados del suceso relacionado**
  - **-El procesador puede seleccionar otro hilo de este u otro proceso**

- *-Desbloqueo:*
  - **-Al ocurrir el suceso, el hilo bloqueado pasa a la cola de Listos.**
  - **-Terminación: Se liberan el contexto y las pilas.**

-Implementación de los hilos:

- -Pueden ser “User level” Threads (ULT) o “Kernel Level” Threads (KLT)
- -ULT:
  - -Toda la gestión de hilos en la aplicación -> núcleo no es consciente de su existencia.
  - -El programador hará uso de la **biblioteca de hilos**, que **ofrece funciones para crear, destruir, sincronizar, planificar, comunicar, salvar y restaurar hilos**
  - -La aplicación arranca con **un único hilo.**
  - -Este puede crear otro mediante una función de librería.
  - -La librería toma el control, reserva espacio y crea las ED necesarias.
  - -La librería planifica la ejecución de un nuevo hilo en función de una política concreta establecida.
  - -La librería retorna el control de ejecución al hilo elegido.
  - -Cuando la librería toma el control, salva el contexto del hilo para restaurarlo cuando cede el control al mismo.
  - -Todas estas operaciones se realizan en el espacio de usuario.
  - -El núcleo desconoce la existen de los hilos.
  - -Ventajas:
    - **-El cambio de hilo no necesita privilegios de Kernel.**
    - **-Se pueden realizar planificaciones específicas y a medida.**
    - **-Los ULT pueden ejecutar en cualquier SO.**

  - -Desventajas:
    - **-En un SO la mayoría de las llamadas al sistema son bloqueantes**
    - **-Una estrategia ULT puede aprovechar las ventajas de un sistema multiprocesador. El kernel asigna el proceso a un único procesador.**

- -KLT:
  - -Todo el trabajo de gestión de los hilos lo realiza el núcleo
  - -En el código de la aplicación no hay código de gestión de hilos, solo llamadas al API del núcleo para la gestión de hilos
  - -Se puede programar cualquier aplicación como multihilo
  - -El kernel mantiene la información del proceso y de todos sus hilos

- -El kernel realiza la planificación a nivel de hilo, incluso en múltiples procesadores
- -Ventajas:
  - -Se pueden usar múltiples procesadores
  - -No se bloquean todos los hilos de un proceso si uno se bloquea
  - -Las rutinas del kernel suelen ser multihilo
- -Desventajas:
  - -Para cambiar de un hilo a otro dentro del mismo necesitamos un cambio a modo kernel

- -KLT y ULT combinadas:
  - -Algunos SO ofrecen una aproximación combinada
  - -Permiten asociar a uno o varios ULT un KLT
  - -Se generan varios modelos de asociación
  - -El programador puede ajustar el número de KLT para cada aplicación y vincular que ULT va con qué KLT
  - -Una aplicación correctamente diseñada puede combinar las ventajas de ambos modelos

Tema 6: Principios de la concurrencia

-Conceptos:

-Concurrencia:

- *-Multiprogramación:* **gestión de varios procesos en un sistema monoprocesador**
- *-Multiprocesamiento:* **gestión de varios procesos en un sistema multiprocesador**
- *-Procesamiento distribuido:* **gestión de varios procesos ejecutándose en sistemas de computadores** múltiples o distribuidos
- *-Principales dificultades:*
  - *-Monoprocesador:*
    - -Compartir recursos globales está lleno de **riesgos**.
    - **-Difícil la asignación óptima de recursos.**
    - -**Difícil depuración.**
    - -Sistemas multiprocesador:
      - **Problemas** propios por la **ejecución simultánea** de procesos.

  - *-Aspectos de Diseño y Gestión:*
    - -El SO debe ser **capaz de seguir la pista de distintos procesos** -> Gracias a los PCB.
    - -El SO debe **ubicar y desubicar varios recursos** para cada proceso activo:
      - *-Tiempo de procesador* -> **Planificación de procesos**
      - *-Memoria* -> **Gestión de la memoria virtual**
      - *-Ficheros* -> **Gestión de ficheros**
      - *-Dispositivos de E/S* -> **Gestión de E/S**
    - -**El SO debe proteger los datos y recursos de cada proceso.**

        - -**El funcionamiento y resultado de un proceso debe ser independiente. de la velocidad de su ejecución en relación con otros procesos.**

-Interacción de procesos:

- **-Procesos Cooperantes por Compartición:**
    - -Los procesos son conscientes de que **otros pueden acceder a los recursos** compartidos, **pero no saben quiénes son.**
    - -Los procesos **deben cooperar para garantizar la integridad** de los datos compartidos.
    - -Dos tipos de acceso a los datos(lectura-escritura) con problemas:
        - **-Exclusión Mutua**
        - **-Interbloqueo**
        - **-Inanición**
    - -Solo las operaciones de **escritura** deben ser **mutuamente excluyentes.**
    - -**La coherencia de los datos debe mantenerse**

- **-Procesos no cooperantes:**
    - -Competidores **entran en conflicto cuando compiten por el mismo recurso**
    - -Los procesos **utilizan los recursos y los dejan como estaban**
    - -Los procesos pueden verse afectados por **asignaciones de recursos**
    - -Si hay competencia se plantean 3 problemas a resolver:
        - **-Exclusión mutua**
        - **-Interbloqueo**
        - **-Inanición**
    - -El SO debe actuar
    - -Los procesos deben ser **proactivos**

- **-Procesos Cooperantes por Comunicación:**
    - -Los procesos **son conocedores** de la identidad **de los otros procesos**.
    - -Están **diseñados para colaborar** en la consecución de un fin.
    - -Para ello necesitan comunicarse:
        - **-Para sincronizarse.**
        - **-Para coordinarse.**
    - -Se **utilizan primitivas de paso de mensajes.**
    - -No se comparte nada (***no es necesaria la exclusión mutua***).
    - -Los **problemas de interbloqueo e inanición siguen presentes**.

-Sección crítica:

- -Es la **zona de código donde se accede a los recursos compartidos** y que **no puede ser ejecutada cuando otro proceso esté en la misma sección crítica.**

- -Dos zonas de código diremos que son la misma sección crítica si acceden a los mismos recursos compartidos

-**Exclusión Mutua:**

- -Es el requisito que **garantiza** que **dos procesos, que comparten secciones críticas, no pueden ejecutarse simultáneamente dentro de ellas.**

-Requisitos para la Exclusión Mutua:

- -**Sólo un proceso debe tener permiso para entrar en la sección crítica** por un recurso en un instante dado.
- -Un **proceso que se interrumpe** en una sección crítica debe **hacerlo sin interferir con los otros procesos.**
- ***-No puede permitirse el interbloqueo o la inanición.***
- -Cuando ningún proceso está en su sección crítica, **cualquier proceso que solicite entrar en la suya debe poder hacerlo sin dilación.**
- **-No** se deben **hacer suposiciones sobre la velocidad relativa de los procesos** o el número de procesadores.
- -Un proceso permanece en su sección crítica **sólo por un tiempo finito.**

-Solución Software:

-Algoritmo de Dekker:

- -Es un algoritmo que permite a **dos procesos compartir recursos sin conflicto**.
- -Si ambos procesos intentan acceder a la sección crítica **simultáneamente**, el algoritmo elige **un proceso según una variable de turno**. Si otro proceso lo está usando espera su finalización.

-Soluciones HW:

- **-Inhabilitación de interrupciones:**
  - -Un proceso continuará ejecutándose **hasta que solicite un servicio del sistema operativo o hasta que sea interrumpido**
  - -Para garantizar la exclusión mutua **es suficiente con impedir que un proceso sea interrumpido**
  - *-Problemas:*
    - -Se limita la capacidad del procesador para intercalar programas.
    - -Multiprocesador: Inhabilitar las interrupciones de un procesador no garantiza la exclusión mutua.

- **-Instrucciones especiales de máquina:**
  - -Se realizan en **un único ciclo de instrucción.**
  - **-No están sujetas a injerencias** por parte de otras instrucciones.
  - *-Ventajas:*
    - -Es **aplicable a cualquier número de procesos** en sistemas con memoria compartida, tanto de monoprocesador como de multiprocesador.
    - **-Es simple y fácil de verificar.**
    - -Puede usarse para disponer de **varias secciones críticas.**
  - *-Desventajas:*
    - -La espera activa consume tiempo del procesador.

- -Puede producirse inanición cuando un proceso abandona la sección crítica y hay más de un proceso en espera.
- -Interbloqueo.

Tema 7: Semáforos

-Concepto de semáforo:

- **-Es un mecanismo basado en señales entre procesos para poder sincronizarse.**
- -El semáforo **lo crea el SO a petición de un proceso** y se puede compartir.
- **-Los procesos pueden enviar y recibir señales a y del semáforo.**
- -Sobre un semáforo sólo se pueden hacer tres operaciones:
  - -**Para inicializar**: llamamos a "init" (S, valor).
  - **-Para recibir una señal** (vía el semáforo S): llamamos a "wait" (s)
  - **-Para transmitir una señal** (vía el semáforo S): llamamos a "signal" (S)

- **-Init, wait y signal son llamadas al sistema:**
  - -El semáforo lo gestiona el SO, por eso hay que hacer llamadas al sistema.

- **-Desde el punto de vista del programador es una variable que tiene un entero con el que se puede crearlo(init), restarle 1(wait), sumarle uno(signal).**
- **-Si S está en negativo se bloquea el proceso**, si no continua
- **-Inicializamos a 0 para que no pase ni el primero que llegue**
- **-Inicializamos a 1 para que pase el 1er wait (s)**
- **-Inicializamos a n para que pasen n wait (S)**

-Tipos de semáforos:

- *-Generales:* **con contador o enteros.**
- *-Binarios:* **solo puede ser 0 o 1.**
- *-Fuertes:* **planificación de la cola FIFO**
- *-Débil:* **planifica la cola mediante alguna política de planificación**

-Exclusión mutua con semáforos:

- -Tenemos n procesos.
- -Todos necesitan acceder a los mismos recursos.
- -Cada proceso tiene una sección crítica que accede a los recursos.
- *-Claves para cumplir la exclusión*:
  - **-Declara un semáforo s compartido para regular el acceso.**

    - -En cada proceso:
        - **-Wait antes de entrar en sección crítica.**
        - **-Signal al salir.**

-Barreras con semáforos:

- **-Una barrera es el punto en el código en el que ningún proceso lo traspasa hasta que todos han llegado a su ejecución.**
    - -Si todos los procesos llegan a esa barrera continúan.
    - -Si hay dos procesos ambos se detienen y notifican al otro que han llegado.

Tema 8: Monitores

- **-Un monitor es una estructura del lenguaje de programación**
- **-Consta de:**
    - -Uno o más procedimientos.
    - -Una secuencia de inicio.
    - -Datos que simbolizan el recurso compartido.

- **-Es un mecanismo de abstracción de datos que:**
    - -Permite representar de forma abstracta un recurso compartido por medio de un conjunto de variables.
    - -Los procedimientos son visibles desde el exterior.
    - -El acceso a las variables solo es posible mediante los procedimientos del monitor.

- **-Un programa que usa monitores tiene dos tipos de procesos:**
    - *-Pasivos:*
        - **-Implementan monitores.**
        - -A la espera que los activos hagan uso de los procedimientos exportados del monitor.
        - -Solo puede interactuar con variables compartidas del interior del monitor.
    - *-Activos:*
        - **-Hacen uso de los procedimientos del monitor.**

- **-Ventajas:**
    - **-Los programadores de los procesos activos no tienen por qué conocer la implementación.**
    - -Si se modifica la forma de gestión del recurso sin modificar la interfaz, los procesos activos no se ven afectados.
    - **-El programador del monitor solo se preocupa de que funcione.**
    - -Se puede implementar en módulos diferentes.

- **-Exclusión mutua:**
    - -Un monitor está construido de tal forma que:
        - **-La ejecución de los procedimientos no se ejecutan nunca de forma solapada.**

        - **-Se impide que dos procesos activos estén ejecutando simultáneamente procedimientos del monitor.**

- **-Un monitor se compone de:**
    - **-Conjunto de variables locales que se usan para almacenar el estado del recurso que representa.**
    - -Estas variables no varían entre sucesivas llamadas al monitor.
    - **-El código de inicialización se ejecuta antes de la primera instrucción del programa que usa el monitor.**
    - -El conjunto de procedimientos pueden modificar las variables locales.
    - **-Los procedimientos pueden aceptar parámetros.**

- **-Un proceso activo:**
    - -Ve al monitor como un conjunto de procedimientos que permiten usar un recurso compartido.
    - **-En la llamada está implícito el uso que se hace del recurso.**
    - -Decimos que:
        - -Un proceso activo está dentro del monitor **cuando está ejecutando uno de los procedimientos exportados** de éste
        - -Un proceso activo **abandona el monitor cuando termina de ejecutar el código de procedimiento** exportado.

    - -El acceso exclusivo en la ejecución de los procedimientos exportados del monitor se basa en:
        - *-La existencia de una cola del monitor:*
            - *-Cuando un proceso está dentro del monitor*: **si otro intenta acceder al monitor se tiene que bloquear en la cola**
            - *-Cuando un proceso abandona el monitor:* **selecciona el proceso en cabeza y lo desbloquea**

    - -Un monitor incluye variables de condición que solo se pueden ver dentro del monitor. Estar variables son el mecanismo de sincronización.

-Operaciones sobre las condiciones:
- *-Delay(c) o cwait(c):* bloquea el proceso que ejecuta el monitor. **Se bloquea al final de la cola FIFO**
- *-Resume(c) o csignal(c)*: **permite desbloquear al proceso cabecera de la cola de la condición c**
- *-Cola de cortesía:* **una cola de mayor prioridad** frente a la cola de monitor
- *-empty(c):* devuelve true o false si la cola está llena o vacia
- -En la versión de Lampson y Redell:
    - **-Se sustituye resume por cnotify**
    - **-Permite sacar todos los procesos de una condición con cbroadcast**

Tema 9: Paso de mensajes

- **-El paso de mensajes además es un sistema distribuido.**
- -Intercambio de información. *Primitivas:*
  - **-Send** (destino, mensaje)
  - **-Receive** (origen, mensaje)
- *-Aspectos de diseño:*
  - -Acuse de recibo. Perdida del mensaje. Perdida del acuse. Secuenciación de mensajes
  - -Verificación de autenticidad
  - -Rendimiento. **En la misma máquina puede ser lento el mecanismo de mensajes frente a monitores o semáforos**
- **-Sincronización:**
  - *-Primitivas bloqueantes o no bloqueantes:*
    - **-Envío bloqueante, recepción bloqueante:** permite sincro estricta entre varios procesos
    - **-Envío no bloqueante, recepción bloqueante:** combinación más común, requiere de ACK si se quiere verificar recepción
    - **-Envío no bloqueante, recepción no bloqueante.**
  - *-Posibles problemas:*
    - -Un envío no bloqueante tiene **el peligro de reenvíos continuos** sin bloqueo del emisor.
    - **-Una recepción bloqueante puede ser indefinida** si el mensaje del emisor se pierde.
  - *-Posibles soluciones:*
    - -Se puede permitir que un proceso **compruebe la existencia del mensaje** por el que espera antes de solicitar.
    - -Esto es útil si puede haber varios emisores.

- **-Direccionamiento:**
  - *-Directo:*
    - -Las primitivas **incluyen identificador del proceso con el que comunicar**
    - -Requiere el **conocimiento de los procesos origen y destino**
  - *-Indirecto:*
    - -Se usan buzones para intercambiar los mensajes
      - -Este esquema desacopla emisor y receptor
      - -Los buzones pueden ser estáticos o dinámicos
    - -El propietario del buzón puede ser:
      - -El emisor
      - -El receptor
      - -El SO

Tema 10: Planificación de procesos

-Multiprogramación y planificación

- *-Sistema multiprogramado*: varios procesos

- -La clave de la multiprogramación es la **planificación** de CPU para mejorar: **Tiempo de respuesta, productividad y eficiencia del procesador**
- -Elementos del kernel relacionados:
    - -Planificador
    - -Despachador
    - -Manejador de interrupciones
- -Planificador a largo plazo: la decisión de añadir un proceso al conjunto de procesos a ser ejecutados

-Planificador a medio plazo

- -Objetivos:
    - -Controlar el grado de multiprogramación mediante la admisión de nuevos programas en:
        - -Listo (cola de corto plazo) Memoria
        - -Listo Suspendido (cola de medio plazo) Zona de Intercambio
- -Sistemas de proceso por Lotes:
    - -Los procesos se incorporan a la cola de procesos por lotes en disco
    - -De la cola de proceso por lotes a Listo. Decisiones:
        - -Si el SO puede acoger a más procesos en Listo: control del grado de multiprogramación
        - -Decidir cual será el elegido de la cola de lotes: por algoritmo de planific, por herramienta de rendimiento o por planific E/S
- -Sistemas Interactivos:
    - -Cuando un usuario intenta conectarse se crea una solicitud al planificador a largo plazo
    - -Se aceptan todas hasta un límite. Se informa de sistema saturado

- *-Planificación a medio plazo:* **forma parte de la función de intercambio.**

- *-Planificación a Corto Plazo:*
    - **-Objetivos:**
        - **-Todos los procesos sean tratados por igual**
        - -Optimizar el uso de recursos
        - **-Evitar sobrecarga del sistema**
        - -Evitar inanición de procesos
        - -Evitar degradación paulatina
    - **-Se ejecuta siempre que un proceso abandona la CPU**

-Criterios de planificación a corto plazo

- **-Orientados al Usuario:**
    - *-Prestaciones:* tiempo de estancia, tiempo de respuesta, fecha tope.
    - *-Otros:* previsibilidad.

- **-Orientados al sistema:**
    - *-Prestaciones:* rendimiento, utilización del procesado.
    - *-Otros:* Equidad, imposición de prioridades, equilibrado de recursos

- **-Criterios de planificación:**
  - -Utilización de CPU
  - -Productividad(throughput)
  - -Tiempo de retorno
  - -Tiempo de respuesta
  - -Tiempo de espera

-Algoritmos de planificación de Procesos

- -La decisión de planificación puede ocurrir:
  - **-Cuando un proceso pasa de ejecución a espera**
  - **-Cuando un proceso termina**
  - **-Cuando un proceso pasa de ejecución a listo**
  - **-Cuando un proceso pasa de espera a listo**
  - **-Cuando llega una interrupción**
  - -Periódicamente
- -Los algoritmos se pueden clasificar por grupos en función de la forma de abandonar la CPU:
  - -Sin expulsión
  - -Con expulsión

-Planificación de procesos-Algoritmos

- **-FCFS (First Come, First Served):**
  - **-Política no primitiva**
  - -La cola de preparados es FIFO
  - -Abandona la CPU cuando: realiza una llamada al sistema o termina
  - **-Cuando el proceso actuál cesa su ejecución, se selecciona el proceso más antiguo**
  - -*Ventajas:* **no hay inanición**
  - -*Desventajas*: **favorece procesos largos**, con carga de CPU y no son válidos para sistemas de tiempo compartidos

- **-SPN (Shortest Process Next):**
  - -Política no preemtiva
  - -Abandona la CPU cuando:realiza llamada al sistema o termina
  - -Se selecciona el proceso con menor tiempo esperado de ejecución
  - -Un proceso corto salta a la cabeza de la cola, sobrepasando a trabajos largos
  - -Ventajas:Siempre obtendremos el mínimo tiempo medio de respuesta
  - -Inconvenientes: procesos largos sufren inanición, se recurre a estimaciones de ese tiempo

- -Uso de prioridades:
  - -Politica no preemtiva
  - -Tiene multiples colas de preparados para representar cada nivel de prioridad
  - -Un proceso no se ejecutará mientras haya procesos de mayor prioridad esperando

- -Abandona la CPU cuando: realiza llamada al sistema o termina
- -Los procesos de prioridad más baja pueden sufrir inanición

- **-RR (Round Robin**)
  - -Política apropiativa(preemtiva)
  - -Asignación de intervalo de tiempo: “quantum”
  - -Es el más sencillo y justo
  - -La cola de preparados se gestiona como una cola FIFO circular
  - **-Habiendo n procesos, un proceso no esperará más de (n-1)q unidades de tiempo, siendo q la duración del quantum**
  - -Abandona la CPU cuando: realiza una llamada al sistema, termina o se agota el quantum
  - -El quantum puede ser fijo o variable
  - -Ventajas: no hay inanición, especialmente diseñado para multiusuario
  - -Inconvenientes: rendimiento pobre de los procesos que realizan muchas operaciones de E/S

- **-SRTN (Shortest Remaining Time Next):**
  - -Es una versión apropiativa de la política de SPN
  - -Debe estimar tiempo de proceso
  - -Abandona la CPU cuando: realiza llamada al sistema, termina, si llega un proceso más corto

- **-Uso de prioridades apropiativo**
  - -Igual que el uso de prioridades, solo que abandona la CPU al llegar un proceso más importante

- **-MLQ:**
  - -Politica apropiativa
  - -Apropiado cuando los procesos se pueden clasificar en diferentes grupos
  - -Diferentes colas, con prioridad distinta y algoritmo de planific común o diferente
  - -No se puede ejecutar ningún proceso de una cola mientras queden en colas de mayor prioridad
  - -Inconvenientes: puede haber inanición

- **-MLQ con realimentación:**
  - -Politica apropiativa
  - -Soluciona el problema del anterior
  - -Penaliza los trabajos que han estado ejecutando durante más tiempo