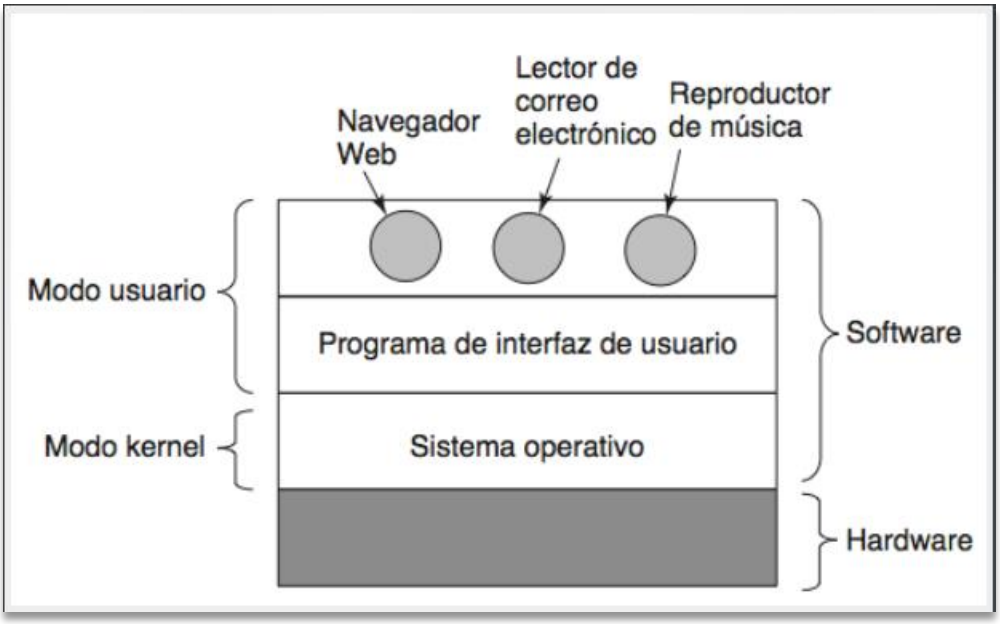
Contenido prueba1 sistemas operativos

- Un computador actual consta de varios elementos de hardware que lo convierten en un sistema complejo de administrar. Por esto existe el S.O. que es quien proporciona a los programas de usuario un "modelo" de computador más simple, además de encargarse de la administración de los recursos.



- Finalmente un computador se compone del hardware y software. El software se divide en dos modos: Kernel y Usuario.
- El S.O. es la pieza más importante de software, se ejecuta en el modo Kernel lo que le permite tener acceso total al hardware y a todas las instrucciones que la máquina pueda ejecutar.
- Todo el software que utilizamos y creamos (tanto shell como GUI) se ejecutan en modo Usuario, en donde se tiene un acceso restringido a las instrucciones de máquina.

Interfaz GUI

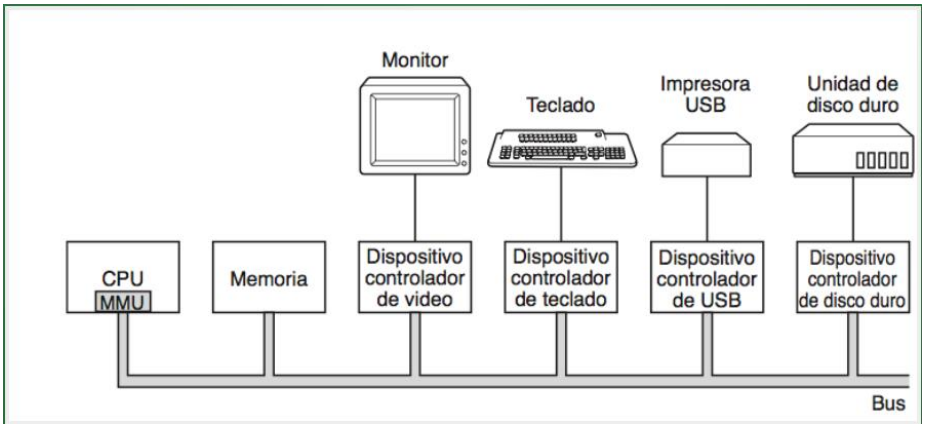
- La interfaz de usuario GUI o shell es el nivel más bajo de software en modo usuario, sobre él se ejecutan otras piezas de software como: Juegos, Navegador, Reproductor, etc.

¿QUÉ ES UN S.O.?

- S.O. como máquina virtual extendida: el desarrollador "promedio" no desea involucrarse en el proceso y funcionamiento a bajo nivel de algún elemento del hardware, si no, desea una "abstracción" de alto nivel que permita operar. (ej: el concepto "archivos" para referirse al contenido de un hdd).
- El trabajo de un S.O es crear buenas abstracciones, ocultar el hardware y presentar a los "programas “abstracciones agradables, elegantes, simples y consistentes
- S.O. como administrador de recursos: La anterior perspectiva es conocida como Top-Down. En el caso de considerar el S.O. como la herramienta para administrar las "piezas" de un sistema complejo es conocida como Bottom-up. En esta perspectiva el S.O. es encargado de entregar asignaciones ordenadas y controladas de cada elemento de hardware, entre los diversos programas que solicitan estos recursos.
- Manejar concurrencia en el consumo de recursos: Competencia. La tarea principal es llevar un registro de qué programa esta utilizando qué recurso, contabilizar su uso y entregarlo a quien lo requiere.

HARDWARE

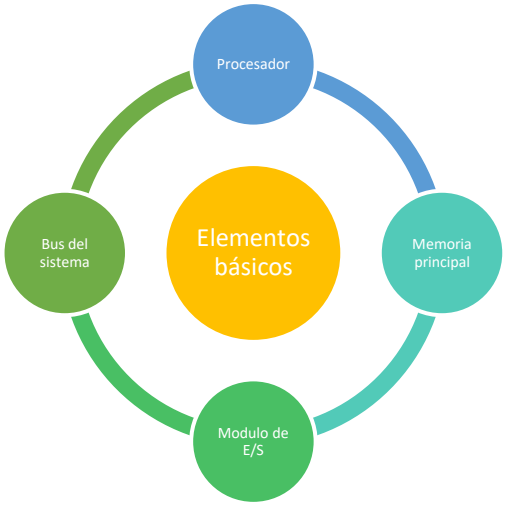
- Un S.O está relacionado con el hardware donde se ejecuta.
- Extiende las instrucciones y administra sus recursos.
- Conceptualmente una computadora es un modelo simple:



PROCESADOR

- Es el "cerebro" de toda computadora. Obtiene instrucciones desde la memoria y las ejecuta retornando resultados.
- Cada CPU tiene un conjunto específico de instrucciones posibles de ejecutar, generando incompatibilidad entre diversas arquitecturas, modelos e incluso "marcas".
- Toda CPU contiene "registros", pequeños espacios de memora de alta velocidad ocupados para almacenar variables temporales, instrucciones, etc.
- El S.O. debe estar al tanto de todos los registros. Cuando la CPU se multiplexa en el tiempo el S.O. detiene un proceso/trabajo para (re)iniciar otro. En cada detención de un proceso el S.O. debe almacenar los registros de la CPU para volver a utilizarlos.

Elementos básicos de un computador

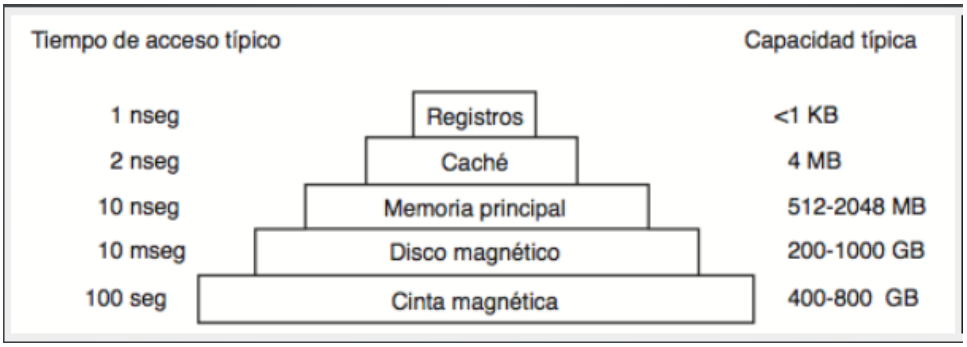


MULTIHILO Y MULTINÚCLEO

- Con el veloz crecimiento en la capacidad de producción de chips pequeños y de mayor poder fue posible multiplicar las capacidades de las CPU.
- El paso inicial fue multiplicar no sólo las unidades funcionales, si no, parte del sistema de control.
- A partir del Pentium 4 nace la tecnología "multihilo“ (multithreading). Esto permite que la CPU contenga el estado de 2 hilos de ejecución y alterne entre ellos, esto simula un paralelismo ya que la alternancia es en nanosegundos.
- Para el S.O. las capacidades multihilo tiene consecuencias directas ya que cada hilo es visto por el S.O. como una CPU individual, lo que implica que el S.O. debe tener capacidad de decidir que CPU activar para cada trabajo a procesar.
- Más allá del multihilo, están las CPU multinúcleo, chips con 2,4 o más procesadores completos. Para poder aprovechar estas capacidades el S.O. debe ser multinúcleo

MEMORIA

- Como segundo componente importante dentro de una computadora está la memoria.
- En teoría la memora es muy rápida (más rápida que la ejecución de una instrucción) de gran tamaño y económica. (en teoría)



DISPOSITIVOS E/S

- Otro recurso del sistema que deben ser administrados por el S.O. son los elementos de E/S. Estos constan de dos partes. El controlador y el dispositivo en sí.
- Cada dispositivo (controlador) requiere software para "entenderse" con el chip, esto es lo que conocemos como driver.
- El driver es un puente de software entre el controlador y el S.O. Cada fabricante de dispositivos tiene que suministrar un driver específico para cada S.O.
- El driver funciona en modo Kernel.

CONCEPTOS DE LOS S.O.

- Algunos conceptos básicos y abstracciones utilizados en todo S.O. ayudan a comprender su funcionamiento.
- Procesos: Es en esencia un programa en ejecución. Cada proceso tiene asociado un espacio de direcciones y una lista de ubicaciones de memoria que el proceso puede leer/escribir.
- También se le asocian algunos recursos que comúnmente incluye registros, lista de archivos abiertos, alarmas, procesos relacionados, etc.
- Un proceso es un recipiente que guarda toda la información necesaria para ejecutar un programa.

Ejecución de instrucciones

- Un programa que va a ejecutarse en un procesador consta de un conjunto de instrucciones almacenado en memoria. En su forma más simple, el procesamiento de una instrucción consta de dos pasos: el procesador lee (busca) instrucciones de la memoria, una cada vez, y ejecuta cada una de ellas. La ejecución del programa consiste en repetir el proceso de

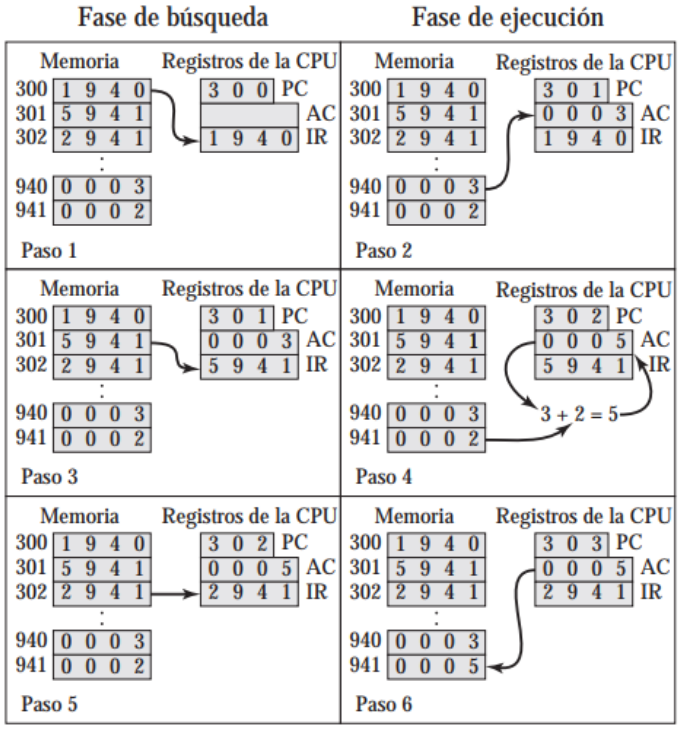
búsqueda y ejecución de instrucciones. La ejecución de la instrucción puede involucrar varias operaciones dependiendo de la naturaleza de la misma.

Búsqueda y ejecución de una instrucción

- La instrucción leída se carga dentro de un registro del procesador conocido como registro de instrucción (IR). La instrucción contiene bits que especifican la acción que debe realizar el procesador.
- El procesador interpreta la instrucción y lleva a cabo la acción requerida. En general, estas acciones se dividen en cuatro categorías:

- Procesador - Memoria
- Procesador – E/S
- Procesamiento de datos
- Control

Ejemplo

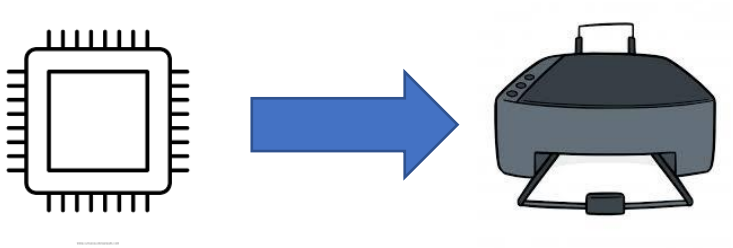


SISTEMA DE E/S

- Se pueden intercambiar datos directamente entre un módulo de E/S (por ejemplo, un controlador de disco) y el procesador. Al igual que el procesador puede iniciar una lectura o una escritura en memoria, especificando la dirección de una posición de memoria, también puede leer o escribir datos en un módulo de E/S. En este caso, el procesador identifica un dispositivo específico que está controlado por un determinado módulo de E/S. Por tanto, podría producirse una secuencia de instrucciones similar a la del ejemplo anterior, con instrucciones de E/S en vez de instrucciones que hacen referencia a memoria.

INTERRUPCIONES

- Básicamente, las interrupciones constituyen una manera de mejorar la utilización del procesador.
- Por ejemplo, la mayoría de los dispositivos de E/S son mucho más lentos que el procesador.



Clases de interrupciones

De programa	Generada por alguna condición que se produce como resultado de la ejecución de una instrucción, tales como un desbordamiento aritmético, una división por cero, un intento de ejecutar una instrucción de máquina ilegal, y las referencias fuera del espacio de la memoria permitido para un usuario.
Por temporizador	Generada por un temporizador del procesador. Permite al sistema operativo realizar ciertas funciones de forma regular.
De E/S	Generada por un controlador de E/S para señalar la conclusión normal de una operación o para indicar diversas condiciones de error.
Por fallo del hardware	Generada por un fallo, como un fallo en el suministro de energía o un error de paridad en la memoria.

Ciclos de interrupciones

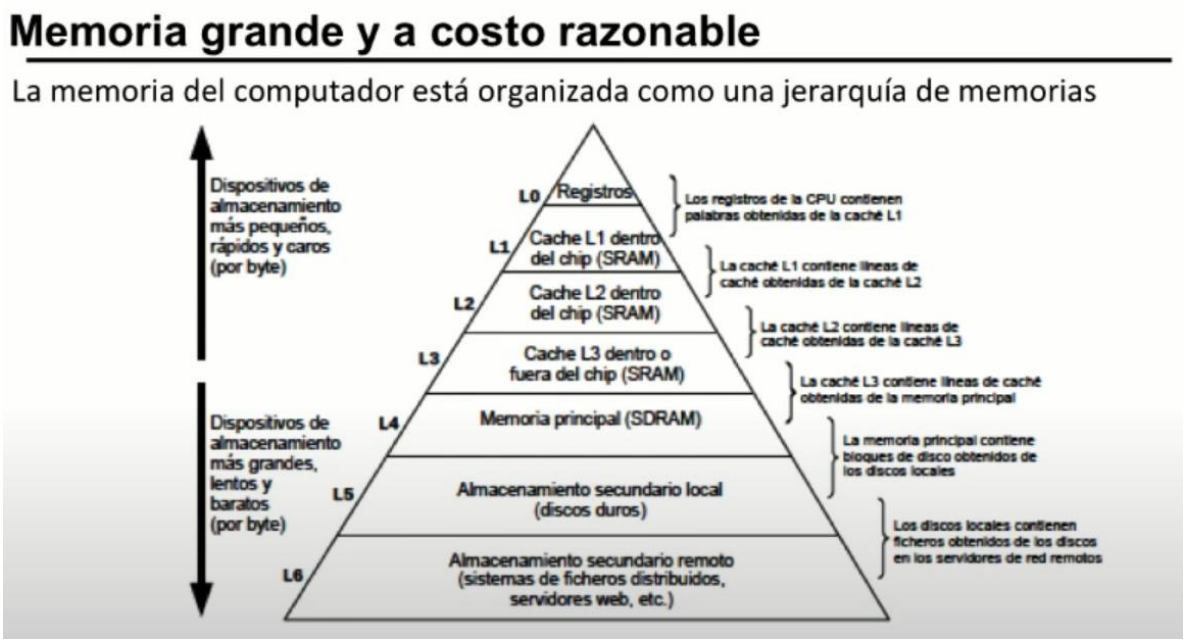
- Cuando el dispositivo externo está listo para ser atendido, es decir, cuando está preparado para aceptar más datos del procesador, el módulo de E/S de este dispositivo externo manda una señal de petición de interrupción al procesador. El procesador responde suspendiendo la ejecución del programa actual, saltando a la rutina de servicio específica de este dispositivo de E/S, conocida como manejador de interrupción, y reanudando la ejecución original después de haber atendido al dispositivo.

LA JERARQUÍA DE MEMORIA

- ¿cuál es su capacidad? ¿Cuál es su velocidad? ¿Cuál es su coste?
- Como se podría esperar, hay un compromiso entre las tres características fundamentales de la memoria: a saber, coste, capacidad y tiempo de acceso. En cualquier momento dado, se utilizan diversas tecnologías para implementar los sistemas de memoria. En todo este espectro de tecnologías, se cumplen las siguientes relaciones:
    - Cuanto menor tiempo de acceso, mayor coste por bit.
    - Cuanto mayor capacidad, menor coste por bit.
    - Cuanto mayor capacidad, menor velocidad de acceso.

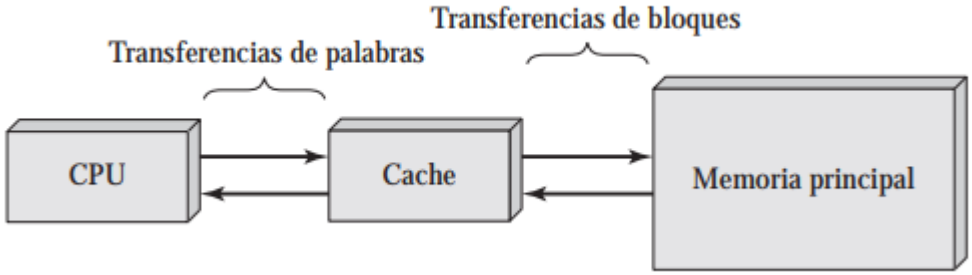
Tecnología	Tiempo de acceso típico	€ por MB
SRAM	1 ns	20 €
SDRAM	5 ns	0,01 €
Disco magnético	8.500.000 ns	0,0001 €

- Las 3 relaciones presentan un dilema.
- La solución a este dilema consiste en no basarse en un único componente de memoria o en una sola tecnología, sino emplear una jerarquía de memoria. Según se desciende en la jerarquía, ocurre lo siguiente:
    - a) Disminución del coste por bit.
    - b) Aumento de la capacidad.
    - c) Aumento del tiempo de acceso.
    - d) Disminución de la frecuencia de acceso a la memoria por parte del procesador.

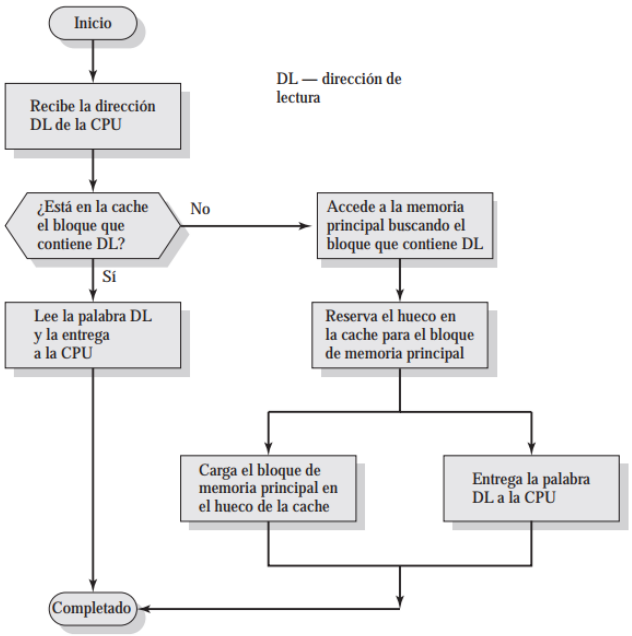


Memoria Cache

- El propósito de la memoria cache es proporcionar un tiempo de acceso a memoria próximo al de las memorias más rápidas disponibles y, al mismo tiempo, ofrecer un tamaño de memoria grande que tenga el precio de los tipos de memorias de semiconductores menos costosas. Hay una memoria principal relativamente grande y lenta junto con una memoria cache más pequeña y rápida. La cache contiene una copia de una parte de la memoria principal. Cuando el procesador intenta leer un byte de la memoria, se hace una comprobación para determinar si el byte está en la cache. Si es así, se le entrega el byte al procesador. En caso contrario, se lee e introduce dentro de la cache un bloque de memoria principal, que consta de un cierto número fijo de bytes, y, a continuación, se le entrega el byte pedido al procesador.



Operación de lectura de la cache

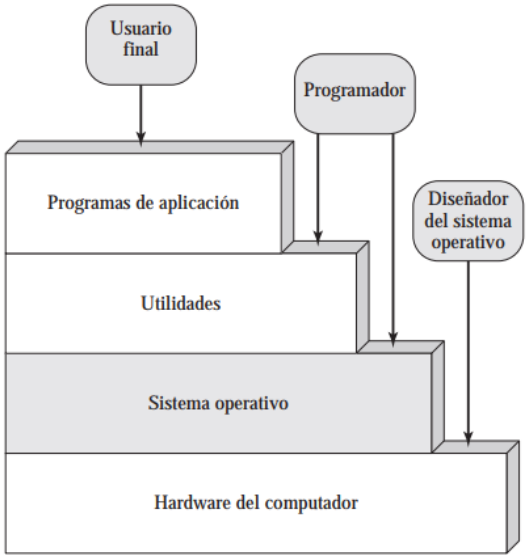


Servicios proporcionados por el S.O.

- De forma resumida, el sistema operativo proporciona normalmente servicios en las siguientes áreas:
- Desarrollo de programas: El sistema operativo proporciona una variedad de utilidades y servicios, tales como editores y depuradores, para asistir al programador en la creación de los programas. Normalmente, estos servicios se ofrecen en la forma de utilidades que, aunque no forman parte del núcleo del sistema operativo, se ofrecen con dicho sistema y se conocen como herramientas de desarrollo de programas de aplicación.
- Ejecución de programas: Se necesita realizar una serie de pasos para ejecutar un programa. Las instrucciones y los datos se deben cargar en memoria principal. Los dispositivos de E/S y los ficheros se deben inicializar, y otros recursos deben prepararse. Los sistemas operativos realizan estas labores de planificación en nombre del usuario.
- Acceso a dispositivos de E/S: Cada dispositivo de E/S requiere su propio conjunto peculiar de instrucciones o señales de control para cada operación. El sistema operativo proporciona una interfaz uniforme que esconde esos detalles de forma que los programadores puedan acceder a dichos dispositivos utilizando lecturas y escrituras sencillas
- Acceso controlado a los ficheros: Para el acceso a los ficheros, el sistema operativo debe reflejar una comprensión detallada no sólo de la naturaleza del dispositivo de E/S (disco, cinta), sino también de la estructura de los datos contenidos en los ficheros del sistema de almacenamiento. Adicionalmente, en el caso de un sistema con múltiples usuarios, el sistema operativo puede proporcionar mecanismos de protección para controlar el acceso a los ficheros.

- Acceso al sistema: Para sistemas compartidos o públicos, el sistema operativo controla el acceso al sistema completo y a recursos del sistema específicos. La función de acceso debe proporcionar protección a los recursos y a los datos, evitando el uso no autorizado de los usuarios y resolviendo conflictos en el caso de conflicto de recursos.
- Detección y respuesta a errores: Se pueden dar gran variedad de errores durante la ejecución de un sistema de computación. Éstos incluyen errores de hardware internos y externos, tales como un error de memoria, o un fallo en un dispositivo; y diferentes errores software, tales como la división por cero, el intento de acceder a una posición de memoria prohibida o la incapacidad del sistema operativo para conceder la solicitud de una aplicación. En cada caso, el sistema operativo debe proporcionar una respuesta que elimine la condición de error, suponiendo el menor impacto en las aplicaciones que están en ejecución. La respuesta puede oscilar entre finalizar el programa que causó el error hasta reintentar la operación o simplemente informar del error a la aplicación.
- Contabilidad. Un buen sistema operativo recogerá estadísticas de uso de los diferentes recursos y monitorizará parámetros de rendimiento tales como el tiempo de respuesta. En cualquier sistema, esta información es útil para anticipar las necesidades de mejoras futuras y para optimizar el sistema a fin de mejorar su rendimiento. En un sistema multiusuario, esta información se puede utilizar para facturar a los diferentes usuarios.

Capas y vistas de un sistema de computación



S.O. un mecanismo de control inusual en dos aspectos

- Las funciones del sistema operativo actúan de la misma forma que el resto del software; es decir, se trata de un programa o conjunto de programas ejecutados por el procesador.
- El sistema operativo frecuentemente cede el control y depende del procesador para volver a retomarlo.

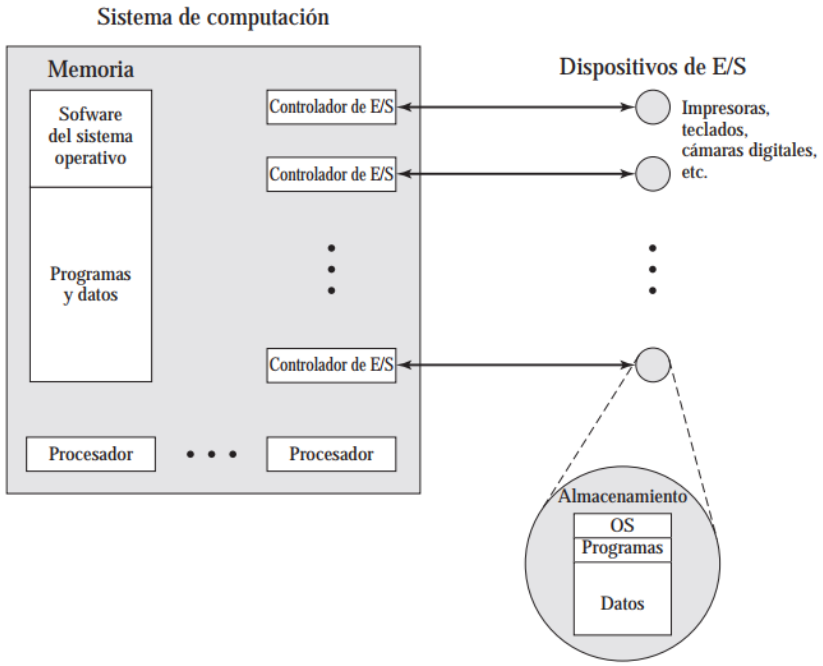


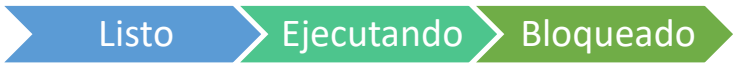
Figura 2.2. El sistema operativo como gestor de recursos.

Proceso

- La tarea fundamental de cualquier sistema operativo moderno es la gestión de procesos. El sistema operativo debe reservar recursos para los procesos, permitir a los mismos compartir e intercambiar información, proteger los recursos de cada uno de ellos del resto, y permitir la sincronización entre procesos. Para conseguir alcanzar estos requisitos, el sistema operativo debe mantener una estructura determinada para cada proceso que describa el estado y la propiedad de los recursos y que permite al sistema operativo establecer el control sobre los procesos.

Descripción y control de procesos

- El objetivo de los sistemas operativos tradicionales es la gestión de procesos. Cada proceso se encuentra, en un instante dado, en uno de los diferentes estados de ejecución, que incluyen:



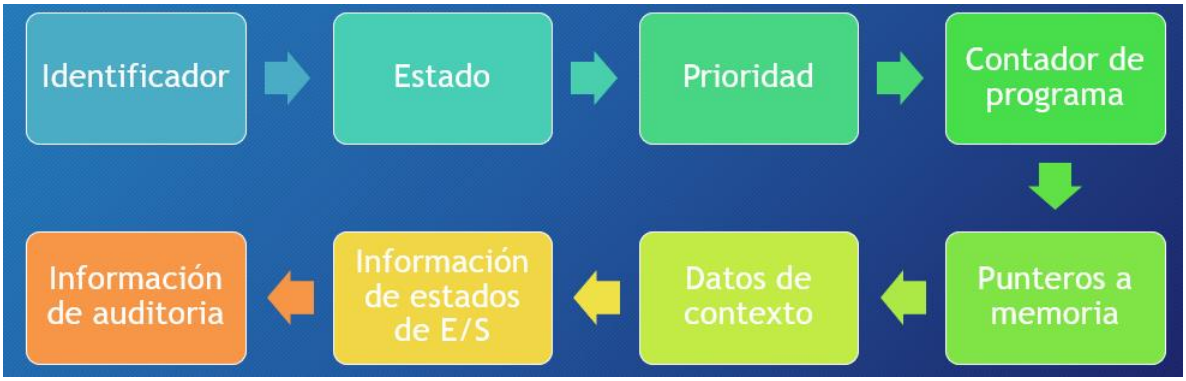
- El sistema operativo sigue la traza de estos estados de ejecución y gestiona el movimiento de procesos entre los mismos. Con este fin el sistema operativo mantiene unas estructuras de datos complejas que describen cada proceso. El sistema operativo debe realizar las operaciones de planificación y proporcionar servicios para la compartición entre procesos y la sincronización.

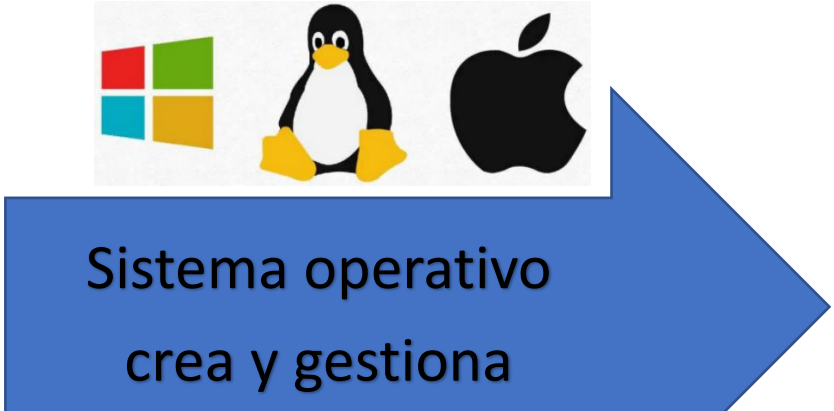
- La mayoría de los requisitos que un sistema operativo debe cumplir se pueden expresar con referencia a los procesos:

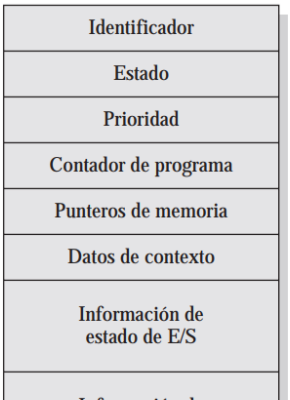
- El sistema operativo debe intercalar la ejecución de múltiples procesos, para maximizar la utilización del procesador mientras se proporciona un tiempo de respuesta razonable.
- El sistema operativo debe reservar recursos para los procesos conforme a una política específica (por ejemplo, ciertas funciones o aplicaciones son de mayor prioridad) mientras que al mismo tiempo evita interbloqueos.
- Un sistema operativo puede requerir dar soporte a la comunicación entre procesos y la creación de procesos, mediante las cuales ayuda a la estructuración de las aplicaciones.

¿QUÉ ES UN PROCESO?

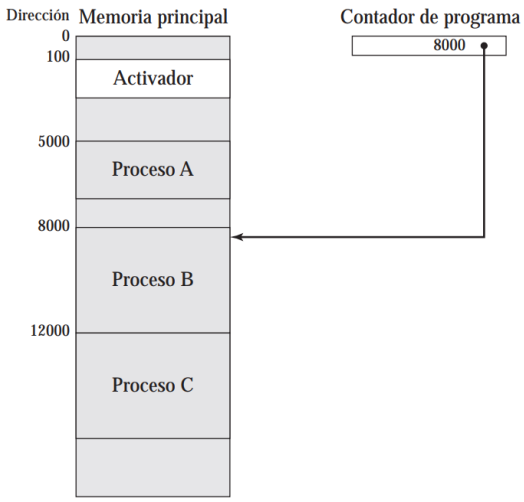
- Un proceso es en esencia un programa en ejecución. Cada proceso tiene asociado un espacio de direcciones, una lista de ubicaciones de memoria que va desde algúń mínimo hasta cierto valor máximo, donde el proceso puede leer y escribir informaciǿn.
- El espacio de direcciones contiene el programa ejecutable, los datos del programa y su pila.
- También hay asociado a cada proceso un conjunto de recursos, que comúnmente incluye registros (el contador de programa y el apuntador de pila, entre otros), una lista de archivos abiertos, alarmas pendientes, listas de procesos relacionados y toda la demás información necesaria para ejecutar el programa. En esencia, un proceso es un recipiente que guarda toda la información necesaria para ejecutar el programa.
- En cualquier instante puntual del tiempo, mientras el proceso está en ejecución, este proceso se puede caracterizar por una serie de elementos, incluyendo los siguientes:







- Se puede caracterizar el comportamiento de un determinado proceso, listando la secuencia de instrucciones que se ejecutan para dicho proceso. A esta lista se la denomina traza del proceso. Se puede caracterizar el comportamiento de un procesador mostrando cómo las trazas de varios procesos se entrelazan.



Creación de procesos

- Creación de un proceso. Cuando se va a añadir un nuevo proceso a aquellos que se están gestionando en un determinado momento, el sistema operativo construye las estructuras de datos que se usan para manejar el proceso y reserva el espacio de direcciones en memoria principal para el proceso. Estas acciones constituyen la creación de un nuevo proceso.

- Cuando un proceso lanza otro, al primero se le denomina proceso padre, y al proceso creado se le denomina proceso hijo. Habitualmente, la relación entre procesos necesita comunicación y cooperación entre ellos.

Terminación de procesos

- Todo sistema debe proporcionar los mecanismos mediante los cuales un proceso indica su finalización, o que ha completado su tarea. Un trabajo por lotes debe incluir una instrucción HALT o una llamada a un servicio de sistema operativo especifica para su terminación.

- Para una aplicación interactiva, las acciones del usuario indicarán cuando el proceso ha terminado. Por ejemplo, en un sistema de tiempo compartido, el proceso de un usuario en particular puede terminar cuando el usuario sale del sistema o apaga su terminal. En un ordenador personal o una estación de trabajo, el usuario puede salir de una aplicación (por ejemplo, un procesador de texto o una hoja de cálculo). Todas estas acciones tienen como resultado final la solicitud de un servicio al sistema operativo para terminar con el proceso solicitante.

Modelo de proceso de cinco estados

- Si todos los procesos estuviesen siempre preparados para ejecutar, la gestión de colas sería efectiva. La cola es una lista de tipo FIFO y el procesador opera siguiendo una estrategia cíclica (round-robin o turno rotatorio) sobre todos los procesos disponibles (cada proceso de la cola tiene cierta cantidad de tiempo, por turnos, para ejecutar y regresar de nuevo a la cola, a menos que se bloquee). Sin embargo, esta implementación es inadecuada.

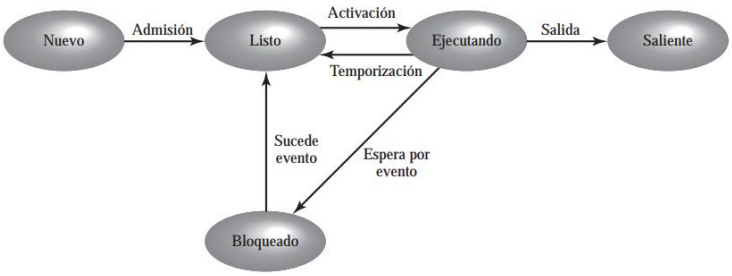


Figura 3.6. Modelo de proceso de cinco estados.

Procesos suspendidos

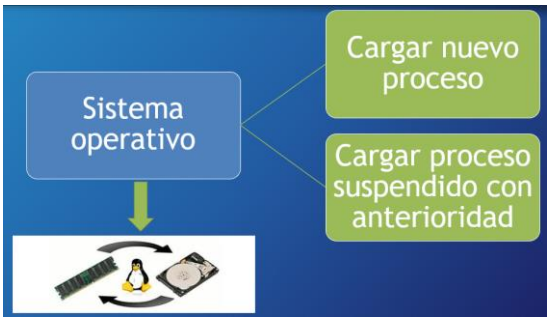
- Los tres principales estados descritos (Listo, Ejecutando, Bloqueado) proporcionan una forma sistemática de modelizar el comportamiento de los procesos y diseñar la implementación del sistema operativo. Se han construido algunos sistemas operativos utilizando únicamente estos tres estados.

- Expandir la memoria principal para acomodar más procesos. Hay dos fallos en esta solución. Primero, existe un coste asociado a la memoria principal, que, desde un coste reducido a nivel de bytes, comienza a incrementarse según nos acercamos a un almacenamiento de gigabytes. Segundo, el apetito de los programas a nivel de memoria ha crecido tan rápido como ha bajado el coste de las memorias. De forma que las grandes memorias actuales han llevado a ejecutar procesos de gran tamaño, no más procesos.

- Otra solución es el swapping (memoria de intercambio), que implica mover parte o todo el proceso de memoria principal al disco. Cuando ninguno de los procesos en memoria principal se encuentra en estado Listo, el sistema operativo intercambia uno de los procesos

bloqueados a disco, en la cola de Suspendidos. Esta es una lista de procesos existentes que han sido temporalmente expulsados de la memoria principal, o suspendidos. El sistema operativo trae otro proceso de la cola de Suspendidos o responde a una solicitud de un nuevo proceso. La ejecución continúa con los nuevos procesos que han llegado.

- Con el uso de swapping tal y como se ha escrito, debe añadirse un nuevo estado a nuestro modelo de comportamiento de procesos, el estado Suspendido. Cuando todos los procesos en memoria principal se encuentran en estado Bloqueado, el sistema operativo puede suspender un proceso poniéndolo en el estado Suspendido y transfiriéndolo a disco. El espacio que se libera en memoria principal puede usarse para traer a otro proceso.

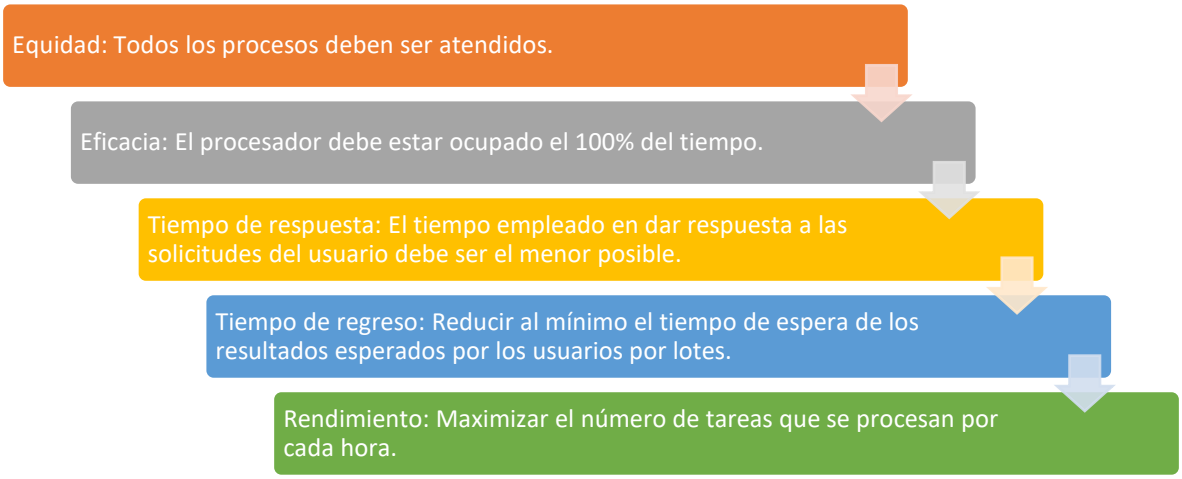


Planificación de procesos

- Conjunto de políticas y mecanismos incorporados al sistema operativo, a través de un módulo denominado planificador, que debe decidir cuál de los procesos en condiciones de ser ejecutado conviene ser despachado primero y qué orden de ejecución debe seguirse. Esto debe realizarse sin perder de vista su principal objetivo que consiste en el máximo aprovechamiento del sistema, lo que implica proveer un buen servicio a los procesos existentes en un momento dado.

Objetivos de la Planificación de procesos

- La Planificación de procesos tiene como principales objetivos la equidad, la eficacia, el tiempo de respuesta, el tiempo de regreso y el rendimiento.



Algoritmos de Planificación

- **Primero en llegar primero en salir:** Conocido como FIFO (First In First Out). Este algoritmo emplea una cola de procesos, asignando un lugar a cada proceso por el orden de llegada. Cuando el proceso llega es puesto en su lugar en la cola después del que llegó antes que él y se pone en estado de listo. Cuando un proceso comienza a ejecutarse no se interrumpe su ejecución hasta que termina de hacerlo.

- **Prioridad al más corto:** Su nombre es SJF (Shortest Job First). El proceso que se encuentra en ejecución cambiará de estado voluntariamente, o sea, no tendrá un tiempo de ejecución determinado para el proceso. A cada proceso se le asigna el tiempo que usará cuando vuelva a estar en ejecución, y se irá ejecutando el que tenga un menor tiempo asignado. Si se da el caso de que dos procesos tengan igual valor en ese aspecto emplea el algoritmo FIFO.

- **Round Robin:** A cada proceso se le asigna un tiempo determinado para su ejecución, el mismo tiempo para todos. En caso de que un proceso no pueda ser ejecutado completamente en ese tiempo se continuará su ejecución después de que todos los procesos restantes sean ejecutados durante el tiempo establecido. Este es un algoritmo basado en FIFO que trata la cola de procesos que se encuentran en estado de listos como una cola circular.

- **Planificación por prioridad:** En este tipo de planificación a cada proceso se le asigna una prioridad siguiendo un criterio determinado, y de acuerdo con esa prioridad será el orden en que se atienda cada proceso.

- **Planificación garantizada:** Para realizar esta planificación el sistema tiene en cuenta el número de usuarios que deben ser atendidos. Para un número "n" de usuarios se asignará a cada uno un tiempo de ejecución igual a 1/n.

- **Planificación de Colas Múltiples:** El nombre se deriva de MQS (Multilevel Queue Schedulling). En este algoritmo la cola de procesos que se encuentran en estado de listos es dividida en un número determinado de colas más pequeñas. Los procesos son clasificados mediante un criterio para determinar en qué cola será colocado cada uno cuando quede en estado de listo. Cada cola puede manejar un algoritmo de planificación diferente a las demás.

Procesos e Hilos

- ¿Qué es un hilo?

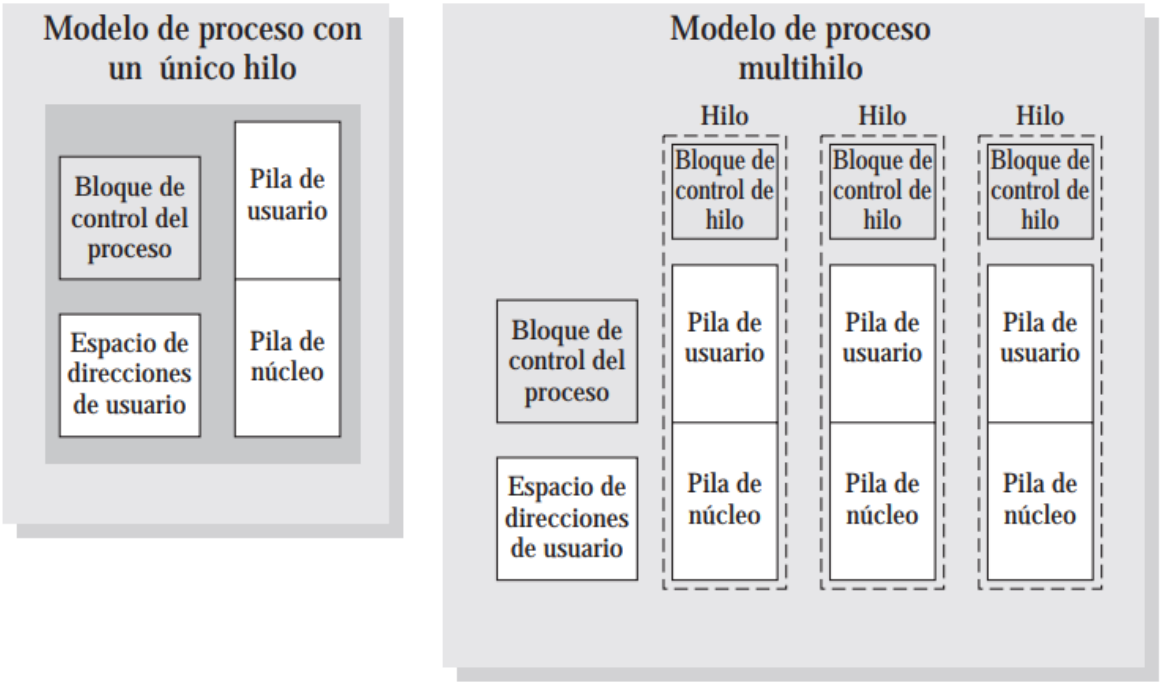
- Es una característica que permite a una aplicación realizar varias tareas simultáneamente.

- MULTIHILO

- Multihilo se refiere a la capacidad de un sistema operativo de dar soporte a múltiples hilos de ejecución en un solo proceso. El enfoque tradicional de un solo hilo de ejecución por proceso, en el que no se identifica con el concepto de hilo, se conoce como estrategia monohilo.

- Dentro de un proceso puede haber uno o más hilos, cada uno con:

- Un estado de ejecución por hilo (Ejecutando, Listo, etc.).
- Un contexto de hilo que se almacena cuando no está en ejecución; una forma de ver a un hilo es como un contador de programa independiente dentro de un proceso.
- Una pila de ejecución.
- Por cada hilo, espacio de almacenamiento para variables locales.
- Acceso a la memoria y recursos de su proceso, compartido con todos los hilos de su mismo proceso.

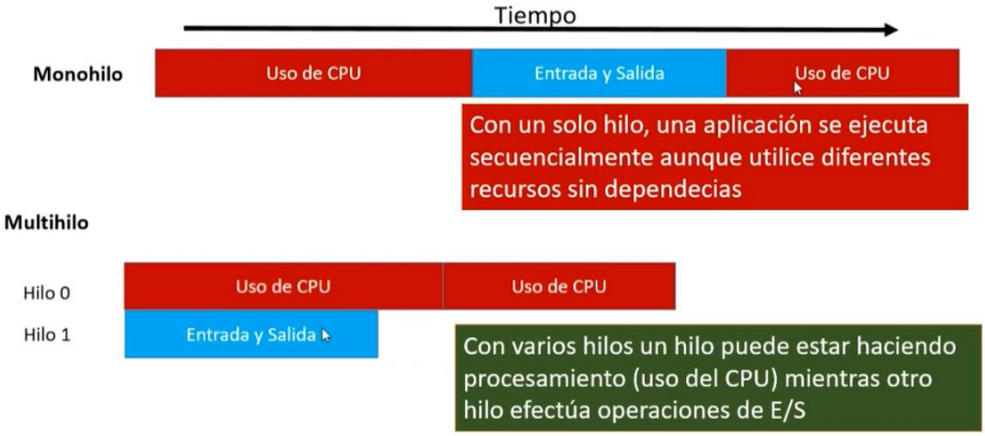


Beneficios de los hilos

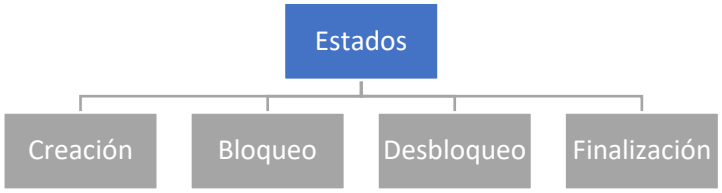
- Los mayores beneficios de los hilos provienen de las consecuencias del rendimiento:

1. Lleva mucho menos tiempo crear un nuevo hilo en un proceso existente que crear un proceso totalmente nuevo. Los estudios realizados por los que desarrollaron el sistema operativo Mach muestran que la creación de un hilo es diez veces más rápida que la creación de un proceso en UNIX [TEVA87].
2. Lleva menos tiempo finalizar un hilo que un proceso.
3. Lleva menos tiempo cambiar entre dos hilos dentro del mismo proceso.
4. Los hilos mejoran la eficiencia de la comunicación entre diferentes programas que están ejecutando. En la mayor parte de los sistemas operativos, la comunicación entre procesos independientes requiere la intervención del núcleo para proporcionar protección y los mecanismos necesarios de comunicación. Sin

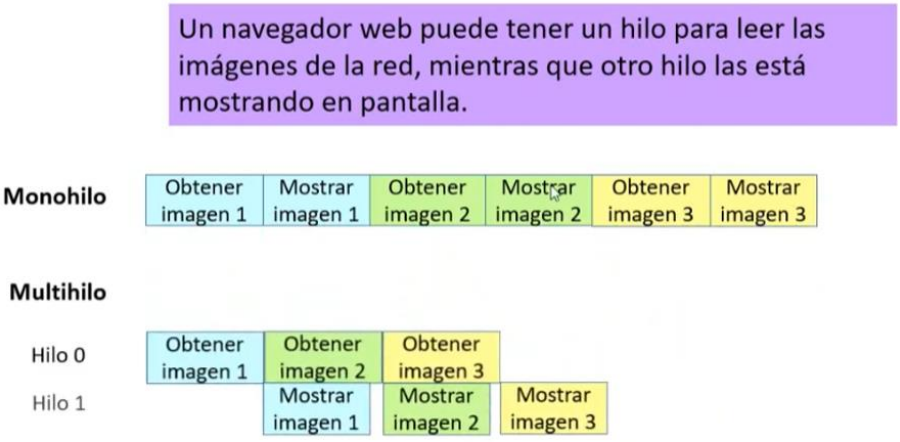
embargo, ya que los hilos dentro de un mismo proceso comparten memoria y archivos, se pueden comunicar entre ellos sin necesidad de invocar al núcleo.



Estados de hilos



Ejemplo



Soporte de hilos

Espacio de usuario ULT – User Level Threads

- Implementados en forma de biblioteca de funciones en espacio de usuario.
- El kernel no tiene conocimiento sobre ellos, no ofrece soporte de ningún tipo.

- Es mucho más rápido pero presentan algunos problemas → Llamadas al sistema bloqueantes.

Espacio de núcleo KLT – Kernel Level Threads

- El kernel se ocupa de crearlos, planificarlos y destruirlos.
- Es un poco más lento ya que hacemos participar al kernel y esto supone un cambio de modo de ejecución.
- En llamadas al sistema bloqueantes sólo se bloquea el thread implicado.
- En sistemas SMP, varios threads pueden ejecutarse a la vez.
- No hay código de soporte para thread en las aplicaciones.
- El kernel también puede usar threads para llevar a cabo sus funciones.

Modelos de hilos

Modelos de múltiples hilos: Muchos a uno

- Hace corresponder múltiples hilos de usuario a un único hilo del núcleo.
- Biblioteca de hilos en espacio de usuario.
- Llamada bloqueante:
  – Se bloquean todos los hilos.
- En multiprocesadores no se pueden ejecutar varios hilos a la vez.

Modelo de múltiples hilos: Uno a uno

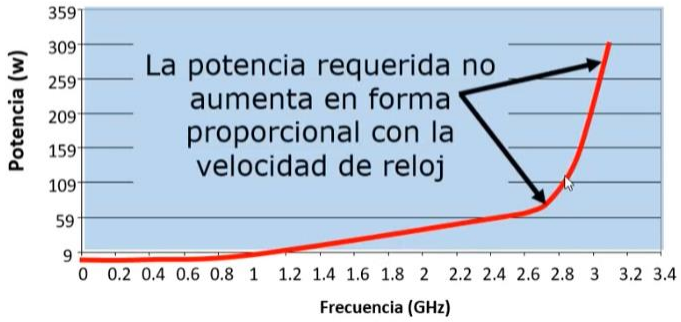
- Hace corresponder un hilo el kernel a cada hilo de usuario.
- La mayoría de las implementaciones restringen el número de hilos que se pueden crear.

Modelos de múltiples hilos: Muchos a muchos

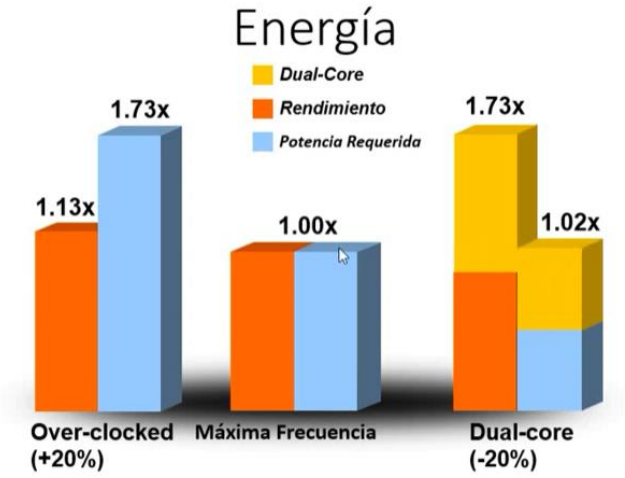
- Este modelo multiplexa los threads de usuario en un número determinado de threads en el kernel.
- El núcleo del sistema operativo se complica mucho.

La velocidad del CPU tendía a duplicarse cada 18 meses

Curva de Potencia vs. Frecuencia para arquitecturas con un núcleo

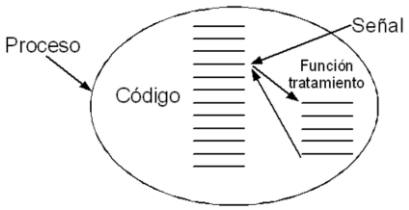


Menor velocidad de reloj nos permite ahorrar potencia para aumentar los núcleos



Señales

- Son un mecanismo que permite avisar a un proceso de la ocurrencia de un evento.
- Ejemplos:
    - Un proceso padre recibe la señal SIGCHLD cuando termina un proceso hijo.
    - Un proceso recibe una señal SIGILL cuando intenta ejecutar una instrucción máquina ilegal.
- Las señales son interrupciones al proceso
- Envío o generación
    - Proceso -- Proceso (dentro del grupo) con el kill
    - SO -- Proceso



Temporizadores

- El sistema operativo mantiene un temporizador por proceso (caso UNIX).
    - Se mantiene en el bloque de control de procesos (BCP) del proceso un contador del tiempo que falta para que venza el temporizador.
    - La rutina del sistema operativo actualiza todos los temporizadores.

    - Si un temporizador llega a cero se ejecuta la función de tratamiento.
- En UNIX el sistema operativo envía una señal SIGALRM al proceso cuando vence su te mporizador.

Puntos a recordar

- Un proceso puede tener varios hilos de ejecución.
- Una aplicación multihilo consume menos recursos que una aplicación multiproceso.
- Cada sistema operativo tiene un modo de soporte de hilos entre ULT y KLT.
- PTHREADS es una biblioteca de hilos de usuario.
- Win32 ofrece hilos en el núcleo con soporte para conjuntos de hilos (*Thread Pools)*.

Concurrencia

- Los temas centrales del diseño de sistemas operativos están todos relacionados con la gestión de procesos e hilos:
    - Multiprogramación. Gestión de múltiples procesos dentro de un sistema monoprocesador.
    - Multiprocesamiento. Gestión de múltiples procesos dentro de un multiprocesador.
    - Procesamiento distribuido. Gestión de múltiples procesos que ejecutan sobre múltiples sistemas de cómputo distribuidos.
- La concurrencia aparece en tres contextos diferentes:
- Múltiples aplicaciones. La multiprogramación fue ideada para permitir compartir dinámicamente el tiempo de procesamiento entre varias aplicaciones activas.
- Aplicaciones estructuradas. Como extensión de los principios del diseño modular y de la programación estructurada, algunas aplicaciones pueden ser programadas eficazmente como un conjunto de procesos concurrentes.
- Estructura del sistema operativo. Las mismas ventajas constructivas son aplicables a la programación de sistemas y, de hecho, los sistemas operativos son a menudo implementados en sí mismos como un conjunto de procesos o hilos.

PRINCIPIOS DE LA CONCURRENCIA

En un sistema multiprogramado de procesador único, los procesos se entrelazan en el tiempo para ofrecer la apariencia de ejecución simultánea. Aunque no se consigue procesamiento paralelo real, e ir cambiando de un proceso a otro supone cierta sobrecarga, la ejecución entrelazada proporciona importantes beneficios en la eficiencia del procesamiento y en la estructuración de los programas. En un sistema de múltiples procesadores no sólo es posible entrelazar la ejecución de múltiples procesos sino también solaparlas.

Tipos de concurrencia

Concurrencia aparente: Hay más procesos que procesadores.

- Los procesos se multiplexan en el tiempo.
- Pseudoparalelismo

Concurrencia real: Cada proceso se ejecuta en un procesador.

- Se produce una ejecución en paralelo.
- Paralelismo real.

Tipos de soluciones para la concurrencia

- Software
  - Algoritmos cuya correctud no dependen de ninguna suposición.
- Hardware
  - Dependen de algunas instrucciones maquina especiales.
- Operación del sistema
  - Provee algunas funciones y estructuras de datos al programador para preservar esto. Es decir, son soluciones que provee el sistema operativo.

Soluciones por software

- Algoritmo de Decker
- Algoritmo de Peterson
- Algoritmo de la panaderia

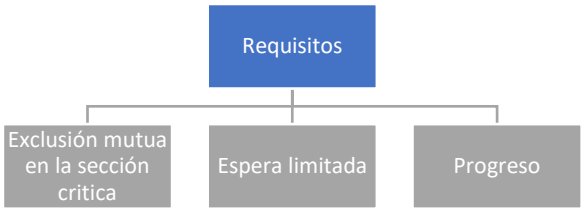
Soluciones por hardware

1era. Solución: Inhabilitación de interrupciones:

Si no hay interrupciones, no se invoca el planificador a corto plazo por lo que al proceso que está en ejecución no se le puede quitar el procesador. Con esta acción nos aseguramos que el proceso que se encuentra en la sección critica no la perderá temporalmente el uso del procesador, no se le asignará el procesador a ningún otro proceso.

2da solución: Instrucciones maquina especiales.

- El acceso a una posición de memoria excluye otros accesos a la misma posición.
- Los diseñadores han propuesto instrucciones máquina que ejecutan 2 acciones atómicas(indivisibles) en la misma posición de memoria.
- La ejecución de tales instrucciones es también mutuamente exclusiva incluso con varios CPUs.
- Pueden usarse para proveer exclusión mutua, pero necesitan complementarse con otros mecanismos para satisfacer los otros 3 requisitos del problema de la sección critica.



Instrucciones estudiadas:

- La instrucción test and set
- La instrucción Xchg

Soluciones del sistema

Sol1. Semáforos

Herramienta de sincronización que provee el sistema operativo que no requiere espera ocupada.

Un semáforo S es una variable que, aparte de la inicialización, solo se puede acceder por medio de 2 operaciones atómicas y mutuamente exclusivas:

Para evitar la espera ocupada cuando un proceso tiene que esperar, se pondrá en una cola de procesos bloqueados esperando un evento y de esta manera no usara la CPU

Tipos de semáforos

- Semáforo binario
  - Solo pueden tener dos valores, 0 y 1.
  - En Windows se llaman mutex
- Semáforo general o entero
  - Pueden tomar muchos valores positivos

Uso de semáforos para controlar procesos

En principio los semáforos binarios son más sencillos de implementar y tienen la misma potencia de expresión que los semáforos generales. Tanto en los semáforos como en los semáforos binarios se emplea una cola para mantener los procesos esperando en el semáforo. La política más equitativa mediante la cual se quitan los procesos de dicha cola es la FIFO. La única exigencia estricta es que un proceso no debe quedar retenido en la cola de un semáforo indefinidamente porque otros procesos tengan preferencia.

Generalmente operadores como WAIT y SIGNAL operan en los semáforos de la siguiente manera. Cuando un proceso ejecuta un operador WAIT que tiene un valor de semáforo en 0, ese proceso se bloquea; si el valor es mayor que cero, el valor del semáforo es disminuido en 1 y el proceso continua. Cuando un proceso ejecuta un operador SIGNAL y hay procesos bloqueados (WAITING), uno de estos procesos es activado (puesto en la cola de listos). Si no hay procesos esperando el valor del semáforo se incrementa en 1. Se asume que procesos bloqueados por semáforos pierden el procesador y entran en una cola de espera (WAITING QUEUE) en vez de producir BUSY WAITING. También se asume que la cola de espera es FIFO.