**Unidad 1 – Introducción a los Sistemas Operativos**

**Sistema Informático:**

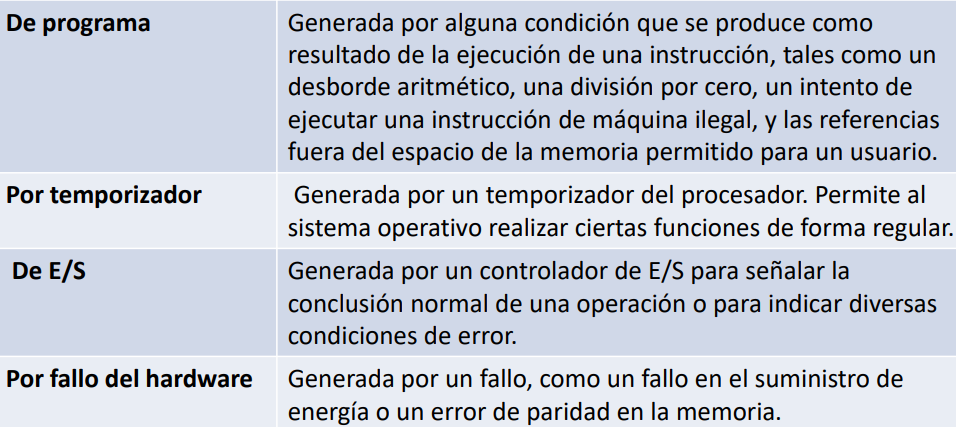
* Hardware – Software – Personas

**Elementos básicos computadora:**

* Procesador – Memoria Principal – Módulos de E/S – Bus del Sistema (comunicación)

**Ciclo de Instrucciones:** es el proceso requerido por **una sola** instrucción que debe entrar y salir.

**Ciclo de Interrupciones:** es un ciclo con **una fase de búsqueda y fase de ejecución**. El ciclo termina en una interrupción. Estas interrupciones mejoran la utilización del procesador. Si el procesador encuentra una interrupción en la ejecución se encarga de esta y sigue con la ejecución luego.

**Clases de Interrupciones:**

**Por temporizador:** si el proceso tarda mucho por una interrupción lo manda al final de la fila.

**De E/S:** // cuando faltan recursos, hasta que estos se habiliten.

Si un programa está manejando una interrupción el procesador va a otra instrucción **UMA** **(Acceso Uniforme a la Memoria)** es compartida uniformemente por todos los procesadores

**NUMA (Non Uniform Memory Access)** Tiempos de acceso dependen de la ubicación del proceso y la memoria

**UMA/NUMA** = Acceso a la memoria

**Ciclo de instrucción con interrupciones:** El procesador comprueba si hay interrupciones.

* Si no hay ninguna solicitud de interrupción, toma la siguiente instrucción para el programa actual
* Si una interrupción está pendiente, se suspende la ejecución del programa actual y se ejecuta la rutina del manejador de interrupciones.

**Mono programación:** El procesador espera que se complete la instrucción de entrada y salida de una ejecución antes de continuar.

**Multiprogramación:** Cuando una tarea necesita esperar por la entrada y salida de una ejecución, el procesador puede cambiar a la otra tarea.

**Sistema Operativo:** programa que administra el hardware.

**Objetivos - SO:**

**# Facilidad de uso:** el SO facilita el uso de la computadora.

**# Eficiencia:** el SO permite que los recursos se manejen de forma eficiente.

**#** **Capacidad para evolucionar:** nuevas cosas y utilidades para el usuario.

**Funciones - SO:**

**# Abstracción (aislar):** El SO se ocupa de generar una abstracción para que los programadores se puedan enfocar en resolver las necesidades de los usuarios.

**# Administración de recursos:** Al gestionar la asignación de recursos, el SO implementa políticas que los asigne de forma efectiva y acorde a las necesidades establecidas.

**# Aislamiento:** Cada usuario no tendrá que preocuparse si ese SO está siendo usado para poder utilizarlo el también.

**Elementos básicos – SO:**

**# Drivers:** programas que forman parte del SO y manejan los detalles de bajo nivel relacionando las operaciones de los distintos periféricos.

**# Nucleo o Kernel:** El SO es la parte central del núcleo a su vez este tiene subsistemas.

**# Interprete de comandos (Shell):** por ejemplo, la consola CMD de Windows donde podemos escribir comandos para el SO.

**# Proceso:** programa en ejecución.

**# Archivo:** secuencia de bits con nombre y permisos para entrar a los documentos.

**# Directorio:** colección de archivos. / **Directorio del sistema:** Ej: la carpeta system32.

**SO – Interfaces**

**Las interfaces son para:** Usuarios – Administradores – Desarrolladores.

**API (Application Programming Interface – Interfaz de Programación de Aplicaciones)**

Son Funciones y protocolos informáticos con los que los desarrolladores pueden crear programas concretos para bases de datos, sistemas operativos, plataformas online o redes sociales. Es la interfaz que facilita que programas distintos puedan comunicarse.

**Tipos de SO:**

**# Procesamiento por lotes (batch processing):** Los trabajos se agrupaban en lotes para ser ejecutados en secuencia.

**# Tiempo Compartido (time sharing):** sistema donde cada proceso dispone de una pequeña rodaja de tiempo (time slice) periódica para hacer uso de la CPU

**# Tiempo Real (real time):** Un sistema diseñado para cumplir tareas que deben completarse en un plazo prefijado

**# Sistemas multiusuario:** cada perfil de usuario tiene privilegios distintos.

**# Máquinas virtuales:** una máquina virtual es un software que simula un sistema de computación y puede ejecutar programas como si fuese una computadora real.

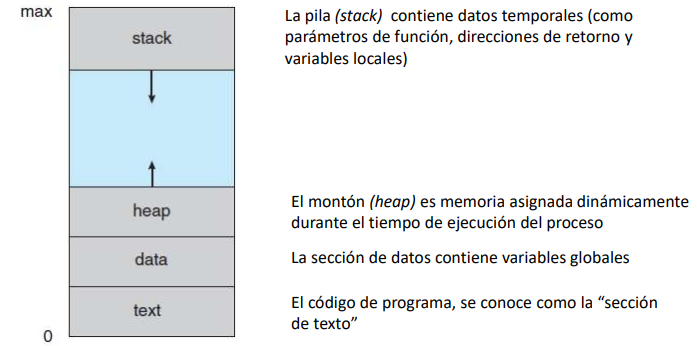
**# Sistemas Distribuidos:** conjunto de computadoras.

**# Sistemas Virtualizados:** los servidores centrales soportan la carga de trabajo de todos los usuarios.

**Unidad 2 – Gestión de procesos**

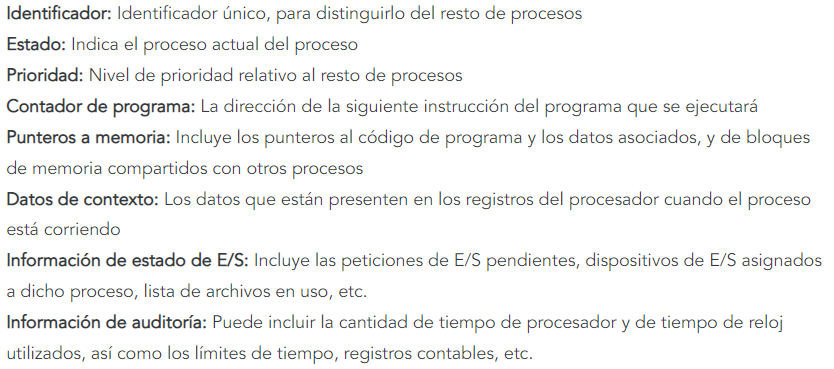
**Proceso**

Es un programa en ejecución este mismo necesita recursos (memoria, dispositivos de E/S, etc). Los procesos pueden crear otros procesos mientras se ejecutan.

**Elementos de un proceso:**

¿En qué se diferencia un proceso bloqueado de uno en estado de ejecución?

**El proceso que está en ejecución está haciendo uso de la CPU y el que está bloqueado no**



**PCB (PROCESS CONTROL BLOCK):** Es un conjunto de datos que necesita el sistema operativo para poder administrar un proceso. Cada proceso debe tener una estructura de datos que almacena su estado e información para su control. Es creado y gestionado por el sistema operativo. contiene suficiente información de forma tal que es posible interrumpir el proceso cuando está corriendo y posteriormente restaurar su estado de ejecución como si no hubiera habido interrupción alguna.

**Proceso – Modelo de 5 Estados**

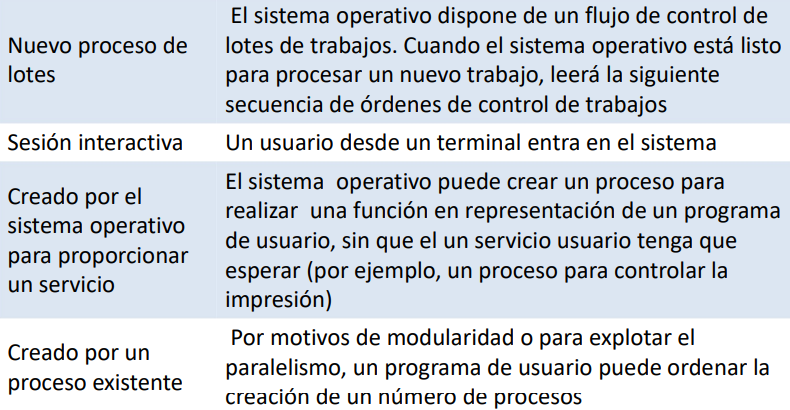
**1 - Ejecutando:** El proceso está actualmente en ejecución. Solo se puede tener un proceso en ejecución por procesador.

**2 - Listo:** Proceso preparado para ejecutar.

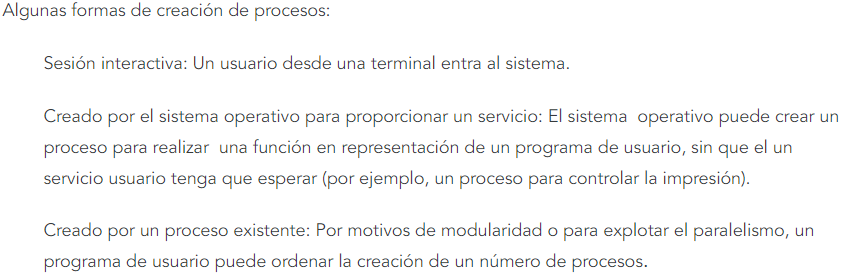
**3 - Bloqueado:** Proceso que no se puede ejecutar por alguna razón.

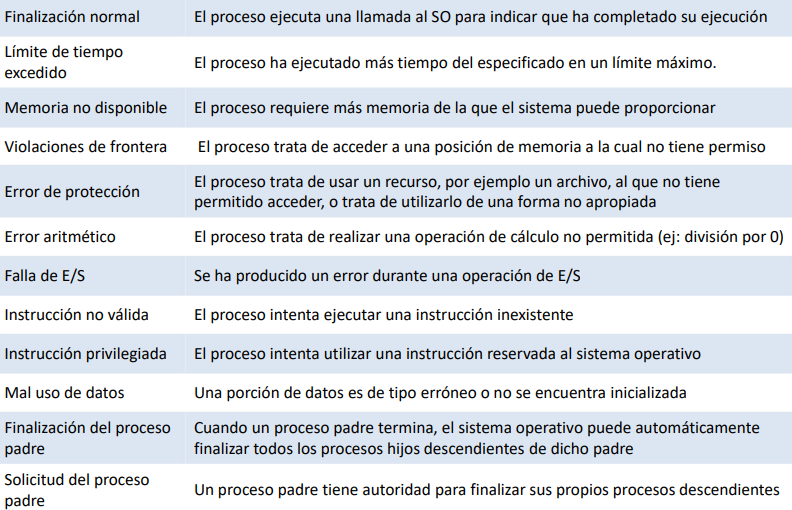
**4 - Nuevo:** Proceso que se acaba de crear.

**5 - Saliente:** Proceso que ha sido liberado del grupo de procesos ejecutables por el sistema operativo, debido a que ha sido detenido o que ha sido abortado por alguna razón.

**Creación de Procesos**

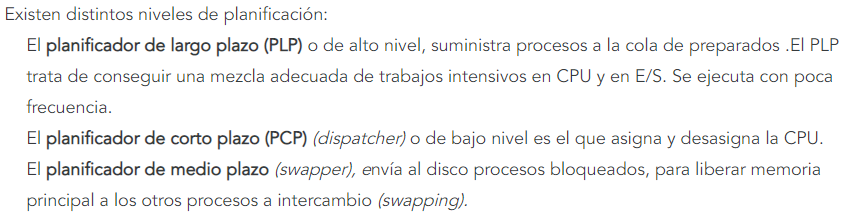
Un proceso puede crear otros mientras se ejecuta. El proceso creador de otros se llama **Padre** y los otros son los **hijos**. Cuando cada uno de estos procesos crea otros se da lugar a un **árbol.** La mayoría de los SO identifican los procesos con un **identificador de proceso unívoco “pid”** que normalmente es un número entero.

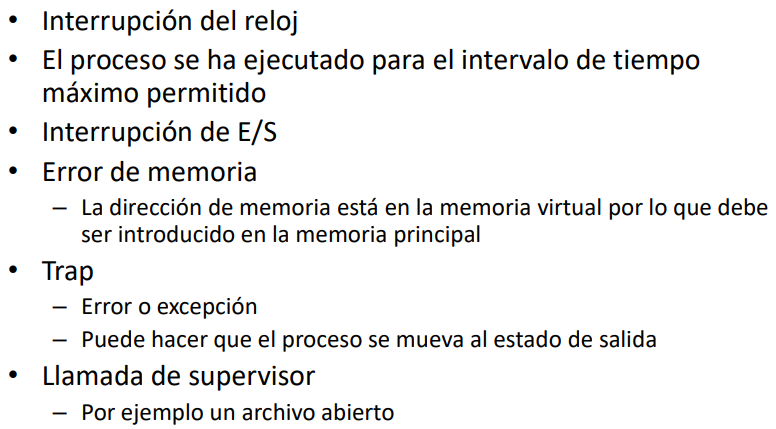


**Terminación de Procesos**

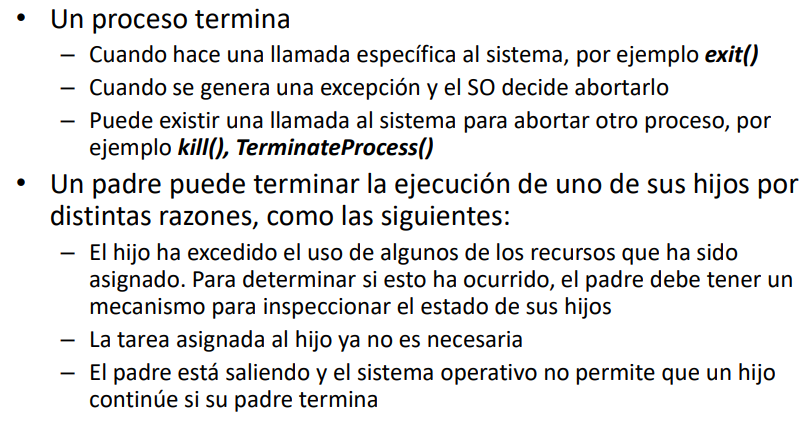
**Cambio de contexto (Context Switch)**

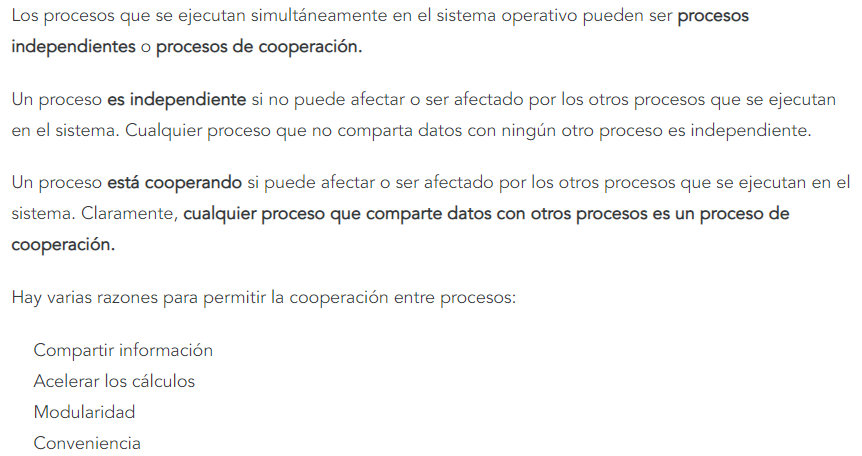
Operación que consiste en desalojar un proceso de la CPU y reanudar o empezar otro. El encargado de esto es el **dispatcher**. Este cambio es tiempo perdido así que debe ser rápido.

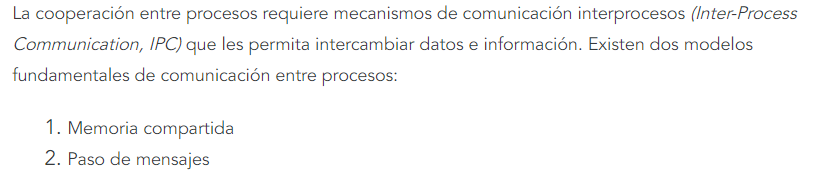


**Cuando cambiar un Proceso**

**Terminación de Procesos**







**Modelo de memoria Compartida:** Se establece una región de memoria para que los procesos puedan compartir datos entre ellos.

**Modelo de paso de mensaje:** Los procesos se comunican mediante mensajes y no usan una región de memoria compartida.

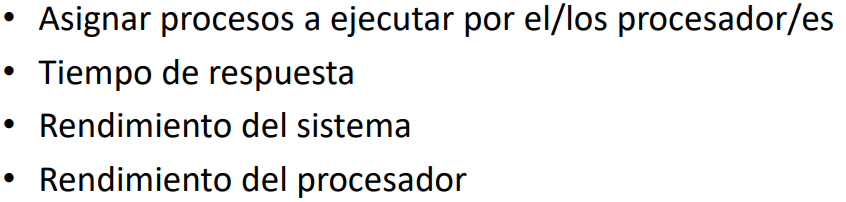
**Comunicación entre sistemas cliente-servidor / estrategias**

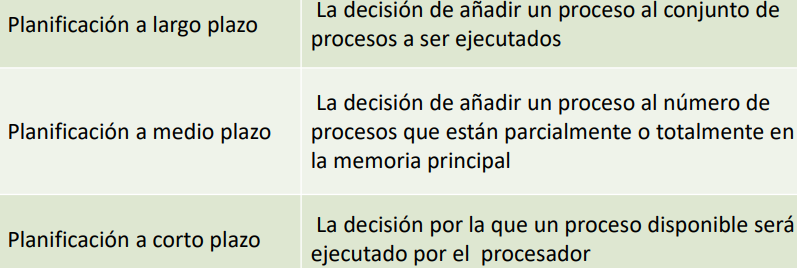
* **Sockets (zócalos):** método para la comunicación entre un programa del cliente y un programa del servidor. El socket es identificado por una dirección IP.
* **Llamada a procedimiento Remoto:** permiten que un proceso (el proceso de cliente) indique a otro proceso (el proceso de servidor) que ejecute llamadas de procedimiento como si el proceso de cliente hubiera ejecutado las llamadas en su propio espacio de direcciones.
* **Tuberías (Pipes):** conducto que comunica dos procesos.

**Hilo:**

Es una unidad básica de utilización de la CPU. (Cantidad básica de utilizar la CPU)

**Unidad 3 – Planificación**

**Objetivos de la planificación del SO**

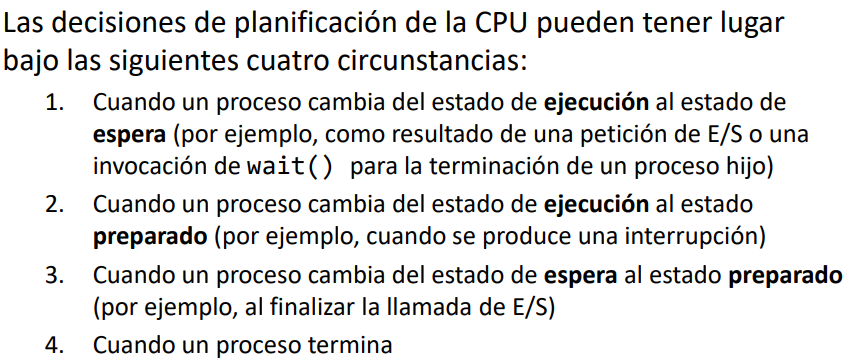
**Tipos de planificación**

**Planificación del procesador (processor scheduling)**

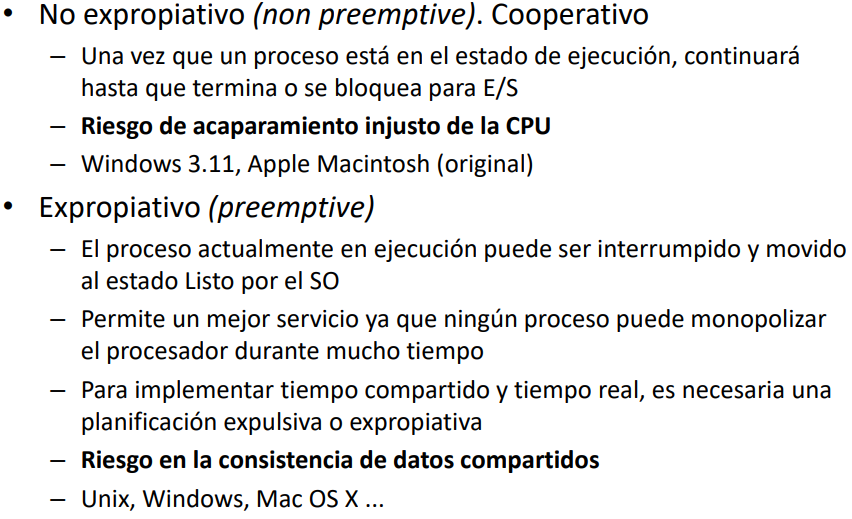
Como ya sabemos un sistema de un solo procesador solo puede ejecutar un proceso a la vez. La multiprogramación es cuando se tienen procesos en marcha en todo momento para maximizar la utilización de la CPU.

**El SO decide** que proceso ocupa el procesador cuando este queda libre. **Este mismo** también decide cuando un proceso que está en ejecución debe abandonar el procesador.

**Modos de decisión**



**Tipos de modos de decisión:**



Cuando la planificación se lleva a cabo en las circunstancias 1 y 4, se dice que el esquema de planificación es **no expropiativo** o **cooperativo**. De lo contrario, es **expropiativo** planificación donde el proceso que actualmente esta en ejecución puede ser interrumpido y movido al estado listo (ready) por el SO.

**Criterios de planificación**

Utilización de la CPU – Rendimiento – Tiempo De Retorno – Tiempo de Espera – Tiempo de Respuesta.

**Algoritmos de planificación**

* FCFS (First come, First served): el proceso que primero solicita la CPU es el primero que se le asigna.
* SPN/SJF (Shorted Process Next/Shorted Job First): Se le asigna a la CPU el proceso con la ráfaga de longitud más pequeña.
* SRT (Shortest Remaining Time): Elige el proceso con duración mas corta.

**Planificación basada en prioridades**

# Cada proceso tiene una prioridad asignada.

# La CPU toma el proceso con mayor prioridad.

# Prioridades definidas:

* Interna por el SO.
* Externa por el usuario.

**Turno rotativo (Round Robin) o también conocido como planificación clínica**

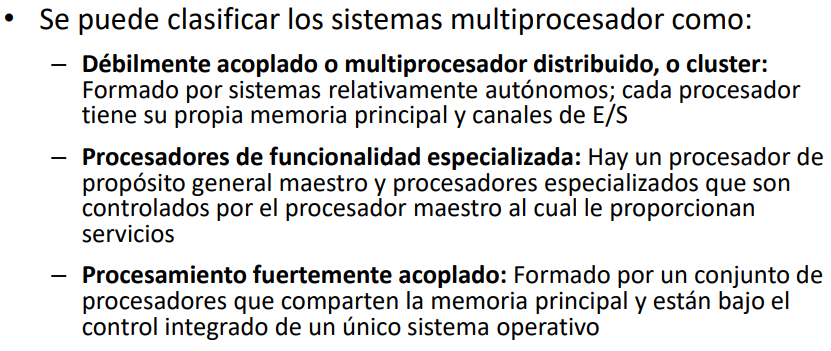
# Similar a FCFS, pero cada proceso dispone de una “ventana” (quantum) de tiempo de duración Q para estar en la CPU.

# Si transcurre Q y el proceso no liberó la CPU, el planificador lo desaloja y lo ingresa al final de la cola de listos.

**Planificación de multiprocesadores – SO con más de un procesador**

**El objetivo de la multiprogramación es:**

Tener algún proceso en marcha en todo momento, para maximizar la utilización de la CPU



**Tipos de procesamientos en multiprocesadores:**

**# Procesamiento Asimétrico:** Un procesador se encarga de planificar el trabajo de los restantes (técnica en desuso).

**# Procesamiento simétrico:** Cada procesador tiene su planificacion. Una cola x procesador. Se debe asegurar que dos procesadores elijan el mismo proceso.

**Política para la planificacion de multiprocesadores**

**# Afinidad al procesador:** Mantener el proceso siempre en el mismo procesador.

**# Equilibrio de carga:** Mantener a los procesadores no muy ocupados o muy ociosos.

* Migración comandada (push): Una tarea supervisora comprueba la carga y traslada procesos.
* Migración Solicitada (pull): Un procesador ocioso puede “robar” procesos a otro más cargado

¿Qué es un sistema multiprocesador **"débilmente acoplado o multiprocesador distribuido"**?

Es un sistema formado por sistemas relativamente autónomos; donde cada procesador tiene su propia memoria principal y canales de E/S

¿Qué es un sistema multiprocesador con **procesadores de funcionalidad especializada**?

Es un sistema con un procesador de propósito general maestro y procesadores especializados que son controlados por el procesador maestro al cual le proporcionan servicios

En los sistemas multiprocesadores, ¿a qué se denomina **afinidad al procesador**?

A la planificación que intenta mantener a un proceso siempre en un mismo procesador, para que aproveche los datos que puedan estar en la memoria caché de la CPU

En los sistemas multiprocesadores, ¿a qué se denomina **equilibrio de carga**?

A la planificación que tiene como objetivo mantener el trabajo de cada procesador uniformemente distribuido entre todos los procesadores

**Pregunta:** ¿Cuál de las siguientes sentencias marca una desventaja en el uso del algoritmo de planificación "el proceso con tiempo más corto es el siguiente para ejecutar" (SRT-Shortest Remaining Time)?

**Respuesta:** Se debe estimar el tiempo de procesamiento

**Pregunta:** ¿Cuál de las siguientes sentencias marca una desventaja en el uso del algoritmo de planificación "el proceso más corto es el siguiente para ejecutar" (SPN-Short Process Next)?

**Respuesta:** Riesgo de inanición de los procesos con ráfagas de larga duración

**Pregunta:** En la planificación por turno rotativo, cada proceso dispone de una “ventana” (quantum) de tiempo de duración Q para estar en la CPU. ¿Qué sucede si este tiempo Q está mal dimensionado?

Si Q es muy pequeño, ocurren más cambios de contexto y baja el rendimiento del sistema

**Unidad 7 – Virtualización**

**Virtualizar**

Significa aparentar que algo con una forma tiene otra. Virtualizar una computadora se trata de simular múltiples computadoras diferentes. También esta misma permite que los SO se ejecuten como apps dentro de otros SO.

**Desventajas:**

# Complejidad añadida a la ejecución del sistema operativo invitado. Hay capas intermedias hasta llegar al hardware.

# Pérdida de prestaciones ocasionada por las capas intermedias entre el sistema invitado y el hardware, además de las limitaciones impuestas por la compartición de los recursos hardware entre diferentes sistemas operativos.

**Máquina virtual**

Es aquella que **emula** a una computadora completa. Las máquinas virtuales **abstraen el hardware de la computadora,** formando entornos de ejecución diferentes. **También** podemos decir que las maquinas virtuales no es más que un software que es capaz de albergar en su interior otro SO. Las maquinas virtuales no significa que no tengan componentes, sino que los componentes están en la máquina host.

**Ventajas:**

# Permite ejecutar diferentes SO simultáneamente sobre un único hardware.

# Permite crear instantáneas, que es guardar un estado para luego volver en el momento que se necesite.

# Las apps ejecutadas están aisladas del SO anfitrión.

# Los sistemas virtualizados pueden ser “Portados” a otro equipo.

**Operaciones posibles con máquinas virtuales:**

* **Exportar** una máquina virtual
* **Importar** una máquina virtual
* **Clonar** una máquina virtual

**Operaciones con máquinas virtuales – redes:**

Conectar múltiples máquinas virtuales permite crear redes virtuales para una variedad de propósitos.

**Tipos de conexiones de red en máquinas virtuales (MV):**

En Virtual box, una máquina virtual puede tener hasta 8 tarjetas de red Ethernet.

* **Modo No conectado:** existe una tarjeta de red per no hay una conexión.
* **Modo Adaptador puente (bridge):** forma mas sencilla de acceder a una red.
* **Modo NAT:** En modo NAT, Virtual Box coloca un router virtual entre el exterior (hacia donde hace NAT) y el SO invitado. modo adecuado para conexión a internet. No se requiere configuración de red.
* **Modo Red Interna:** se pueden construir redes aisladas. Las MV no podrán comunicarse con el equipo anfitrión ni viceversa.
* **Modo Red NAT:** la red funciona como una LAN. Este modo es una extensión del modo NAT. Se trata de la unión entre las características una red NAT (para Internet) y una red Interna (conexión entre máquinas virtuales)
* **Modo solo anfitrión:** Se utiliza para crear una red interna a la que pertenecerá también el equipo anfitrión.

**Escritorios remotos**

Se utilizan para realizar una conexión remota de una PC a otra. Existen dos formas básicas de realizar una conexión remota o escritorio remoto:

* Por software de terceros
* Por Sistema Operativo

**Máquina virtual de proceso**

A diferencia de las maquinas virtuales comunes las maquinas virtaules de proceso **ejecutan un proceso concreto**. Por ejemplo, una app.

**Máquina virtual JAVA**

Interpreta y ejecuta instrucciones expresadas en un código binario especial, intermedio, denominado **bytecode**, el cual es generado por el compilador del lenguaje Java.

**Hipervisores**

El software de virtualización implementa lo que se denomina **hipervisor o VMM (Virtual Machine Monitor)** que consiste en una capa de abstracción entre el hardware de la máquina física y la máquina virtual formada por hardware y software virtualizado, haciendo el papel de intermediario entre lo real y lo virtualizado. Este mismo gestiona la CPU, memoria, almacenamiento y conexiones de red.

**Anfitrión (host):** es el SO con el que arranca.

**Huésped (guest):** es el SO que se instala en la VM.

**Tipos:**

**# Hipervisor tipo 1:** denominado también como “virtualización en modo nativo”, se caracteriza porque este software se instala directamente sobre el equipo haciendo las funciones tanto de sistema operativo (SO) como de virtualización. Se utiliza generalmente en las empresas.

**# Hipervisor tipo 2:** denominado también alojado (hosted), se caracteriza porque debe ser instalado en un equipo que cuente con un SO previo anfitrión (como Linux, MS Windows, Mac OS X, etc.). Este es el método de virtualización apropiado para probar software (sistemas operativos y aplicaciones) sin riesgo de afectar a nada de lo que haya en la máquina anfitriona. Este tipo es menos eficiente que el anterior.

**Unidad 8 – Microservicios**

**Sistema Monolítico**

Es un sistema autónomo el cual no depende de otras piezas de software y obedece únicamente a los componentes que están en su interior

# Los procesos están estrechamente asociados y se ejecutan como un solo servicio.

# Agregar o mejorar características de una aplicación monolítica se vuelve más complejo a medida que crece.

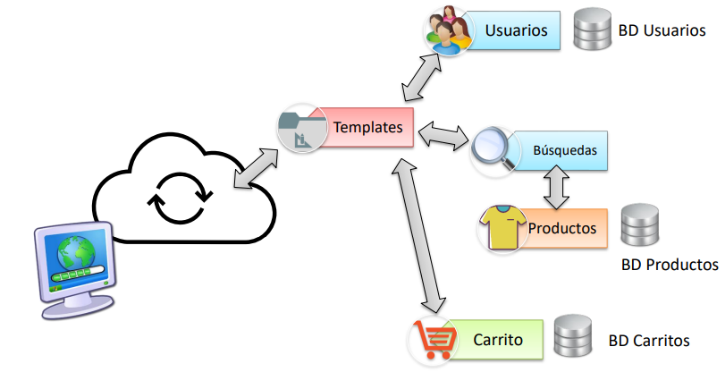
# Las arquitecturas monolíticas aumentan el riesgo de la disponibilidad de la aplicación porque muchos procesos dependientes y vinculados aumentan el impacto del error de un proceso.

**Microservicios**

Es un distintivo **sistema de desarrollo de software y tienen una sencilla escalabilidad**. Los microservicios son un enfoque arquitectónico y organizativo para el desarrollo de software donde el software está compuesto por pequeños servicios independientes que se comunican a través de APIs bien definidas.

* Los propietarios de estos servicios son equipos pequeños independientes.
* Las arquitecturas de microservicios permiten rapidez en el desarrollo de apps.
* También permite innovación.

**Arquitectura de microservicios**



**Apis y Servicios Web**

API (Application Programming Interface – Interfaz de Programación de Aplicaciones) – método de interacción entre los programas.

Son Funciones y protocolos informáticos con los que los desarrolladores pueden crear programas concretos para bases de datos, sistemas operativos, plataformas online o redes sociales. Es la interfaz que facilita que programas distintos puedan comunicarse.

Una API lleva a cabo sus funciones desde dentro de un programa de software. Si se trata de aplicaciones web, la API tiene su base en la web. Las APIs no necesitan, por defecto, basarse en web. Las APIs son un conjunto de especificaciones y reglas que permiten que un programa se comunique con otro.

Un **Servicio Web** utilizará, la mayor parte de las veces, protocolo HTTP para lograr la comunicación. También puede emplear SOAP, REST y XML-RPC.

**Microservicios – Características**

**Autónomos**

* Cada servicio componente en una arquitectura de microservicios se puede desarrollar, implementar, operar y escalar sin afectar el funcionamiento de otros servicios.
* Los servicios no necesitan compartir ninguno de sus códigos o implementaciones con otros servicios.

**Especializados**

* Cada servicio está diseñado para un problema específico.
* Si los desarrolladores aportan más código a un servicio a lo largo del tiempo y el servicio se vuelve complejo, se puede dividir en servicios más pequeños.

**Microservicios – Beneficios**

**Agilidad:** Los equipos se apropian de los servicios y trabajan de forma más rápida.

**Escalado Flexible:** permiten que cada servicio escale de forma independiente y se acomoden dependiendo el aumento de la demanda de ese servicio.

**Implementación sencilla:** Los microservicios permiten la integración y la entrega continuas, lo que facilita probar nuevas ideas y revertirlas si algo no funciona correctamente.

**Libertad tecnológica:** Los equipos eligen la herramienta para resolver problemas específicos.

**Código reutilizable:** La división del software en microservicios permite que un servicio especifico para una tarea pueda ser utilizado por muchas tareas más.

**Resistencia a errores:** cuando surge un error se puede arreglar sin bloquear toda la aplicación ya que se divide en microservicios. Esto mismo no se logra con una arquitectura monolítica.

**Separación del problema:** El microservicio podrá ser desarrollado con cualquier lenguaje de programación o base de datos, podrá ser compilado y ejecutado por cualquier sistema operativo.

**Proyectos pequeños:** Cualquier cambio podrá ser monitoreado y en caso de algún fallo este no afectará al funcionamiento global de la aplicación.

**Componentes de la arquitectura de microservicios**

* **Registro:** Proporciona independencia de la ubicación lógica o física.
* **Servidor perimetral:** Es el punto de acceso a cualquier llamada externa.
* **Balanceador:** Reparte la carga entre los servicios.
* **Servidor de configuración:** Proporciona la configuración del sistema.
* **Sistema de tolerancia a fallos:** Objetivo evitar fallos en cascada.
* **Gestion de logs:** Sistema para poder explotar la trazabilidad de los servicios.

**Microservicios – Desventajas**

**Alto consumo de recursos:** Cada microservicio tiene sus propios recursos y bases de datos, por lo tanto, consumen más memoria y CPU.

**Inversión de tiempo inicial:** Se necesita mas tiempo para la comunicación de microservicios.

**Complejidad en la gestión:** Con un gran número de microservicios, será más complicado controlar la gestión e integración de los mismos.

**Perfil de desarrollador:** Se requiere desarrolladoras de alto nivel de experiencia.

**No uniformidad:** Se produce una arquitectura no uniforme si no se gestiona el equipo.

**Dificultad en la realización de pruebas:** Los test de pruebas son complicados de hacer.

**Coste de implementación alto:** El costo es alto debido a costes de infraestructura y pruebas distribuidas.