El Sistema Operativo y la Administración de Procesos

Prof. Aris Castillo de Valencia

2010

Tabla de Contenido

Objetivos:

- *Comprender el concepto de procesos y la administración de procesos.*
- *Describir los distintos estados por los que pasa un proceso*
- *Explicar el control que realiza el sistema operativo respecto a ellos.*

Repaso

Tipos de Sistemas Operativos:

- **Monolíticos**: *(Linux, Unix, FreeBSD)*
    - *Núcleo grande y complejo, engloba todos los servicios (planificación, sistemas de archivos, redes, controladores de dispositivos, gestión de memoria) del sistema.*
    - *No modular; mayor rendimiento que micro núcleo.*
    - *Cualquier cambio requiere recompilación del núcleo y reinicio del sistema.*
    - *Problemas para modificar y agregar nuevos servicios y funcionalidades.*
    - *Una ramificación del diseño es la de módulos ejecutables en tiempo de ejecución que brinde algunas ventajas del micronúcleo.*
    - *Ejemplos de Sistemas Operativos Monolíticos:*
        - *Tipo Unix: Linux, BSD, Solaris.*
        - *Tipo DOS: DIRDOS, MS-DOS incluye Win 9x (95, 98, me)*
        - *Tipo MAC: hasta MAC OS 8.6*
- **Micronúcleo**:

- *Contiene un conjunto de llamadas al sistema o primitivas mínimas, para implementar servicios básicos como espacios de direcciones, comunicación entre procesos y planificación básica.*

- *Todos los otros servicios (gestión de archivos de memoria, sistemas de archivos, operaciones E/S, etc), por lo general provistos por el núcleo, se ejecutan como procesos servidores en espacio de usuario.*

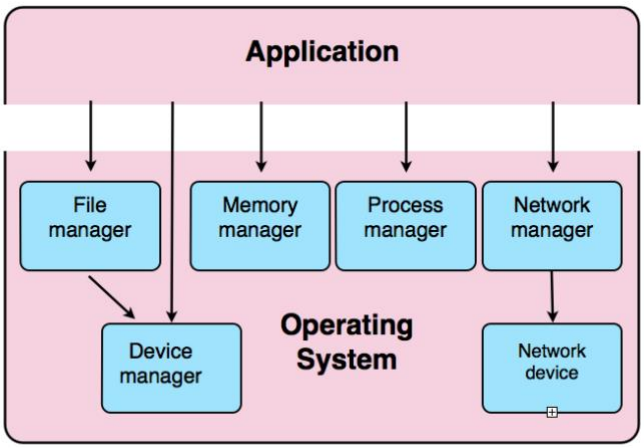
    - *Problema: sincronización de los módulos que componen el micronúcleo y su acceso a la memoria e integración con aplicaciones.*

    - *Ventaja: descentralización de fallos y fiabilidad para depurar y crear controladores de dispositivos. Ej: Amiga OS, Minix, Chorus, SymbOS, SO3, otros.*

- **Híbridos**: *micronúcleo que tienen algo de código no esencial en núcleo para que éste se ejecute más rápido que si se estuviese en espacio de usuario. Entran la mayoría de los sistemas operativos modernos. Ej: XNU (usado en MAC OS X), DragonFly BSD, React OS.*

Funciones del sistema operativo

Las funciones del OS están clasificadas en el núcleo, los servicios y el intérprete de comandos.



Los procesos del núcleo interaccionan directamente con el hardware de la máquina y tienen que ver con la gestión de recursos – procesador, memoria y tratamiento de interrupciones.

Los servicios se clasifican según – gestión de

procesos, gestión de memoria, gestión de Entrada/Salida, comunicación y sincronización de procesos, y la seguridad y protección.

La interfaz de llamadas al sistema o de servicios que utiliza el usuario directamente desde sus programas – POSIX y Win32.

El intérprete de comandos sea textual o gráfico.

- *Existen servicios para la creación de procesos, la ejecución de procesos, y la terminación de procesos.*
- *Los procesos básicos de administración de memoria son solicitar memoria, liberar memoria y compartir memoria.*

  En cuanto a la comunicación, sea síncrona o asíncrona, y la sincronización de procesos, los servicios incluyen la creación de mecanismos, la utilización de éstos y su destrucción.
- *En cuanto a la gestión de entrada y salida, los servicios incluyen crear archivo/directorio, abrir archivo/directorio, escribir y leer, cerrar archivo/directorio y borrar archivo/directorio.*
- *Los servicios en el estudio están relacionados con entrada / salida (la lectura de dispositivo externo y la escritura a disco duro), así como procesos de comunicación (navegación en Internet).*

Procesos

Qué es un proceso?

Proceso es cualquier instancia de un programa que deba ser ejecutada por el procesador. También se dice que es una entidad asignable y ejecutable. Puede constar de una sola traza de tareas o de varias.

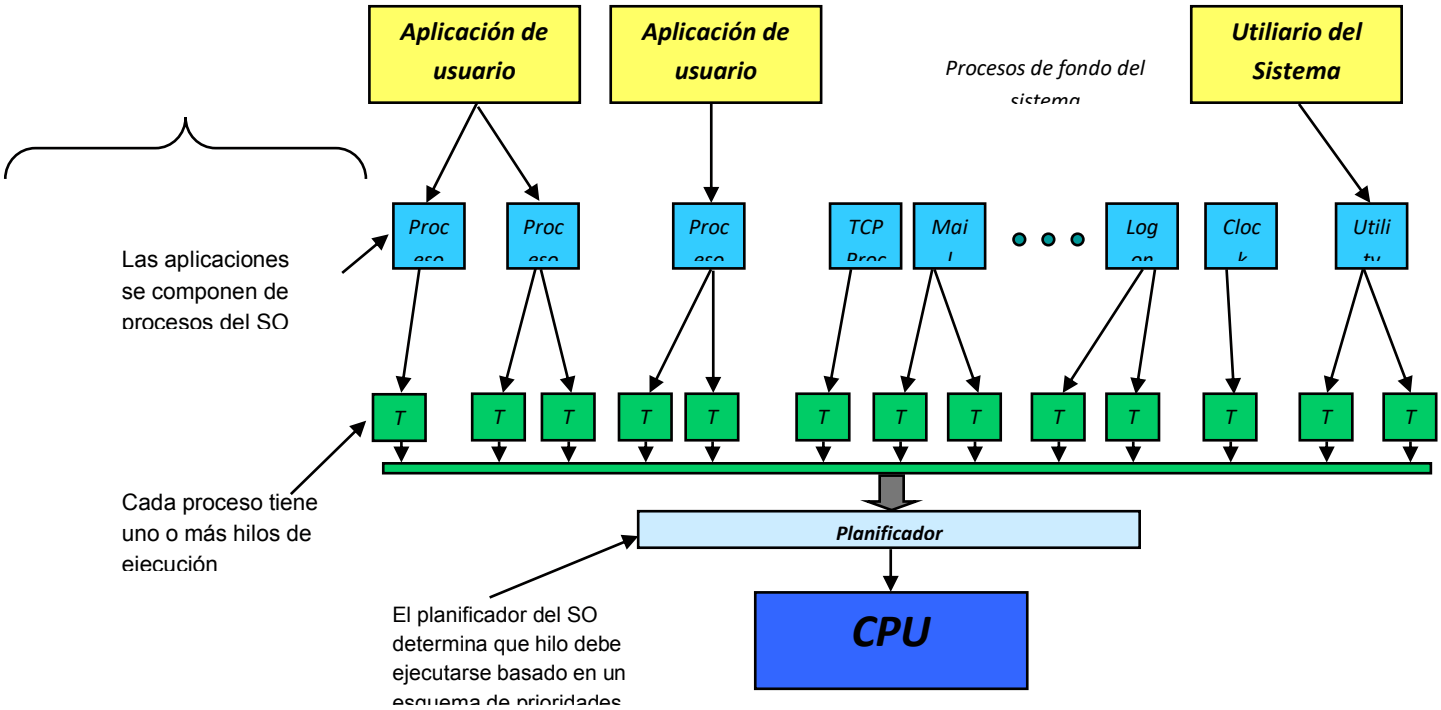
Cuál es la diferencia entre un procesador y un proceso?

El procesador es el componente físico que ejecuta las instrucciones de máquina que residen en memoria principal, generalmente en forma de procesos. El sistema operativo programa al procesador entre los diversos procesos activos en el sistema. Es decir, el procesador es un recurso administrado por el OS.

Un proceso puede ser controlado por usuarios, aplicaciones o por el sistema operativo. Entre sus componentes tenemos:

- *Un programa ejecutable*
- *Los datos asociados (variables, espacios de trabajo, buffers)*
- *El contexto de ejecución (información del OS para administrar el proceso y para que el procesador lo pueda ejecutar). Incluye registros – PC, IR, de datos, prioridades, y el estado del proceso.*

Muchos tienden a referirse a las aplicaciones como procesos, y aunque lo son, tienen la capacidad de iniciar otros procesos al ejecutar tareas como la comunicación con otros dispositivos u otras computadoras. También existen procesos que se ejecutan en segundo plano, es decir, sin intervención directa del usuario. Un ejemplo de estos procesos sería el manejo de la red, administración de memoria, virus, etc.



La figura muestra como el CPU es planificado entre los distintos procesos activos. Los procesos como se ve pueden ser desde una aplicación hasta alguna tarea rutinaria del sistema operativo.

Administración de Procesos

El sistema operativo es el encargado de crear los procesos y de administrarlos. La administración de procesos involucra tareas como creación, eliminación, asignación de espacios de memoria, programación para uso de CPU y de dispositivos de E/S.

Para mantener y dar seguimiento a un proceso, el sistema operativo utiliza una estructura de datos llamada Bloque de Control de Proceso (PCB, Process Control Block) donde guarda toda la información

necesaria sobre dicho proceso. Esta información se carga al CPU desde la memoria RAM cada vez que el proceso se deba ejecutar. El bloque de control de proceso contiene:

- *Un ID que identifica el proceso*
- *Punteros a memoria: incluye los punteros asociados al código de programa y los datos asociados a dicho proceso, además de cualquier bloque de memoria compartido con otros procesos*
- *Contenidos de Registros.*
- *Estado de los flags y switches.*
- *Una lista de archivos abiertos por los procesos.*
- *La prioridad del proceso.*
- *El estado de todos los dispositivos de entrada y salida necesitado por el proceso.*

La multiprogramación y las transacciones de tiempo compartido imponen la necesidad de las interrupciones, las cuales son señales enviadas por el hardware y el software al CPU. Esto es lo que permite que se pueda intercambiar el CPU entre diversos procesos activos. Por ejemplo, la suspensión de un proceso, como la ejecución de un programa del computador, ocasionada por un evento externo y realizado de manera que la ejecución del proceso pueda ser reanudada por medio del PCB.

Esto involucra que el procesador tenga que salvar el contexto del proceso y dividirse a la rutina de tratamiento de la interrupción, allí determina el tipo, la procesa y luego reanuda la ejecución del proceso interrumpido. Algunos problemas que se pueden dar son sincronización inapropiada, exclusión mutua falla, operación no determinista de programas, deadlock.

Existen dos tipos de interrupciones:

- *Enmascarado (masked): significa que el sistema operativo puede ignorar la solicitud de atención desde una fuente.*

- *No Enmascarable (non maskable interrupts, NMIs): significa que se debe atender inmediatamente la solicitud sin tener en cuenta el manejo de otras tareas.*

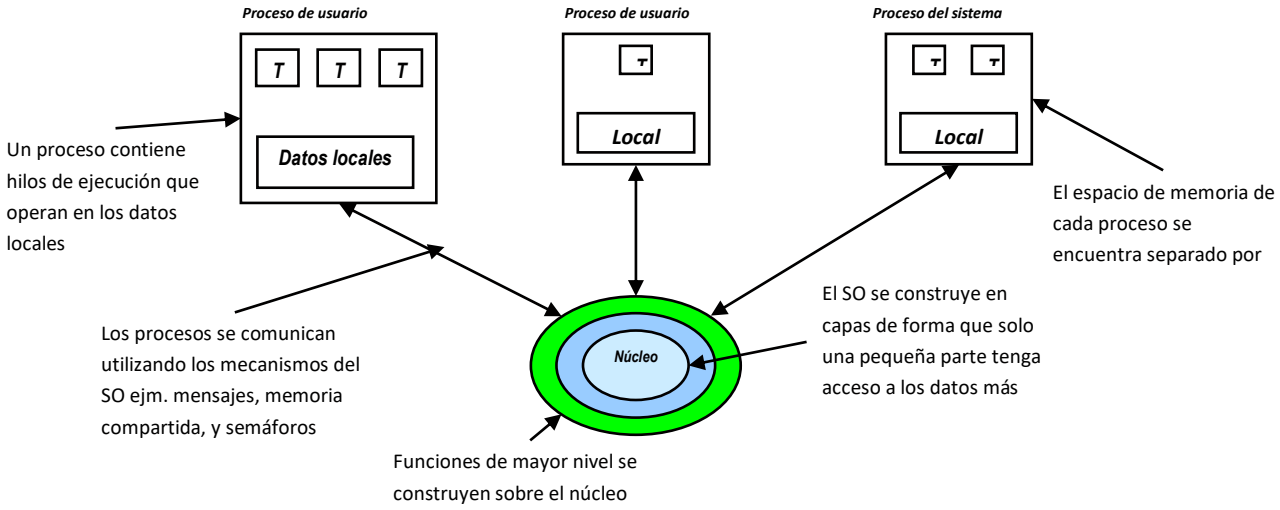
En un sistema monotarea el manejo de las interrupciones es bastante complicado debido a que solo puede ejecutar una tarea a la vez, por lo que, para procesar la interrupción, el sistema operativo tiene que intercambiar entre miles de procesos, miles de veces en un segundo. Este proceso se describe a continuación:

- *Un proceso ocupa una cantidad segura de RAM.*
- *Cuando dos procesos son multitarea, el sistema operativo permite un número de ciclos de ejecución para un programa.*
- *Después que el número de ciclos, el sistema operativo hace una copia de todos los registros, pilas y colas utilizados por los procesos y anota el punto en que el proceso en ejecución es pausado.*
- *Luego, se cargan todos los registros, pilas y colas utilizadas por el segundo proceso y permite un número de ciclos de CPU.*
- *Cuando se completa la carga, realiza una copia de todos los registros, pilas y colas utilizados por el segundo programa y carga el primer programa. Esto es lo que se conoce como intercambio de procesos.*

Existen procesos que no consumen tiempo de CPU hasta que son invocados por el usuario, mientras están esperando, el OS lo coloca en estado de suspendido. Cuando finalmente es invocado por el usuario, el OS cambia su estado.

Mientras el estado del proceso cambia, la información en el PCB debe ser utilizada como los datos en cualquier otro programa para direccionar la ejecución de la porción de intercambio de tarea del sistema operativo.

El proceso de suspensión puede ocurrir sin la intervención directa del usuario y cada proceso consigue suficientes ciclos de CPU para realizar su tarea en una cantidad razonable de tiempo. El problema puede venir si un usuario tiene muchos procesos funcionando al mismo tiempo ya que puede consumir la mayor parte de los ciclos del CPU para suspender los procesos en lugar de ejecutarlo. Cuando esto ocurre, es llamado thrashing y requiere de la intervención directa del usuario para detener los procesos y regresar el sistema a la normalidad. Una manera de combatir el problema del thrashing es reduciendo la necesidad de nuevos procesos para desempeñar varias tareas.

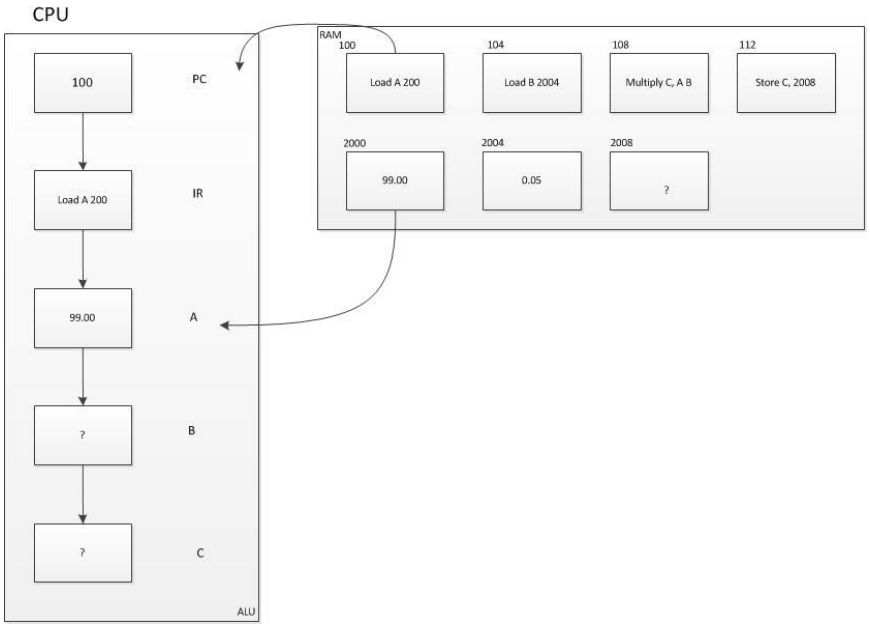


La figura muestra la interacción de los procesos con el Sistema Operativo.

Qué es el rastro de un proceso?

Se refiere a la lista de la secuencia de instrucciones ejecutadas por un proceso y que debe ser controlada por el sistema operativo para que sea posible la ejecución del proceso en diversos periodos de tiempo. Sin este rastro, el OS no sería capaz de manejar multitarea. Este concepto involucra la capacidad del OS de programar entre distintos procesos el uso del CPU, incluida la ejecución del OS como tal entre ellos.

Ejemplo:



1. *PC es 100. El CPU carga la instrucción en la dirección 100 al IR.*
2. *El CPU incrementa el valor del PC para la siguiente instrucción.*
3. *El CPU carga el contenido de la dirección 2000 en el registro A.*
4. *PC=104. El CPU carga la instrucción de la dirección 104 al IR.*
5. *El CPU incrementa el valor del PC para la siguiente instrucción.*
6. *El CPU carga el contenido de la dirección 2004 al registro B.*
7. *El PC=108; El CPU carga la instrucción 108 al IR.*

8. *El CPU incrementa el valor del PC para contener la siguiente instrucción.*

9. *El CPU usa el ALU para multiplicar el contenido del registro A y B y guardarlo en C.*

10. *El PC=112, El CPU carga la instrucción de la dirección 112 en el IR.*

11. *Se incrementa el valor de PC a la siguiente instrucción.*

12. *El CPU almacena el contenido del registro C a la dir 2008.*

Nota: Cada ciclo en ejecución tiene 3 pasos: obtener la instrucción, incrementar el PC y ejecutar la tarea.

Estados de los procesos

En general los procesos pueden transitar entre ***nueve estados o transiciones:***

1. *de nada a nuevo*

2. *de nuevo a listo*

3. *de listo a ejecución*

4. *de ejecución a terminado*
5. *de ejecución a listo*
6. *de ejecución a bloqueado*
7. *de bloqueado a listo*
8. *de listo a terminado (cuando el proceso padre termina, destruye al hijo)*
9. *de bloqueado a terminado*

Nuevo.

El proceso no está en memoria principal; sólo se está definiendo y creando. Tiene dos pasos: crear las tablas de control; identificadores son asociados a los procesos. En este estado el OS crea las estructuras de datos que se necesitarán para manejar los procesos y asigna un espacio de direcciones en memoria para el proceso. También le asigna un ID, inicializa el bloque de control de procesos basado en valores estándares y los atributos solicitados por el proceso y establece enlaces.

Un proceso puede crear otros procesos (Parent –Child) para brindar servicios de E/S, también cuando el usuario teclea algo, o cuando se inician varios trabajos.

Ejecución.

El proceso está usando el CPU. Esto significa que el PCB del proceso se pasa de la memoria principal a los registros del CPU.

Listo.

Cuando el proceso se crea entra en una cola de Listo. De vez en cuando el OS interrumpe un proceso y el despachador selecciona otro proceso para ejecutar, el proceso interrumpido se coloca en la cola de

Listo. El OS mantiene información para seguirle el rastro a cada proceso, por ejemplo estado y localidad en memoria.

El proceso en estado Listo está en memoria principal, esperando que el CPU esté libre. Cuando el OS tiene que seleccionar otro proceso, lo hace de la cola de listo usando FIFO o prioridad. Por otro lado, un proceso que esté en ejecución y se le acabó el tiempo (time-out) programado, entonces el OS lo pasa a la cola de Listo (estado Listo) a esperar nuevamente su turno de uso del CPU.

También este estado recibe a procesos de la cola de Bloqueado. Cuando algunos eventos suceden, aquellos procesos esperando en la cola de bloqueados son movidos por el OS a la cola de listos.

Bloqueado.

El proceso está en memoria principal. Está en espera de algún recurso del sistema. Mientras no obtenga éste, el sistema operativo no le cambiará su estado y por lo tanto no podrá ser programado para ejecución. Puede darse el caso de que existan varias colas de bloqueado. Esta forma es más eficiente ya que el OS no perderá mucho tiempo buscando eventos en una cola simple, sino que existe una cola para cada caso. Es eficiente para OS grandes.

Terminado.

Tiene dos pasos: terminación – los datos todavía están en el sistema, pero no es elegible para ejecución, aunque el OS mantiene sus tablas y datos. Cuando ningún otro proceso requiere del proceso, los datos relacionados y el proceso mismo son eliminados del sistema. La instrucción HALT genera una interrupción para anunciar al SO que el proceso ha terminado.

Cuándo se terminan los procesos?

Cuando el usuario finaliza una aplicación, el proceso padre termina; cuando el usuario hace log-off, cuando hay errores.

Suspendido.

El OS mueve algunos de los procesos a memoria secundaria (Intercambio o swapping) para que haya más espacio en memoria principal para nuevos procesos. Esto incrementa el desempeño del OS.

*En este caso hay dos posibles **estados**:*

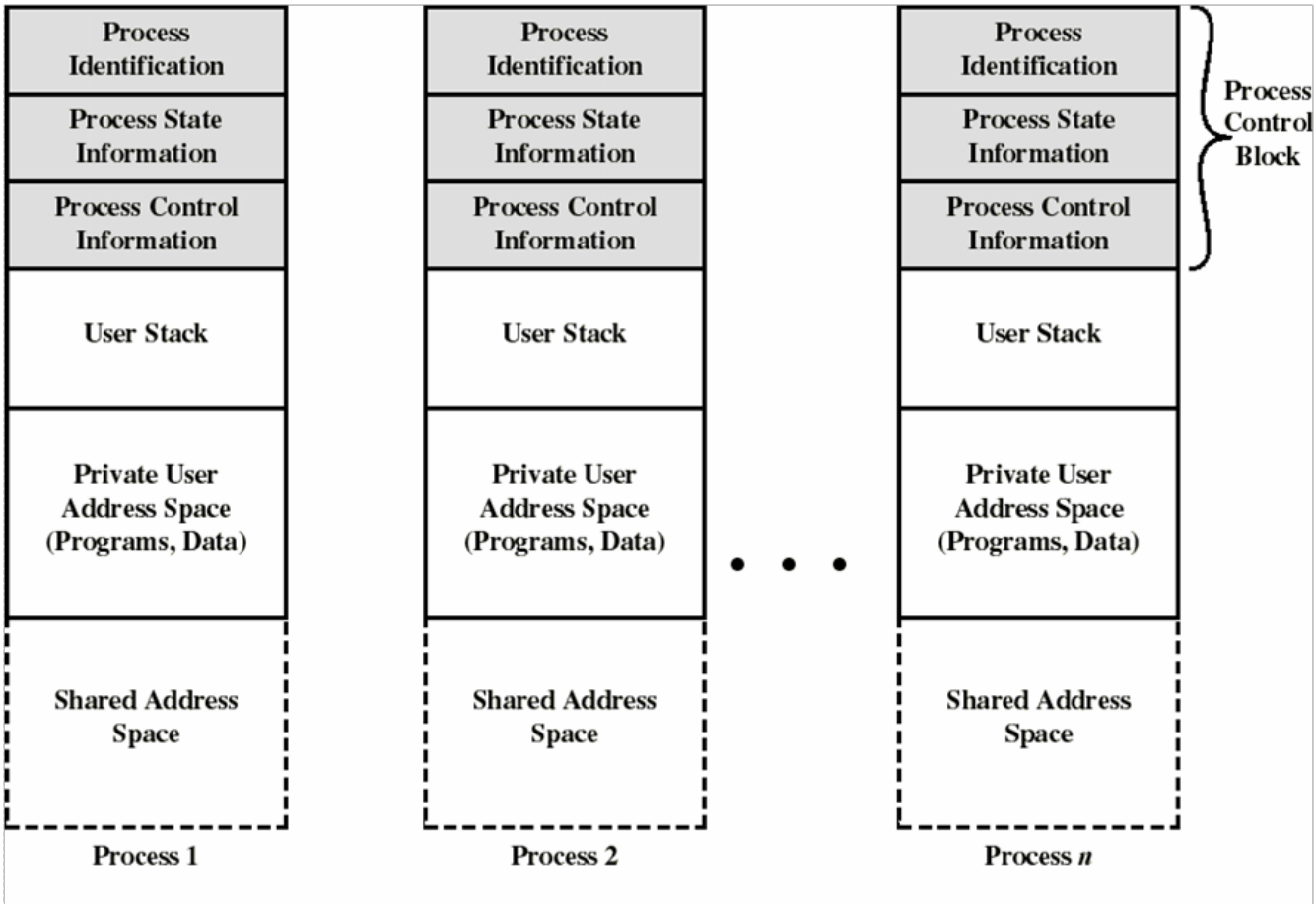
- *Bloqueado / suspendido: los procesos están en memoria secundaria, esperando por un evento.*
- *Listo / suspendido: los procesos están en memoria secundaria, pero está disponible para ejecución tan pronto como sea cargado en memoria principal, cuando se libere algún espacio.*

Descripción de procesos

***Tablas de Control de OS.** El OS maneja los recursos del sistema de los distintos procesos. ¿Qué necesita el OS para realizar esta función? Necesita información sobre el estado actual de cada proceso y sus recursos. Así crea las tablas de cada entidad. Todas las tablas (Memoria, E/S, archivos, procesos) deben estar relacionadas o enlazadas de manera que se pueda compartir información entre ellas.*

Para crear estas tablas el OS colecta la información del sistema cuando es inicializado para saber cuánta memoria hay, que dispositivos de E/S, etc. (VER PROCESO DE ARRANQUE DEL OS)

***Información sobre el proceso.** Para que el OS maneje y controle un proceso debe saber información específica de cada proceso: ubicación (memoria física y virtual), atributos de control del OS (PCB) – identificación (ID), estado, y control.*



Fuente: Stallings, Sistemas Operativos

El Bloque de Control de Procesos (PCB) es la parte esencial requerida para que el proceso pueda entrar en Ejecución, el resto de la imagen puede estar en disco. Contiene tres grupos de atributos de procesos, necesaria para que el OS lo pueda administrar:

- ***Identificación:*** *ID para identificarlo y para referencia cruzada entre tablas de los procesos. Puede haber ID del proceso, ID del proceso padre, y ID del usuario.*
- ***Estado:*** *incluye los registros visibles al usuario (instrucciones de lenguaje de máquina), registros de control y estado, punteros de pila. El PSW que tiene los códigos de condición y otra información del estado del proceso.*

- ***Control del proceso:*** *contiene información de planificación para que el OS coordine y controle los distintos procesos activos. Incluye las estructuras de datos, comunicación entre procesos, privilegios, propiedad de recursos y utilización.*

La imagen de un proceso es la colección de programa, datos, pila, y atributos que el sistema operativo construye para manejar y controlar el proceso. La imagen del proceso se mantiene en memoria secundaria generalmente y no necesariamente será un bloque continuo de información. La pila es usada para mantener una secuencia de las llamadas a procedimientos y paso de parámetros entre procedimientos.

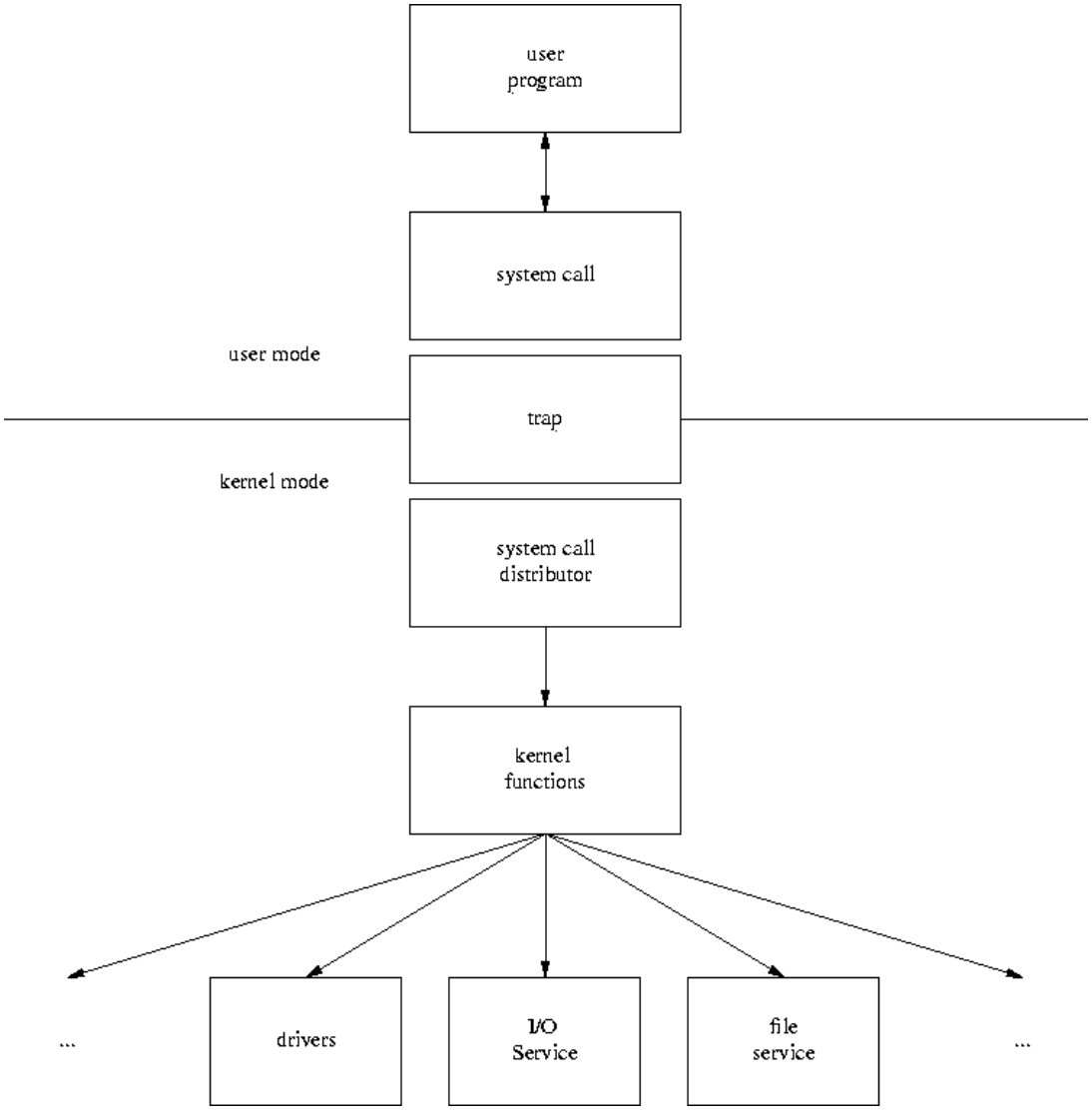
Cambio de Modo vs Cambio de Proceso

Los cambios de modo se realizan con el objetivo de proteger el PCB del proceso, el OS y las tablas de control de la interferencia de programas de usuario, ya que el PCB puede ser modificado por muchos módulos del OS – planificación, asignación de recursos, tratamiento de interrupciones, y rendimiento.

Se hace que todas las rutinas del OS pasen por una rutina de gestión (modo de sistema).

- **modo de usuario***: menos privilegiado*
- **modo de sistema** ***(control, núcleo):*** *en este modo se puede leer y escribir los registros de control, instrucciones primitivas de E/S, instrucciones de gestión de memoria, tratamiento de interrupciones. En modo núcleo el software tiene control completo del CPU y de todas sus instrucciones, registros y memoria.*

El modo puede cambiar mientras un proceso está en ejecución y se da por llamadas al sistema, interrupciones o por una excepción.



Fuente: System Calls [3]

El cambio de modo se da a través de un bit en el PSW (Program State Word) como respuesta al suceso. Por ejemplo, cuando el proceso hace una solicitud de servicio del OS, el modo cambia a núcleo. Cuando termina la rutina, el modo cambia nuevamente a usuario.

- *Algunos cambios de contexto no requieren un cambio de proceso total. Por ejemplo, cuando se retorna al mismo proceso después de una interrupción.*
- *No requiere guardar / restaurar totalmente el estado del nuevo / viejo proceso.*

- *Sólo el estado del proceso necesita ser guardado y restaurado.*

Incluye:

- *guardar el contexto*
- *configurar el contador de programa a la dirección inicial del manejador de interrupción*
- *cambiar de modo usuario a núcleo o viceversa*

Cambio de proceso

Un cambio de proceso significa que el proceso que está actualmente en el CPU pasará a la memoria y que otro proceso pasará a hacer uso del CPU. Puede ser causado por interrupciones o por una llamada de un proceso supervisor:

- ***Interrupción pura:*** *el manejador de interrupción hace algunas operaciones y luego la rutina del OS es llamada. Es producida por un suceso externo independiente del proceso en ejecución. Ocurre asincrónicamente y puede crear una señal a un evento que causa que otro proceso pase a Listo. Ejemplos:*
    - ***Interrupción de reloj:*** *es usada para señalar la culminación del quantum de tiempo. El OS podría pasar otro proceso de la cola de Listo al CPU para ejecución.*
    - ***Interrupción de Entrada/Salida (I/O):*** *señala que una operación de entrada/salida ha terminado. Puede causar que un proceso en estado Bloqueado pase a la cola de Listo. Puede ocasionar que siga el mismo proceso en Ejecución o que se cambie a otro de mayor prioridad.*
    - ***Fallo de memoria:*** *error del proceso. Puede ocasionar que el OS termine el proceso creador del fallo.*

- ***Trap (error):*** *el OS determina el error o condición dentro del proceso en ejecución, y toma acción para resolverlo o elimina al proceso. Es producida por la instrucción actual. Puede crear una señal de error. Cuando se da un trap puede ocurrir lo siguiente:*
    - *Cambio de modo del proceso, de usuario a kernel*
    - *El proceso de usuario es cambiado a proceso de kernel*
    - *El proceso adquiere privilegios especiales: modificar páginas de memoria, cambiar la prioridad.*
- ***Llamada del supervisor:*** *el proceso solicita una función del OS. Generalmente bloquea el proceso.*

En el ciclo de interrupción, el procesador verifica si una interrupción ha ocurrido. De no ser así el procesador entra al ciclo de FETCH, toma la siguiente instrucción del proceso actual. Si hay interrupción entonces el manejador de instrucción ejecuta algunas tareas básicas como resetear banderas para la interrupción, verifica si hay errores. Si la interrupción fue por reloj el manejador envía la interrupción al despachador para darle más tiempo al proceso.

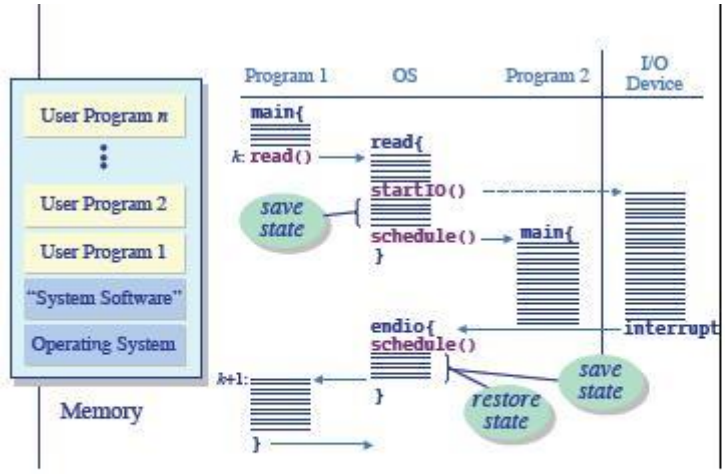
El cambio de proceso involucra:

- *Guardar el contexto del procesador (PC y los demás registros)*
- *Actualizar el PCB del proceso en Ejecución con su nuevo estado y con otra información asociada.*
- *Mover el PCB a la cola apropiada (Listo, Bloqueado, Suspendido, etc.)*
- *Seleccionar otro proceso para ejecución*
- *Actualizar el PCB del proceso seleccionado*

- *Restaurar el contexto del CPU del proceso seleccionado.*

Ejemplo de intercambio de procesos:

Proceso A = direcciones 5000 – 5011; Proceso B=8000-8003; Proceso C=12000-12011; OS (Dispatcher) = 100 - 105. (ver fig.3.3, Stallings)



- *se ejecutan 6 ciclos de instrucción del proceso A*
- *hay una interrupción (para evitar que un solo proceso se apodere del CPU)*
- *el despachador ejecuta su código – tareas administrativas de control (100-105)*
- *el proceso B entra en ejecución. El mismo llama un módulo de E/S, así que el proceso B se detiene en su cuarta instrucción*
- *el despachador ejecuta su código (100-105)*
- *sigue la ejecución del proceso C (6 instrucciones)*
- *el CPU para a C, entra el despachador y luego le da oportunidad a A para que continúe (A termina) y revisa también si el dispositivo de E/S terminó para darle respuesta a B.*

- *Finalmente continúa con las 6 instrucciones restantes de C.*

Cómo se aplicarían los estados de los procesos y los cambios de modo en este ejemplo?

Consideraciones de diseño de un sistema operativo multiprocesador:

Concurrencia: *ya que varios procesadores pueden ejecutar paralelamente el mismo código del núcleo, debe haber técnicas para evitar interbloqueos y operaciones inválidas.*

1. *Interbloqueo (deadlock): situación en que 2 o más procesos son incapaces de actuar porque cada uno está esperando que alguno de los otros haga algo.*
2. *Inanición (starvation) un proceso preparado para avanzar es soslayado indefinidamente por el planificador; aunque es capaz de avanzar, nunca se les escoge.*
3. *Círculo vicioso (livelock): dos o más procesos cambian continuamente su estado en respuesta a cambios en los procesos, sin realizar ningún trabajo útil.*

Planificación: *incluye técnicas para evitar que el mismo proceso sea elegido por varios procesadores.*

Sincronización: *incluye técnicas como el cerrojo y otras para forzar la exclusión mutua y el orden de eventos, ya que el paralelismo contempla compartir espacios de direcciones de memoria y otros recursos de E/S.*

- ***Exclusión mutua:*** *requisito para que cuando un proceso esté en una sección crítica que accede a recursos compartidos, ningún otro proceso puede estar en una sección crítica que accede a ninguno de esos recursos compartidos. Hay dos formas de implementar la exclusión mutua:*
  - *Por Hardware: se puede deshabilitar interrupciones o a través de instrucciones especiales de máquina, como test&set, exchange.*
  - *Por semáforos: mecanismo de sistema operativo y lenguaje de programación.*

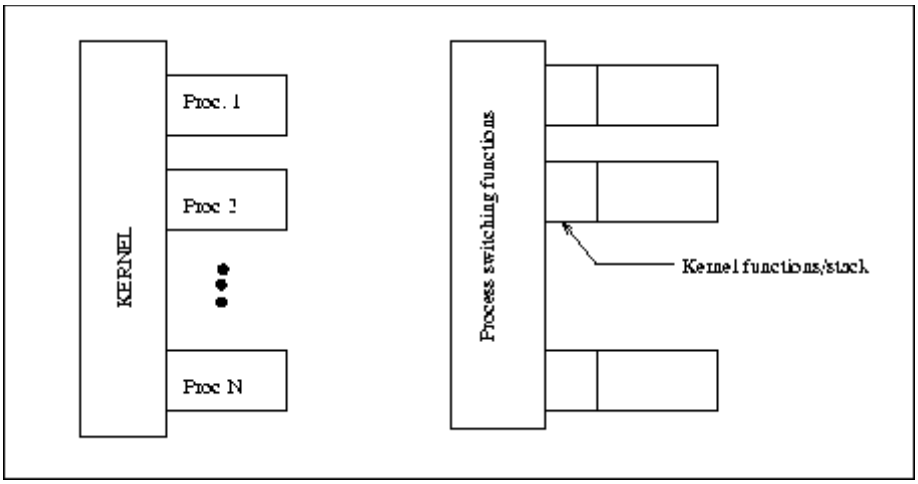
Gestión de memoria: *asegurar consistencia cuando varios procesadores comparten una página o segmento y por el reemplazo de los mismos.*

Fiabilidad y tolerancia a fallos: *acciones en caso de falla de un procesador.*

Ejecución del OS

El OS puede ser diseñado para ejecutar las funciones de administración y control de procesos de distintas formas:

- ***Núcleo separado:*** *el núcleo está fuera de los procesos, los procesos se aplican solamente a programas de usuario, el código del OS es ejecutado como una entidad separada en modo privilegiado.*
- ***Ejecución dentro de procesos de usuario:*** *casi todo el OS está dentro del contexto de procesos; el OS es una colección de rutinas que el usuario llama para realizar diferentes funciones ejecutadas dentro de los procesos de usuarios.*
- ***OS basado en procesos:*** *el OS es implementado como una colección de procesos del sistema; ventajas: es modular, las funciones no críticas son implementadas como procesos externos, bueno para sistemas multiprocesadores, incrementa el desempeño ya que la mayoría de las funciones son procesos independientes.*



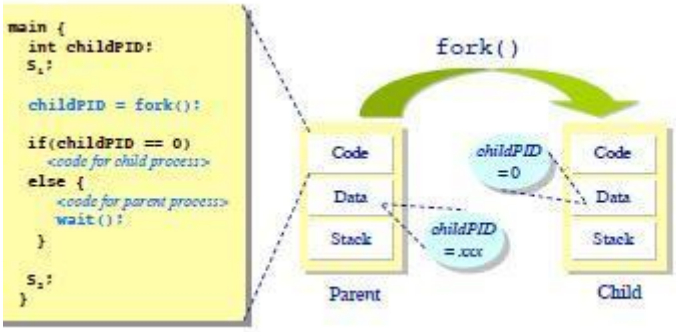
Fuente: Ejecución del OS [11]

La tendencia es de los sistemas operativos de kernel monolítico, donde el código se ejecuta fuera de todos los procesos, hacia kernels que se ejecutan mayormente en el contexto de los procesos mismos,

a microkernels que tienden a la modularización haciendo del sistema operativo mismo una colección de procesos especiales que operan entre sí.

Situaciones de Aprendizaje

Fork es la función de UNIX para crear un proceso hijo a partir del padre. El proceso hijo es una copia idéntica del proceso padre: variables y memoria y registros de CPU, excepto uno.



Use un editor de texto en Linux y el CLI para verificar la creación de procesos a través del código brindado. Luego abra otra sesión y ejecute los comandos top y ps.

Paso 1: Escriba el código del siguiente programa:

```
#include <sys/types.h>
#include <stdio.h>
main( )
{
        pid_t pid;
        int i;    int n = 10;

        for (i = 0; i < n;  i++)
        {
                pid = fork( );
                if (pid != 0)
        break;
        }
        printf("El padre del proceso %d es %d\n", getpid( ), getppid( ));
}
```

Paso 2: Compile el programa: gcc - o *programaejecutable programa.c*

Paso 3: Ejecutar el programa: ./programa

Paso 4: Registre y discuta los resultados.

Referencias bibliográficas

1. *Stallings, William. Sistemas Operativos. V Edición. Prentice Hall. 2000.*
2. *Carretero, Sistemas Operativos: Una visión aplicada. Mc-Graw Hill. Segunda edición.*
3. *System Call: http://www.cs.rit.edu/~hpb/Lectures/SIA/OS1/all-inOne-3.html*
4. *Process Control, architectural design issues: http://www.cim.mcgill.ca/~franco/OpSys-304-427/lecture-notes/node11.html*

Administración de Memoria – Cap. 7 y 8

La memoria son localidades de almacenamiento con una dirección para identificarla.

Tareas del OS en cuanto a administración de memoria

1. Dividir la memoria para alojar los procesos necesarios. El OS debe determinar la cantidad de bloques o particiones administrados. Pueden haber tres escenarios:
    a. Una partición – un solo proceso autoejecutable sin OS.
    b. Dos particiones – un proceso + OS. Es el caso de OS monoproceso.
    c. Múltiples particiones – una partición para el OS y varias para procesos. Es el caso de OS multitarea.

   Los mecanismos de asignación para control de memoria son: particionamiento, paginación, segmentación y memoria virtual.
2. Ubicar el proceso en memoria (asignarle espacio), es decir decidir dónde cargar el nuevo proceso. Algunos algoritmos son:
    a. Best fit: elige la zona libre más pequeña donde quepa la imagen del proceso. Objetivo – elegir el bloque que desperdicie menos. Requiere mucho tiempo de comprobación y ordenamiento.
    b. First fit: busca secuencialmente el primer bloque libre con tamaño suficiente para el proceso. Es el más eficiente.
    c. Next fit: busca el bloque a partir de la última asignación realizada.
3. Determinar cuando una página será llevada a memoria principal. Éstas son las técnicas de carga:
    a. Por demanda
    b. Prepaginación
4. Determinar qué página será llevada a memoria secundaria cuando una nueva página debe ser cargada a memoria principal. Éstos son los algoritmos de reemplazo:
    a. Óptimo
    b. LRU
    c. FIFO
    d. Clock
    e. Page buffering

Técnicas de asignación de memoria

1. Partición fija – obsoleta
2. Partición dinámica – obsoleta
3. Paginación simple – no se usa en solitario
4. Segmentación simple – no se usa en solitario
5. Memoria virtual paginada
6. Memoria virtual segmentada

Partición fija

Permitía dos formas de división de la memoria, a saber:

1. La memoria se divide en segmentos (particiones) fijas de igual tamaño. Cualquier proceso puede ser asignado a una partición aunque no use todo el espacio. Puede

que el proceso no quepa en la partición. Hay desperdicio o fragmentación interna (el programa o proceso es mucho más pequeño que el bloque; sobra espacio dentro de los bloques). Fig. 1

2. La memoria se divide en particiones fijas de distinto tamaño. La asignación de los espacios se puede realizar de dos formas:
    a. Hay colas por cada tamaño de manera que los procesos sean asignados de acuerdo con su tamaño. Puede no haber una partición libre de un tamaño X, pero sí de otros tamaños. Fig. 2ª
    b. Hay una sola cola de manera que los procesos sean asignados a un espacio adecuado o al que esté libre para evitar esperas. Fig. 2b

Suponiendo una RAM total de 512 MB

1	64 KB	64000*8 = 512 MB
2	64 KB	
3	64 KB	
4	64 KB	
5	64 KB	
6	64 KB	
7	64 KB	
8	64 KB	

Fig. 1

Suponiendo una RAM total de 512 MB

Colas X partición

1	25 KB	Total = 512 MB
2	34 KB	
3	40 KB	
4	50 KB	
5	75 KB	
6	60 KB	
7	100 KB	
8	128KB	

Fig. 2ª

Una sola cola

1	25 KB	Total = 512 MB
2	34 KB	
3	40 KB	
4	50 KB	
5	75 KB	
6	60 KB	
7	100 KB	
8	128KB	

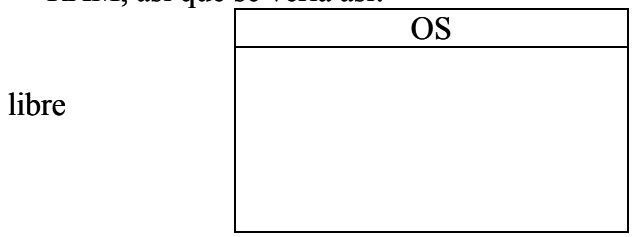
Fig. 2b

Partición dinámica

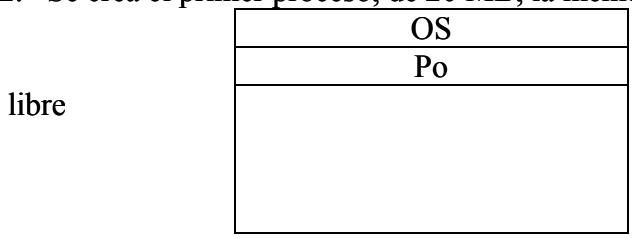
- Las particiones se crean de acuerdo con el tamaño del proceso nuevo.
- Problema: se crean muchos huecos entre bloques o fragmentación externa.
- A través de compactación (desplazamiento de procesos que estén contiguos de forma que queden bloques libres más grandes) se resuelve este problema, pero esta operación consume tiempo; desperdicia tiempo del CPU.

Ejemplo con una memoria principal de 64 MB:

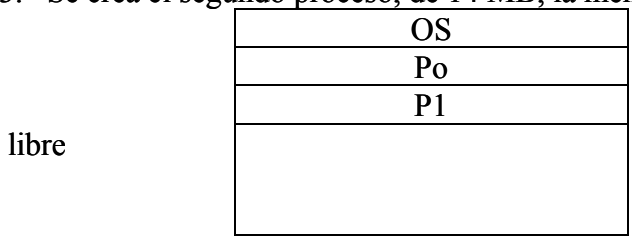
1. En momento recién iniciado el sistema, sólo el OS está en la memoria RAM, así que se vería así:



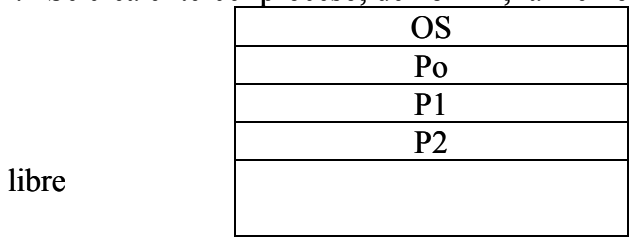
2. Se crea el primer proceso, de 20 MB, la memoria se vería así:



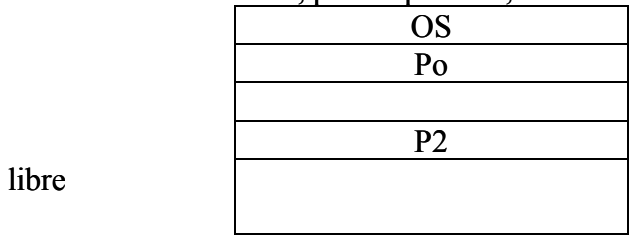
3. Se crea el segundo proceso, de 14 MB, la memoria se vería así:



4. Se crea el tercer proceso, de 18 MB, la memoria se vería así:



5. El proceso 1 se termina o suspende; quedando un segmento de de 14 MB y otro de 4 MB libres, pero separados; la memoria se vería así:



6. Se crea el proceso 3, de 8 MB; quedando un segmento de de 6 MB y otro de 4 MB libres, pero separados, así que sólo procesos de esos tamaños se podrían crear; la memoria se vería así:

	OS	8 MB ocupado
	Po	20 MB ocupado
	P3	8 MB ocupado
		6 MB libre
	P2	18 MB ocupado
libre		4 MB libre

7. P0 termina; quedando tres segmentos libres – 20 MB, 6 MB y 4 MB separados, la memoria se vería así:

	OS	8 MB ocupado
		20 MB libre
	P3	8 MB ocupado
		6 MB libre
	P2	18 MB ocupado
libre		4 MB libre

8. El P1 de 14 MB regresa a la memoria principal; quedando tres segmentos libres – 20 MB, 6 MB y 4 MB separados, la memoria se vería así:

	OS	8 MB ocupado
	P1	14 MB ocupado
		6 MB libre
	P3	8 MB ocupado
		6 MB libre
	P2	18 MB ocupado
libre		4 MB libre

Se requiere compactación para hacerlos más usables; 16 MB.

Ubicación / Reubicación

Como los procesos pueden ser cargados en espacios distintos de memoria cada vez, las ubicaciones de datos e instrucciones cambian, por tanto se requiere un proceso de traducción de las direcciones de los procesos. Hay tres referencias a la dirección de un proceso en memoria:

1. Dirección física – absoluta, es la dirección real en memoria.
2. Dirección lógica – es una referencia a una posición de memoria independiente de la asignación actual de datos a memoria. Requiere traducción a una dirección física.
3. Dirección relativa – es una dirección lógica que se da respecto a un punto conocido, normalmente con respecto al inicio del programa.

Los programas o procesos se cargan en memoria a través de un cargador dinámico, o sea que las direcciones se calcularán con respecto al origen del programa en el área de memoria en el que quedó ubicado, usando un mecanismo de hardware. Proceso:

1. Se le asigna la dirección al registro base (inicio) y al registro límite (posición final del programa) en el momento que el programa se carga a la memoria.
2. Todas las demás direcciones de referencia (registros de instrucción, bifurcación, datos, etc.) se deben verificar antes de ejecutar la instrucción para asegurarse que están en el área correcta. Sino se envía un trap.
    a. BP + Relative address = Absolute address
    b. BP + Relative Address < Stack Pointer
    c. Si BP + Relative Address > Stack Pointer ➔ se envía un trap, sino se le da la dirección absoluta para continuar con la ejecución de la instrucción. Ver Fig. 4

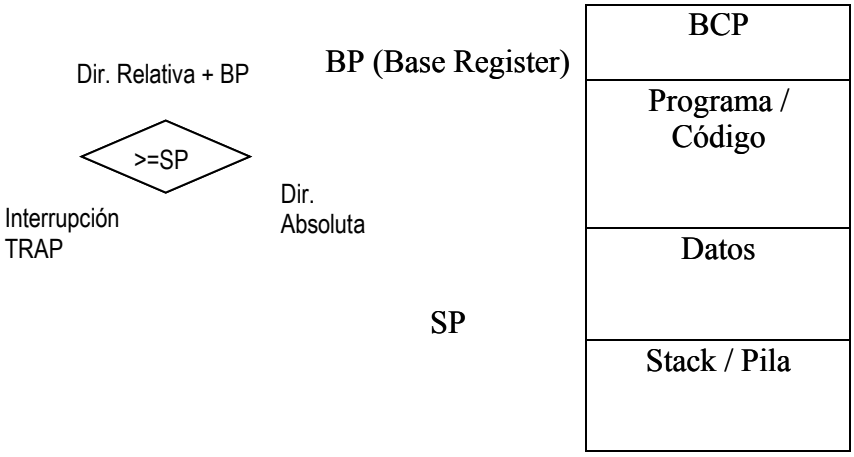


Fig. 4 – Imagen del Proceso

Técnicas de carga

Por demanda: consiste en iniciar la ejecución del proceso sin páginas cargadas; éstas se irán cargando conforme el proceso haga referencia a una posición en dicha página. La tasa de fallos de página es alta al iniciar la ejecución del proceso; luego se estabiliza.

Prepaginación

El OS predice las páginas que serán utilizadas durante la ejecución del proceso. Entnces trae páginas contiguas a la del fallo para evitar traer una a una, lo cual tomaría más tiempo. No es eficiente si las páginas extra no se referencian.

Políticas de reemplazo

1. Óptima: descarga la / las páginas que no se necesitarán en el tiempo más largo en el futuro. Imposible porque se basa en el futuro.
2. LRU (Least Recently Used): descarga la página menos usada recientemente. Por principio de cercanía, ésta tendrá menos probabilidad de ser referenciada en el futuro.

2	3	2	1	5	2	4	5	3	2	5	2
2	2	2	2	2	2	2	2	3	3	3	3
	3	3	3	5	5	5	5	5	5	5	5
			1	1	1	4	4	4	2	2	2
				F		F		F	F		

3. FIFO (First In First Out): reemplaza la página que ha estado más tiempo en memoria. produce errores porque supone que una página introducida hace mucho tiempo pudo haber caído en desuso, lo cual no siempre es cierto.

2	3	2	1	5	2	4	5	3	2	5	2
2	2	2	2	5	5	5	5	3	3	3	3
	3	3	3	3	2	2	2	2	2	5	5
			1	1	1	4	4	4	4	4	2
				F	F	F		F		F	F

4. Clock: una un bit de uso, que se pone en cero (0) la primera vez que se carga la página en memoria y en uno (1) cuando se haga referencia a esta página (cuando hay un fallo de página). Un puntero va pasándose/apuntando al siguiente marco. La primera página encontrada con el bit de uso = 1 se cambia a cero (0). Así se da toda la vuelta y la primera con el bit en cero se reemplaza.

2	3	2	1	5	2	4	5	3	2	5	2
2*	2*	2*	2*	5*	5*	5*	5*	3*	3*	3*	3*
	3*	3*	3*	3	2*	2*	2*	2	2*	2	2*
			1*	1	1	4	4*	4	4	5*	5*
				F	F	F		F		F	

*Bit u=1
Sombra = puntero

5. Page buffering (almacenamiento intermedio):
    a. Usa FIFO.
    b. La página reemplazada se mantiene físicamente en memoria principal, ya sea en la lista de páginas libres o en la de páginas modificadas.
    c. Si el proceso vuelve a hacer referencia a la página, demora menos el activarla y agregar su entrada en la tabla de páginas.
    d. Estas listas son "caché de páginas."

El Sistema Operativo, Tipos, Funcionamiento y Componentes

Prof. Aris Castillo de Valencia
2010

Objetivos:

- Discutir las funciones principales de los sistemas operativos
- Diferenciar los tipos de acuerdo con su operación y funciones
- Evaluar los componentes del sistema operativo
- Diferenciar las estructura de los sistemas operativos
- Características de los sistemas operativos modernos
- Describir el proceso de arranque del Sistema Operativo

Tabla de Contenidos

Definición

Un sistema operativo es un programa que controla la ejecución de programas de aplicación y que actúa como interfaz entre las aplicaciones del usuario y el hardware del computador.

Principales funciones del sistema operativo

Por áreas de servicio

- Desarrollo de programas – ej. editores, debuggers.
- Ejecución de programas – incluye acciones como cargar a memoria, acceso a archivos.
- Acceso a dispositivos de E/S – ejecutar instrucciones y señales de control propias de cada dispositivo.
- Acceso controlado a archivos – compresión, estructura de datos en los archivos, medio de almacenamiento.
- Acceso al sistema – provee protección a recursos y datos.
- Detección y respuesta a errores – errores internos y externos de hardware, acceso a memoria, desbordamiento, etc.
- Contabilidad - estadísticas, monitoreo de rendimiento del sistema.

Por recurso controlado

- La memoria es controlada por el sistema operativo y por el hardware de manejo de memoria en el procesador.
- El OS controla el acceso y utilización de dispositivos de E/S, acceso y utilización de archivos, así como el tiempo de uso del procesador.
- La porción del sistema operativo en memoria incluye el kernel (que contiene las funciones más frecuentemente usadas) y en ciertos momentos, incluye otras partes del OS que estén en uso.

Tipos de Sistemas Operativos

Según el tipo de computadora que controlan y las aplicaciones que soportan, los sistemas operativos se pueden clasificar en cuatro tipos:

Sistema Operativo de tiempo real (RTOS): es un sistema operativo que ha sido desarrollado para aplicaciones de tiempo real y utilizado para controlar instrumentos científicos y sistemas industriales. El objetivo no es un rendimiento alto, sino garantizar el cumplimiento de una tarea. Puede haber dos tipos de sistemas operativos de tiempo real:

- Tiempo real duro (Hard-real time) en que los plazos se alcanzan determinísticamente.
- Tiempo real suave (Soft-real time) en que el objetivo alcanzar un plazo.

Algunas de las filosofías de diseño de estos sistemas operativos incluyen:

- El sistema operativo funciona por eventos, lo que significa que se realiza intercambio de tareas sólo cuando un evento de mayor prioridad requiere servicio, llamado prioridad preferente (preemptive).
- El diseño es de tiempo compartido (time-sharing) lo que significa que las tareas se van intercambiando de acuerdo con interrupciones regulares de reloj o por eventos, generalmente a través de algoritmos Round Robin.

Inicialmente, los diseños de CPUs requerían muchos ciclos para el intercambio tareas, lo que significaba que el CPU no estaba realizando algo útil. Por ejemplo, un procesador 68000 de 20 MHz el intercambio de tareas se realizaba cada 20 microsegundos; mientras que con un procesador de 100 MHz se realiza cada 3 milisegundos. Por esta razón, los diseños iniciales trataban de reducir el intercambio de tareas innecesario.

Algunos ejemplos son:

a. **QNX.** Es un sistema operativo comercial tipo Unix, principalmente en el mercado de los sistemas embebidos. Tiene licencia propietaria. Es un sistema microkernel donde la mayor parte del OS corre como un conjunto de tareas pequeñas conocidas como servidores. Esto difiere de los sistemas monolíticos en que el OS es un gran programa compuesto de un gran número de partes con habilidades especiales. En el caso de QNX, los usuarios pueden desactivar funciones que no están necesiten sin tener que

cambiar el OS mismo. Dichos servidores simplemente no corren. Corre en plataformas PowePC, familia x86, MIPS, SH-4, ARM, StrongARM y XScale. [3]

b. **RTLinux.** Es un sistema operativo tiempo real duro y microkernel que corre Linux por completo como un proceso preferente. Es basado en una máquina virtual ligera (light weight virtual machine) donde el huésped Linux tiene un controlador de interrupción y un temporizador virtualizados, mientras que todos los demás accesos al hardware son directos. Es de licencia abierta (open source) y programado en lenguaje C. [5]
c. **WindowsCE.** Su nombre oficial es Windows Embedded Compact. Es desarrollado por Microsoft usando lenguaje C y su licencia es propietaria. Corre en procesadores Intel x86, MIPS, ARM e Hitachi SuperH. Es de 32 bits y permite integrar capacidades de tiempo real con tecnologías Windows. [6]

Monousuario – Monotarea**:** está diseñado para administrar la computadora de manera que un solo usuario puede ejecutar una tarea a la vez. Ejemplo: Palm OS para computadoras Palm portátiles, DOS de Microsoft. En este caso el sistema operativo no es capaz de realizar más de una tarea en un tiempo determinado. Esto significa que no hay intercambio de tareas en el CPU y que la memoria no requiere ser compartida. En este caso la memoria estaría dividida en dos bloques – uno para el sistema operativo y otro para el proceso activo.

Monousuario – Multitarea**:** es ampliamente utilizado ya que permite a un usuario ejecutar varias tareas al mismo tiempo. Sin embargo, no tiene la capacidad de más de una sesión de usuario a la vez. Ejemplo: las plataformas Microsoft Windows y sistemas operativos MAC de Apple.

Este tipo de OS es más complejo que el monousuario-monotarea ya que requiere capacidad para administrar diversos procesos simultáneamente, lo que implica asignación de identificadores, espacios de memoria privados y compartidos, y planificación de recursos tales como el CPU y los dispositivos de E/S.

Multiusuario**:** permite a diferentes usuarios tomar ventajas de los recursos de la computadora de forma simultánea. Para que este sistema operativo funcione de manera correcta se debe asegurar que los requerimientos de los usuarios estén balanceados de manera que si se presenta algún inconveniente con un usuario, no afecte a los demás usuarios del sistema. El hecho de ser multiusuario implica necesariamente que sea multitarea. Ejemplos: Sistema Operativo Unix, Mainframes, Linux y de Windows Vista en adelante.

Sistema Operativo **Monoprogramado y Multiprogramado vs Monoprocesador y Multiprocesador.**

Desde el punto de vista de las tareas que el sistema operativo es capaz de realizar, se habla de:

- Monoprogramación: modo de operación que solo permite la ejecución de una tarea o programa en un único procesador.
- Multiprogramación: modo de operación permite la ejecución intercalada de dos o más tareas o programas en un único procesador.

Utilización de los procesadores.

Desde el punto de vista, del número de procesadores reconocidos por el sistema operativo, se habla de:

- Sistemas operativos monoprocesador (un solo CPU)
- Sistemas operativos multiprocesador (dos o más CPUs).

Tipos de sistema operativo multiprocesador

Un sistema operativo multiprocesador debe dividir el trabajo de carga entre los CPUs para balancear las demandas de los procesos requeridos con los ciclos disponibles en los diferentes CPUs. Entre los diferentes tipos tenemos:

- Sistema Operativo Asimétrico (AMP)**:** utilizan un CPU para sus propias necesidades (las del OS) y dividen lo procesos de aplicación entre los CPUs restantes.
- Sistema Operativo Simétrico (SMP)**:** las tareas del OS y los procesos de usuario se dividen entre los distintos CPUs, balanceando la demanda contra la disponibilidad del CPU.

Componentes de los Sistemas Operativos

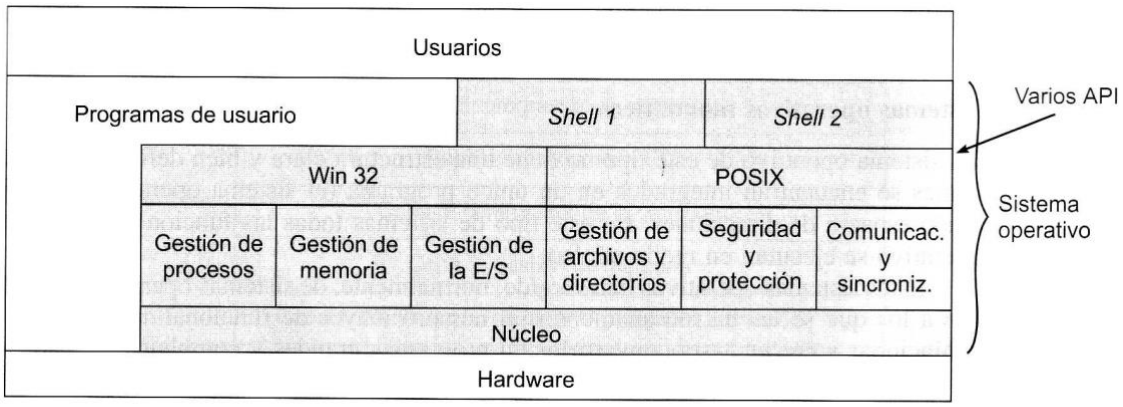


Figura 2.5. *Componentes del sistema operativo.*

Fuente: Carretero, Pag. 41
Veamos cada uno de estos componentes.

Shell

Es una pieza de software que provee una interfaz para los usuarios del sistema operativo, de manera que puedan tener acceso a los servicios del kernel. Estas piezas de software interpretan comandos. Generalmente hay dos categorías:

- Línea de comandos**:** proveen una interfaz de línea de comandos (CLI, command line interface) al sistema operativo. Algunos ejemplos son los shells de Unix tales como Bourne Shell (sh), Bourne-again Shell (bash), Korn Shell (ksh), C shell (csh), entre otros. Shells no Unix incluyen cmd.exe que es el principal shell de OS/2, Windows CE y Windows NT; command.com es el Shell para varios DOS de Windows; Google Shell que es el utilizado en Google Search.
- Gráfico**:** proveen una interfaz gráfica al usuario (GUI, Graphical User Interface). En el caso de sistemas X Windows, existen manejadores X Windows independientes y ambientes desktop completos que dependen de un manejador de ventanas. Ejemplos del primer caso son Blackbox y Fluxbox. En el segundo caso están CDE, GNOME, KDE, Xfce y LXDE. Para otras plataformas, existen otros Shells. [10]

Es importante anotar que también existen shells para Lenguajes de Programación.

- API – Application Program Interface.

  Es una abstracción que describe una interfaz para la interacción entre un conjunto de funciones utilizadas por los componentes de un sistema de software. En otras palabras, las API permiten a los programadores de aplicaciones usar funciones de la computadora y del sistema operativo sin necesidad de seguir los detalles de operación del CPU. El software que provee las funciones descritas por un API se dice que es la implementación del API.

  WIN 32 y POSIX son implementaciones de API. El estándar POSIX define un gran conjunto de funciones computacionales a ser escritas de forma tal que operen en diferentes sistemas. MAC OS X y varias distribuciones BSD implementan esta interfaz, sin embargo, se requiere recompilar el sistema para usarlas. Las API de Microsoft, Windows API o Win32 corren en nuevas versiones a través de Compatiblity Mode. Las API compatibles permiten que no se requieran cambios en el sistema, para lo cual se requieren bibliotecas (Dynamic Link Library, DLL) que permiten las llamadas de las aplicaciones de usuario (user applications) al sistema [11].

- Componente Gestión de Procesos:

  Implica todos los aspectos administrativos que debe realizar el OS para asegurar que varios procesos puedan ser ejecutados en el sistema.

  - Se debe asegurar que cada proceso y aplicación reciba la cantidad suficiente de tiempo del procesador para funcionar correctamente.
  - Se debe permitir la utilización de muchos ciclos de procesos para trabajos reales.

Componente de Gestión de memoria y Almacenamiento

- Cada proceso debe tener suficiente memoria para ejecutar, y tampoco puede correr dentro del espacio de otro proceso.
- Los diferentes tipos de Memoria en el sistema deben ser utilizados adecuadamente.

Incluye aislamiento de procesos (configurar los límites de memoria), que los programas sean alojados dinámicamente en la jerarquía de memoria, soporte de programación modular (programadores pueden crear, destruir, y cambiar el tamaño de módulos dinámicamente), protección y control de acceso, almacenamiento a largo plazo.

El sistema de archivo implementa un almacenamiento a largo plazo, con información almacenada en archivos en memoria secundaria. La memoria virtual permite a los programas direccionar memoria desde un punto de vista lógico, con esto se permite tener más procesos activos concurrentemente residentes entre la memoria principal y la secundaria porque los procesos no requieren tener todas sus páginas en memoria principal. Generalmente, usando un sistema de paginación (los procesos están compuestos de un número de bloques de tamaño fijo llamados páginas.

Para que un programa de usuario corra algunas de sus páginas deben estar en memoria principal. Para el intercambio de datos se usan direcciones virtuales y reales, donde el mapper (MMU- Memory Management Unit) hace la traducción. Stallings, p. 74

Según su velocidad, las memorias se clasifican en memoria caché, memoria principal (RAM), memoria secundaria (almacenamiento por rotación magnética). El sistema operativo debe mantener un balance entre las necesidades de los diversos procesos con la disponibilidad de los diferentes tipos de memoria, moviendo los datos en bloques (llamados páginas) entre la memoria disponible [9].

- Componente de Administración de Dispositivos de E/S:

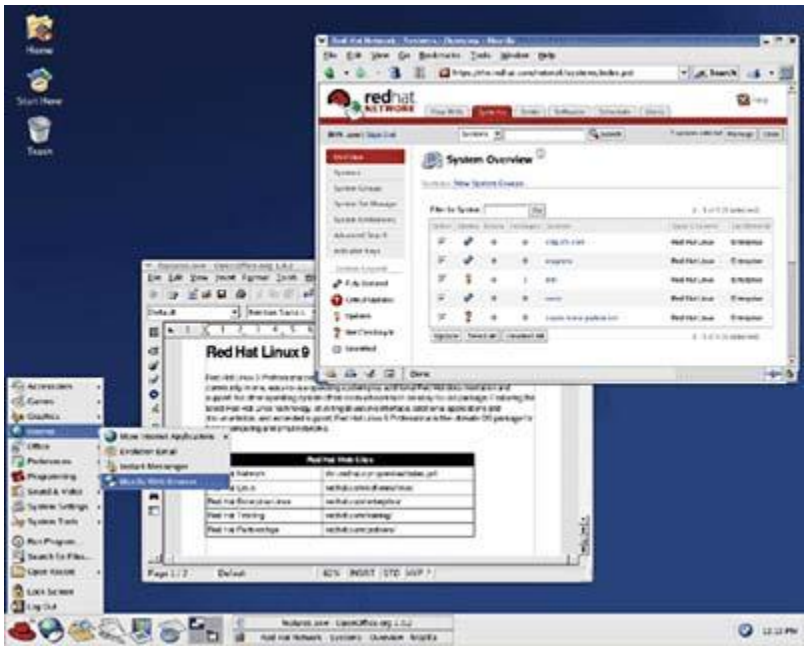
  La ruta entre el sistema operativo y virtualmente todos los componentes de hardware en la tarjeta madre va a través de programas especiales denominados manejadores (drivers). La función principal de un driver es servir de traductor entre las señales eléctricas del hardware y el lenguaje de programación del sistema operativo y los programas de aplicación. Un driver provee una forma en que las aplicaciones puedan hacer uso de subsistemas de hardware sin necesidad de conocer los detalles de operación del mismo.

  La razón por la que los driver son separados del sistema operativo es porque nuevas funciones pueden ser añadidas al mismo de este modo a los subsistemas del hardware sin requerir la modificación del sistema operativo.

La administración de todos los recursos del computador es una gran parte de la función del sistema operativo, y en algunos casos de sistemas operativos en tiempo real [9].

Interfaz de usuario (UI):

Brinda la estructura para la interacción entre un usuario y la computadora. En las últimas décadas casi todo el desarrollo de interfaces de usuario se ha enfocado en el área de interfaz de usuario grafico (GUI), destacándose los modelos Apple´s Macintosh y Microsoft Windows creciendo más atención y ganancia, pero el más popular es el sistema operativo Linux de código abierto que también soporta la interfaz de usuario gráfico.



La base de la función de sistema operativo y la administración del sistema de la computadora está en el kernel del sistema operativo. El administrador de imágenes es separado pensando que podía estar ligado debajo del kernel. Este enlace entre el kernel del sistema operativo y la interfaz de usuario, utilidades entre otros software definen muchas de las diferencias de los sistemas operativos de hoy.

Proceso de Arranque del Sistema Operativo:

Al encender nuestra computadora, el primer programa que corre es un conjunto de instrucciones mantenidos en la Memoria de Solo Lectura (ROM) de la computadora. Este

código examina el hardware del sistema se asegura de que esté funcionando correctamente.

Este POST (power-on self test) revisa el CPU, la memoria y el sistema básico de entrada y salida (BIOS) en busca de errores y almacena el resultado en una localización de memoria especial. Una vez que el POST se completa de manera exitosa, el software ROM (BIOS) empezará a activar los controladores de disco de la computadora. Una vez esto sucede, se encuentra la primera pieza del sistema operativo en disco: el bootstrap loader. El bootstrap loader es un pequeño programa que tiene la función de cargar el sistema operativo a la memoria y le permite iniciar operación. El boostrap configura los drivers y los subsistemas de la computadora; establece las divisiones de memoria para el OS, información de usuario y aplicaciones; establece las estructuras de datos para las señales, banderas y semáforos que se usan para la comunicación entre subsistemas y aplicaciones. Luego de esto, se pasa el control de la computadora al sistema operativo.

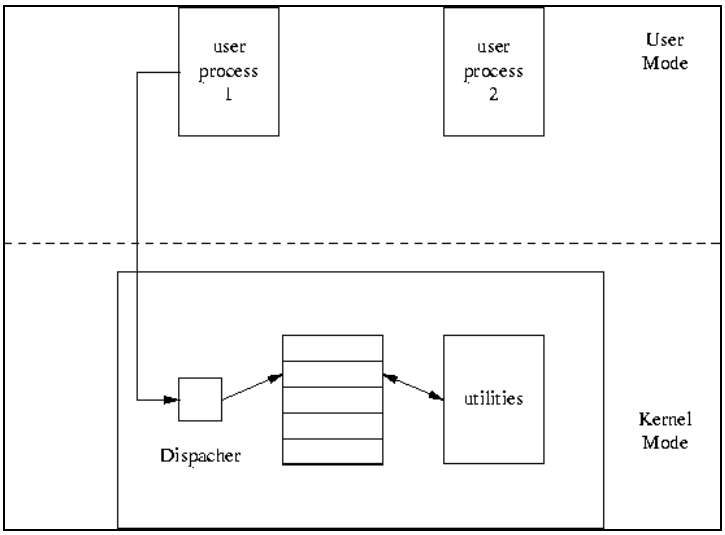
Estructura de los sistemas operativos:

Monolítico

Es la arquitectura más antigua. Un núcleo **monolítico** significa un único proceso muy grande con todos sus elementos integrados en un mismo espacio de memoria; todas las funciones del sistema operativo se ejecutan en modo núcleo.

Características:

- Un programa enorme
- No hay información oculta
- Llamadas supervisor
- Modo usuario vs modo núcleo (kernel)



Fuente: [12] Arquitectura monolítica

Cómo funciona?

1. Un proceso realiza una llamada al sistema – modo usuario a modo kernel
2. Revisión de parámetros
3. Llamada a la rutina de servicio
4. Rutina de servicio llama a la función de utilidad
5. Reprograma y regresa al proceso de usuario

Ejemplo: El núcleo de Unix es el maestro que controla todos los demás programas, les asigna tiempo para acceder al hardware, y maneja el sistema de archivos y el modelo de seguridad. El núcleo se ejecuta como el proceso raíz con privilegios especiales. Todo lo demás corre en cuentas de usuario restringidas. Todo lo que esté fuera del kernel se denomina “espacio de usuario”.

Estructuras de modelos de sistemas operativos monolíticos:

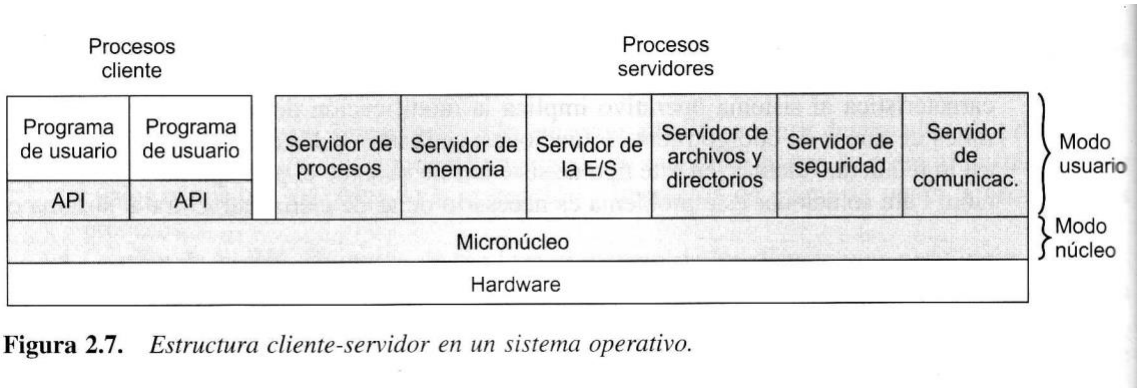


Figura 2.7. *Estructura cliente-servidor en un sistema operativo.*

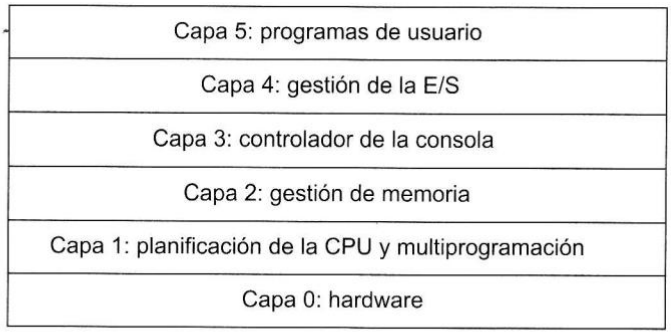


Figura 2.6. *Estructura por capas del sistema operativo THE.*

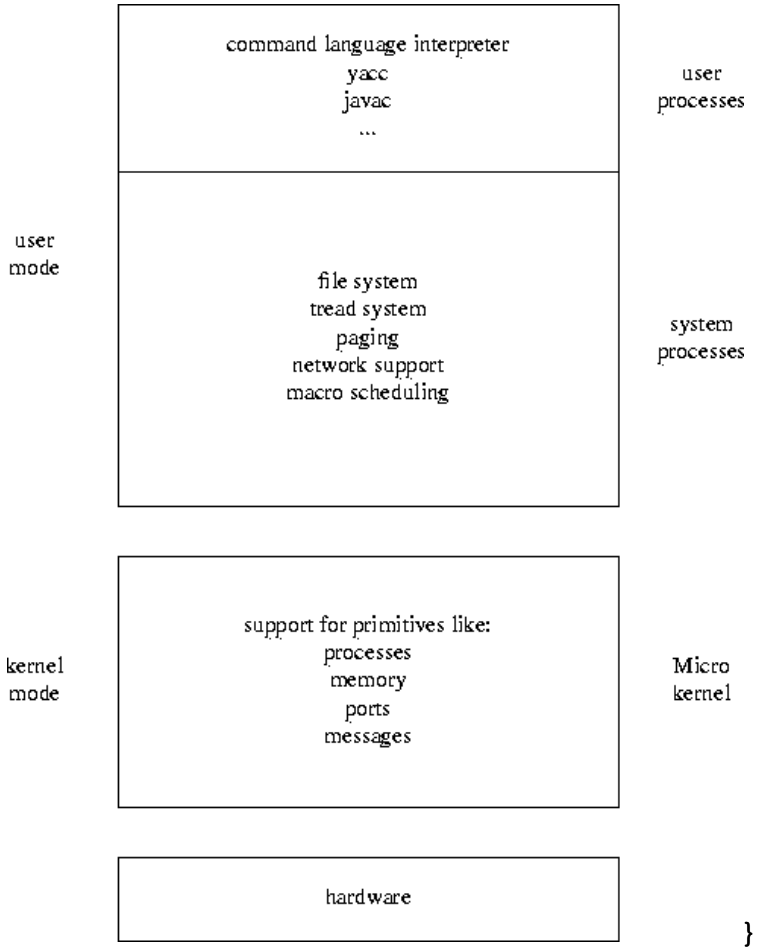
Nota: *Cada capa se comunica sólo con la capa inmediatamente superior y utiliza sólo los servicios de la capa inferior.*

Micronúcleo

La mayor parte del sistema operativo se ejecuta en procesos separados, fuera del núcleo. Se comunican por paso de mensajes. La tarea del núcleo es manejar el paso de mensajes, las interrupciones, administración de procesos de bajo nivel, y tal vez las entradas/salidas. Ejemplos de sistemas operativos con este diseño son: RC4000, Ameba, Chorus, Mach y Minix.

El micronúcleo asigna solamente algunas funciones esenciales al núcleo (espacio de direcciones, comunicación entre procesos, y planificación básica). Otros servicios son provistos por procesos llamados servidores que corren en modo usuario y son tratados como otras aplicaciones por el micronúcleo.

Arquitectura:

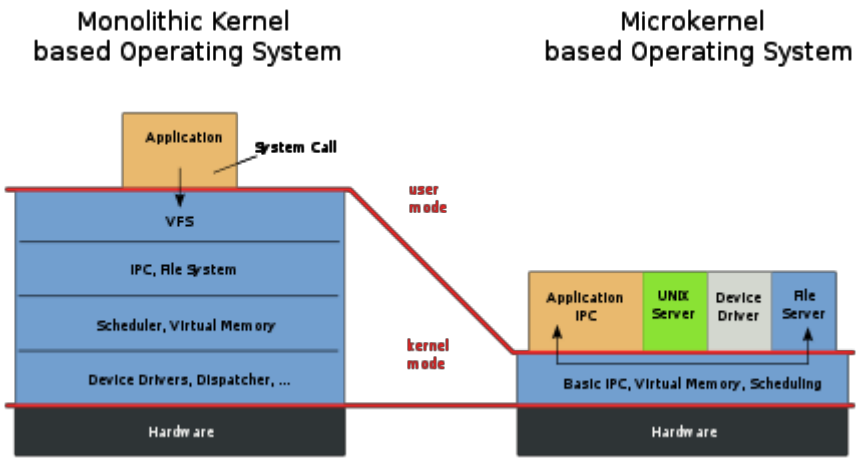


Fuente [12] Arquitectura Microkernel

Sistemas Monolitico vs Microkernel:

- Los sistemas operativos monolíticos son un solo archivo a.out que corre en modo kernel. Dicho archivo binario contiene todo: administración de procesos, de memoria, sistema de archivos, etc. Ejemplos de estos sistemas son UNIX, MS-DOS, VMS, MVS, OS/360, MULTICS y Linux.
- MINIX es un sistema basado en microkernel. EL sistema de archivo y la administración de memoria son procesos separados corriendo fuera del kernel. Los drivers de entrada y salida son también procesos separados.

Arquitecturas:



Fuente: Wikipedia, Microkernel [8]

Híbridos:

Es una arquitectura de sistemas operativos que combina aspectos de monolítico y micronúcleo (también se incluyen los casos de nanokernels y exokernels). El caso más popular de arquitectura híbrida es Windows NT dado que los subsistemas de emulación corren en procesos servidores en modo usuario en lugar de modo kernel como en la arquitectura monolítica. Adicionalmente, la mayor parte de los componentes del sistema corren en el mismo espacio de dirección de memoria que el kernel como en la arquitectura monolítica, mientras que en la arquitectura micronúcleo se usan espacios separados. [13]

Evolución de los sistemas operativos:

La evolución de los sistemas operativos se debe a actualizaciones de hardware, paginación, transferencia de páginas entre la memoria principal y secundaria, nuevos servicios, y las nuevas aplicaciones; correcciones, entre otras.

- **Procesamiento serial.** Los usuarios tienen acceso directo al CPU en serie. Problemas: planeamiento, tiempo de configuración (montar/desmontar cintas, iniciar todo el proceso cuando hay errores, etc.). Desperdicia tiempo del CPU. Es como si no hubiera OS.
-

- **Sistema de lotes simple** (Batch): usa un software residente en memoria (monitor) de manera que no haya acceso directo del usuario a la máquina.

    - Un operador recoge los trabajos secuencialmente en un dispositivo de entrada, para el uso del monitor.
    - Muestra como el monitor controla la secuencia de eventos: Se leen los trabajos desde un dispositivo de entrada, los colocan en el área de programas de usuario, y pasa el control a dicho trabajo.
    - Cuando éste ha terminado, el control es regresado al monitor, y los resultados son enviados a un dispositivo de salida.

    Hay menos tiempo vicioso, se mejora el tiempo de configuración a través de un lenguaje de control de trabajos. Este lenguaje provee instrucciones al monitor para manejar errores, control, etc. Problemas: no tiene protección de memoria – ningún programa de usuario debe alterar el área de memoria que contiene el monitor; temporizador (timer) – cada trabajo debe tener un tiempo limitado; instrucciones privilegiadas – algunas instrucciones deben ser ejecutadas solamente por el monitor; interrupciones. Es monoprogramado.

    CPU: P0 executes then E/S (CPU idle, waiting) P0…E/S…End…P1 can start execution

- **Sistema de lotes multiprogramado –** (ej. Mainframe systems) se busca mantener tanto el procesador como los dispositivos de E/S ocupados la mayor parte del tiempo. Debido a que los dispositivos de E/S son lentos comparados con el procesador, este tipo de sistema permite que el procesador sea intercambiado a otro trabajo mientras el programa de usuario en ejecución espera por la respuesta de un dispositivo E/S. Provee manejo de memoria y planificación.

    El hardware soporta interrupciones E/S y DMA (Direct Access Memory), de manera que el CPU pueda enviar un comando de E/S para un trabajo, proceder con la ejecución de otro mientras el comando de E/S es enviado al controlador de dispositivos. Cuando la operación de E/S es completada, el procesador es interrumpido y el control se pasa al programa manejador de interrupciones en el sistema operativo. Ej. WinXP

P0ex…E/S……..P0ex… Done.
P1 P1ex….E/S… P1ex
P2 P2ex……E/S

* Recordar los términos multiprogramación y monoprogramación; monoprocesador y multiprocesador.

- **Sistemas de tiempo compartido –** el tiempo del procesador es compartido entre múltiples usuarios. Múltiples usuarios accesan simultáneamente el sistema a través de terminales, con el sistema operativo intercalando la ejecución de cada programa de usuario en una pequeña computación. Los usuarios pueden estar desarrollando programas, ejecutando y usando distintas aplicaciones. (Pag. 66 Stallings). Ej. Windows 2003 Server, Red Hat.

- **Sistemas de transacciones de tiempo real –** (ej. Algunas líneas aéreas) parecido al concepto anterior, pero aquí es sólo una aplicación que puede ser consultada y actualizada por los usuarios, lo que puede ocasionar acceso al mismo registro simultáneamente.

Características de los sistemas operativos modernos:

Stallings [1] enumera ciertas características comunes en los sistemas operativos modernos.

→ Arquitectura **micronúcleo**.

→ **Multihilo.** Técnica que divide los procesos ejecutando una aplicación en hilos que pueden correr concurrentemente. Los procesos son colecciones de hilos y recursos del sistema asociados. Incrementa la velocidad de procesamiento en un monoprocesador.

→ **Multiprocesamiento simétrico (SMP).** Hay múltiples procesadores compartiendo el mismo espacio de memoria y E/S, interconectados por un bus común; todos los procesadores pueden realizar las mismas funciones. Es un sistema de un solo usuario con varios procesadores. SMP supera a los monoprocesadores en desempeño – ejecución en paralelo; hay mayor disponibilidad – si un procesador falla, el otro toma su lugar; crecimiento incremental – se pueden añadir más procesadores; escalabilidad – variedad de precios y características de acuerdo

con el # de procesadores. Se puede aplicar en máquinas de un solo hilo (un proceso = un hilo y varios procesos corren concurrentemente en distintos procesadores) o de varios hilos.

→ **Sistema distribuido.** Es un sistema en clusters de computadoras separadas cada una con su propia memoria principal, secundaria, y otros módulos de E/S. Provee la ilusión de un único espacio de memoria y una única memoria secundaria, y acceso a otras facilidades. Provee mecanismos unificados de acceso como por ejemplo un sistema de archivos distribuido. SMP y un sistema distribuido son características diferentes, la primera es monousuario y la segunda multiusuario. Ej. Sistema de archivos distribuidos, BDD.

→ **Diseño orientado a objetos**. Permite a los programadores personalizar un OS sin alterar la integridad del sistema cuando se agregan módulos.

Situaciones de Aprendizaje:

- Hacer un cuadro sinóptico de las características de Windows 2000 y de Unix. Qué incluye cada nivel?
- Leer y discutir a nivel técnico las estructuras Monolíticas, de micronúcleo e híbridas:

  http://en.wikipedia.org/wiki/Monolithic_kernel

  http://en.wikipedia.org/wiki/Microkernel

  http://linuxfinances.info/info/microkernel.html

  http://www.linuxjournal.com/article/6105

- Discutir el cuestionario:

1. Qué es un OS?
2. Cuál es el propósito del OS?
3. Qué hace el OS?
4. Describa los distintos tipos de OS.
5. Describa la secuencia de arranque normal de un computador
6. Qué es un proceso?
7. Qué son interrupciones?
8. Cómo se pueden clasificar las interrupciones?
9. Cuáles son los pasos del multiprocesamiento?
10. Qué significa thrashing?
11. Qué es un thread (hilo)?
12. Cuál es la diferencia entre OS Simétrico y Asimétrico?
13. Cuál es la función del administrador de memoria?
14. Qué es GUI?
15. Cuál es la diferencia entre API y interfaz de usuario?

Referencias

1. Stallings, William. Sistemas Operativos. V Edición. Prentice Hall. 2000.
2. Carretero, Sistemas Operativos: Una visión aplicada. Mc-Graw Hill. Segunda edición.
3. QNX, Wikipedia. http://en.wikipedia.org/wiki/QNX
4. RTLinux, Wikipedia: http://en.wikipedia.org/wiki/RTLinux
5. What is RTLinux: http://www.rtlinuxfree.com/
6. Windows Embedded CE 6.0 Documentation: http://msdn.microsoft.com/en-us/library/bb159115.aspx
7. Open Sources: Voices from the Open Source Revolution: http://oreilly.com/catalog/opensources/book/appa.html
8. Wikipedia, Microkernel: http://en.wikipedia.org/wiki/Microkernel
9. How operating system work: http://computer.howstuffworks.com/operating-system.htm/printable
10. Wikipedia, Shell (computing): http://en.wikipedia.org/wiki/Shell_(computing)
11. Wikipedia, Application Programming Interface: http://en.wikipedia.org/wiki/Application_programming_interface
12. Monolithic Architecture: http://www.cs.rit.edu/~hpb/Lectures/SIA/OS1/all-inOne-3.html
13. Hybrid Kernel: http://en.wikipedia.org/wiki/Hybrid_kernel

Tema: Gestión del Sistema de Archivos

Objetivo:

- Analizar el funcionamiento y manejo del sistema de archivos y de los dispositivos de entrada/salida en un Sistema Operativo.

Concepto

Un sistema de gestión de archivos es aquel conjunto de software de sistema que proporciona servicios a los usuarios y aplicaciones en el uso de archivos. Típicamente, la única forma en la que un usuario o aplicación puede acceder a los archivos es a través del sistema de gestión de archivos. Esto elimina la necesidad de que el usuario o programador desarrolle software de propósito especial para cada aplicación. El sistema de archivos incluye:

- La colección de todos los archivos en el disco.
- Conjuntos de estructuras de datos utilizadas para administrar archivos (directorios de archivos, descriptores de archivos, tablas de asignación de espacio libre y ocupado en disco).
- Un complejo de software de sistema que implementa varias operaciones en archivos: creación, destrucción, lectura, escritura, nomenclatura, búsqueda.

Algunas operaciones del sistema de archivo incluyen:

- Crear
- Borrar
- Abrir
- Cerrar
- Leer
- Escribir

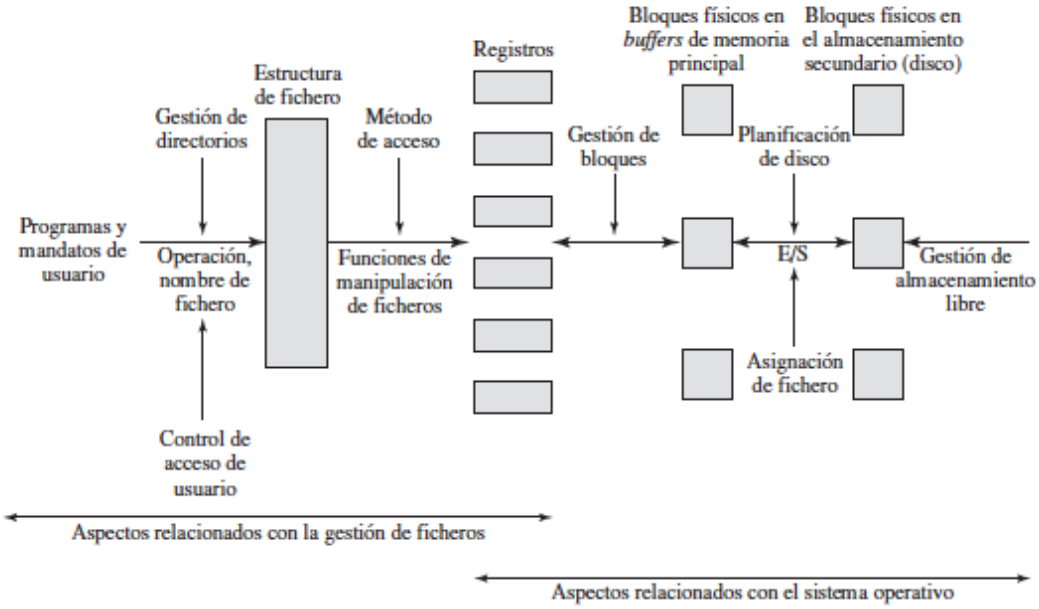


Figura 1. Organización y acceso a archivo

Estas tareas son complicadas en los sistemas operativos multitarea que están diseñados para funcionar como un solo usuario, pero proporcionan la capacidad de ejecutar múltiples procesos simultáneamente. Se agrega una nueva encomienda a las tareas enumeradas anteriormente: compartir un archivo de varios procesos. El archivo en este caso es un recurso compartido, lo que significa que el sistema de archivo (file system, FS) debe resolver todo el complejo de problemas asociados con dichos recursos. En particular: se deben proporcionar medios para bloquear el archivo y sus partes, hacer coincidir copias, evitar carreras, eliminar puntos muertos. En los sistemas multiusuario, surge otra tarea: proteger los archivos del acceso no autorizado de otro usuario.

Archivos

Nombre de archivo

Los usuarios necesitan poder referenciar un archivo mediante un nombre simbólico. Claramente, cada archivo en el sistema debe tener un nombre único a fin de que las referencias al mismo no sean ambiguas.

El uso de un directorio estructurado en forma de árbol minimiza la dificultad de asignar nombres únicos. Cualquier fichero del sistema se puede localizar siguiendo un camino desde el directorio raíz o maestro y bajando por las ramas hasta alcanzar el archivo. El conjunto de nombres de

directorios, finalizando en el nombre del archivo, constituye un nombre de camino para el archivo.

Estructura de un archivo

La estructura de un archivo involucra cuatro elementos, a saber: campo, registro, archivo, base de datos.

Campo. Es un elemento básico que contiene un valor único. Se caracteriza por su longitud y el tipo de datos (por ejemplo, ASCII, cadena de caracteres, decimal). Dependiendo del diseño del fichero, el campo puede tener una longitud fija o variable. En este último caso, el campo está formado normalmente por dos o tres subcampos: el valor real almacenado, el nombre del campo, y en algunos casos, la longitud del campo. En otros casos de campos de longitud variable, la longitud del campo se indica mediante el uso de símbolos de demarcación especiales entre campos.

Registro. Es una colección de campos relacionados que pueden tratarse como una unidad por alguna aplicación. Los registros también pueden ser de longitud fija o variable.

Archivo. Es una colección de campos relacionados. Se trata como una entidad única por los usuarios y por las aplicaciones.

Base de datos. Es una colección de datos relacionados. Los aspectos esenciales de una base de datos son que la relación que exista entre los elementos de datos sea explícita y que la base de datos se diseña para su uso por parte de varias aplicaciones diferentes. Una base de datos podría contener toda la información relacionada con una organización o proyecto, tal como información de negocio o de estudio científico. La base de datos está formada por uno o más tipos de archivos. Normalmente, hay un sistema de gestión de base de datos separado del sistema operativo, aunque hace uso de algunos programas de gestión de archivos.

Métodos de acceso

Los principales métodos de acceso a archivos son: secuencial, acceso directo, acceso indexado.

Acceso secuencial implica que sólo se leen los bytes en orden, empezando por el principio. No se puede saltar a una posición específica. Si hay que ir al inicio, se debe releer el archivo.

Acceso directo o aleatorio permite acceder a cualquier área o registro del archivo **a** partir de un puntero. Se asume que se puede mover aleatoriamente entre los distintos bloques que componen el archivo.

Acceso indexado. En el fichero indexado general, se abandonan los conceptos de secuencialidad y clave única. Los registros se acceden sólo a través de sus índices. El resultado es que no hay restricción en la colocación de los registros siempre que al menos un puntero en un índice se refiera a dicho registro. Además, se pueden emplear registros de longitud variable.

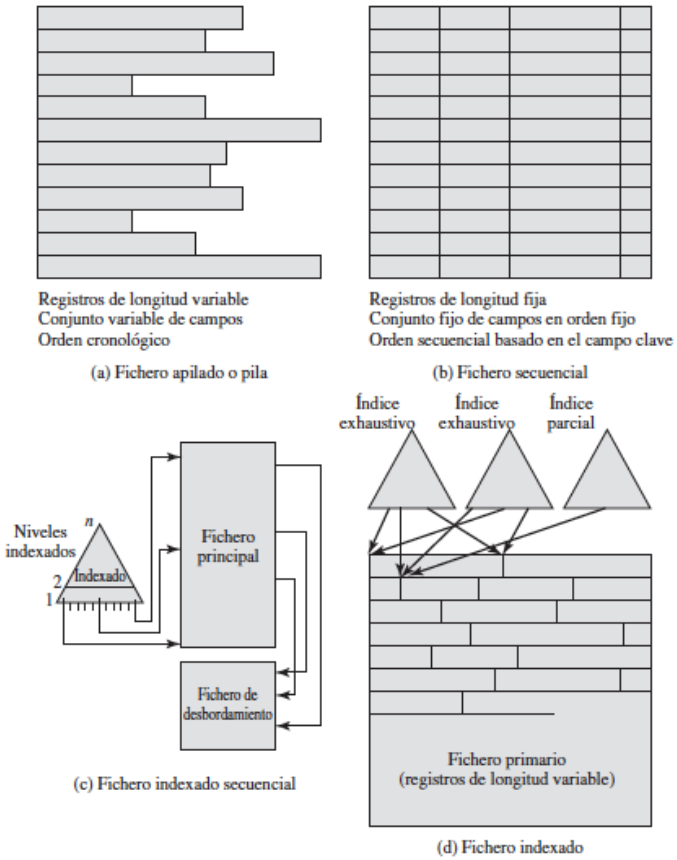


Figura 2. Organización y métodos de acceso

Directorios

Concepto

El directorio contiene información sobre los ficheros, incluyendo atributos, ubicación y propiedad. Gran parte de esta información, especialmente la que concierne a almacenamiento, la gestiona el sistema operativo. El directorio es a su vez un fichero, accesible por varias rutinas de gestión de ficheros. Aunque parte de la información de los directorios está disponible para los usuarios y las aplicaciones, esto se proporciona generalmente de forma indirecta por las rutinas del sistema.

Estructura

La forma más sencilla de estructura para un directorio es una lista de entradas, una por cada fichero. Esta estructura se podría representar como un fichero secuencial simple, con el nombre del fichero actuando como clave. En algunos sistemas iniciales monousuario, se ha utilizado esta técnica. Sin embargo, esta técnica es inadecuada cuando múltiples usuarios comparten el sistema o cuando un único usuario tiene muchos ficheros.

Una primera solución para resolver estos problemas sería pasar a un esquema de dos niveles. En este caso, hay un directorio por cada usuario y un directorio maestro. El directorio maestro tiene una entrada por cada directorio usuario, proporcionando información sobre dirección y control de acceso. Cada directorio de usuario es una lista simple de los ficheros de dicho usuario. Esto implica que los nombres deben ser únicos sólo dentro de la colección de los ficheros de un único usuario y que el sistema de ficheros puede fácilmente asegurar las restricciones de acceso de los directorios. Sin embargo, aún no proporciona ayuda a los usuarios para estructurar su colección de ficheros.

Una técnica más potente y flexible, que es casi universalmente adoptada, es utilizar una estructura jerárquica en forma de árbol. Como en la técnica anterior, hay un directorio maestro, que tiene bajo dicho directorio varios directorios de usuario. Cada uno de estos directorios de usuario, a su vez, podría tener subdirectorios y ficheros como entradas. Esto se cumple para todos los niveles: es decir, en cada nivel, un directorio podría estar formado por subdirectorios y/o ficheros.

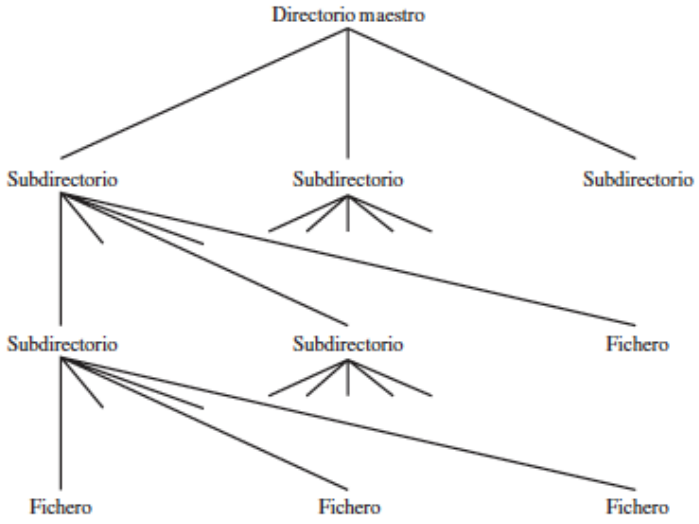


Figura 3. Estructura de directorio en forma de árbol

Nombres jerárquicos

El uso de un directorio estructurado en forma de árbol minimiza la dificultad de asignar nombres únicos. Cualquier archivo del sistema se puede localizar siguiendo un camino desde el directorio raíz o maestro y bajando por las ramas hasta alcanzar el archivo. El conjunto de nombres de directorios, finalizando en el nombre del archivo, constituye un nombre de camino para el archivo.

Se utiliza la barra (/) para delimitar nombres en una secuencia. El nombre del directorio maestro es implícito, porque todos los nombres comienzan en dicho directorio. Es aceptable tener varios archivos con el mismo nombre, siempre que ambos ficheros tengan nombres de camino únicos,

lo que equivale a decir que el mismo nombre de archivo se puede utilizar en diferentes directorios.

Aunque los nombres de camino facilitan la selección de los nombres de archivo, sería complicado para el usuario tener que escribir el camino completo cada vez que se hace una referencia a un archivo. Normalmente, un usuario interactivo o un proceso está asociado con un directorio actual, que se suele denominar directorio de trabajo.

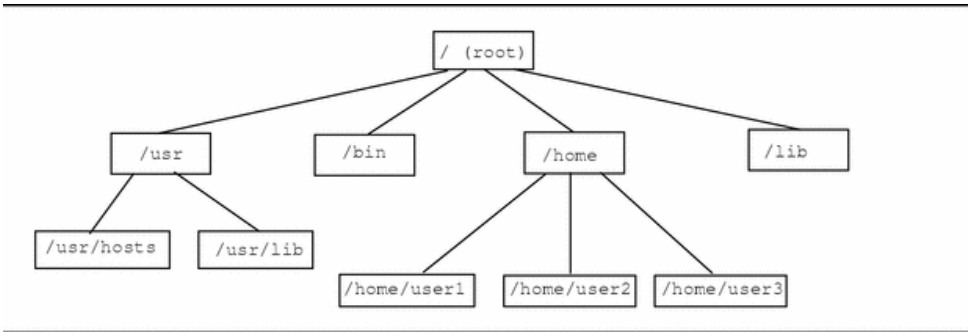


Figura 4. Jerarquía del sistema de archivos

Estructura y Almacenamiento del archivo y directorio

Los archivos por lo general se almacenan en discos. Existen dos estrategias para almacenar un archivo de n bytes. Esta misma situación se da en los sistemas de administración de memoria, caso segmentación y la paginación.

Asignación contigua: Al archivo se le asignan n bytes consecutivos de espacio de disco. Si el archivo crece (hay que tener en cuenta que los archivos son estructuras de datos con una alta volatilidad) probablemente tendrá que ser movido en el disco.

Asignación no contigua: El archivo se divide en m bloques de tamaño fijo que se almacenan en disco en bloques no necesariamente contiguos.

La segunda estrategia es más común. Una vez se ha decidido almacenar los archivos en bloques de tamaño fijo, se debe decidir el tamaño de éstos. El sector, la pista y el cilindro; y en el caso de un sistema de paginación, la página, son los referentes utilizados.

Después de decidir el tamaño del bloque, se debe considerar la forma de registrar los bloques libres. Una forma es llevar una lista ligada de bloques de disco en la cual cada uno contiene tanto bloques libres como sea posible. Otra técnica es llevar un mapa de bits. Los bloques libres se representan con el valor 1 en el mapa y los asignados con el valor 0. Algunos sistemas multiusuarios pueden establecer cuotas en el disco para evitar el uso excesivo de ciertos usuarios.

Sistemas de archivos

Un sistema de archivos son los métodos y estructuras de datos que un sistema operativo utiliza para seguir la pista de los archivos de un disco o partición; es decir, es la manera en la que se organizan los archivos en el disco. El término también es utilizado para referirse a una partición o disco que se está utilizando para almacenamiento, o el tipo del sistema de archivos que utiliza.

Algunos sistemas de archivos son:

- **Ext3.** El sistema de archivos ext3 posee todas las propiedades del sistema de archivos ext2. La diferencia es que se ha añadido una bitácora (journaling). Esto mejora el rendimiento y el tiempo de recuperación en el caso de una caída del sistema.
- **Vfat.** Esta es una extensión del sistema de archivos FAT conocida como FAT32. Soporta tamaños de discos mayores que FAT. La mayoría de discos con MS Windows son vfat.
- **Nfs.** Un sistema de archivos de red que permite compartir un sistema de archivos entre varios ordenadores para permitir fácil acceso a los archivos de todos ellos.

Referencia:

- Stallings, W. ***Sistemas Operativos***. 5a. Edición. Editorial Prentice-Hall. (TEXTO). Marzo 2014.

Infografía:

Sistema de archivos: https://sites.google.com/site/materiasisoperativo/unidad-5-sistemas-de-archivos/5-1-concepto
Trabajar con archivos y directorios: https://docs.oracle.com/cd/E19620-01/805-7644/6j76kloon/index.html
Estructura de sistemas de archivos: https://www.ibm.com/docs/es/aix/7.2?topic=tree-file-system-structure

Capítulo 3 – Hilos, SMP, Micronúcleo

Objetivos:

- Explicar el proceso de manejo del sistema operativo de la multiprogramación a través de la creación de hilos.
- Discutir el concepto de multiprocesamiento simétrico en los sistemas operativos.

Temas:

- Monohilo
- Multihilo
- SMP
- Micronúcleo

Monohilo

Un único hilo de ejecución por proceso.
Monohilo >1H=IP.

Casos:

- Un proceso con un solo hilo. Ej. MS-DOS.
- Múltiples procesos cada uno con uno solo hilo. Ej.: Unix antiguos.

Multihilo

El OS puede soportar múltiples hilos de ejecución dentro de un sólo proceso.

Casos:

- Un proceso con múltiples hilos: Java Virtual Machine.
- Múltiples procesos con múltiples hilos: Windows, Solaris, Mach, OS/2.

Ejemplos del uso de hilos:

1. Trabajos en 1er y 2do plano.

H1: muestra menús; H2: lee entrada de teclado; H3: ejecuta ordenes.

2. Procesamiento asíncrono: un hilo puede guardar 1 vez por minuto en un buffer en caso de pérdida de energía.
3. Velocidad de ejecución: cada hilo en distintos procesos.
4. Estructura modular

Un proceso en un ambiente multihilo consta de:

- espacio virtual para la imagen del proceso
- acceso protegido al procesador, otros procesos, archivos, y recursos de E/S.

Cada hilo de un proceso consta de:

- un estado de ejecución
- un contexto de hilo guardado cuando no está en ejecución
- una pila de ejecución
- un almacenamiento estático para variables locales
- acceso a memoria y recursos de sus procesos compartidos con otros hilos

Ventajas del uso de múltiples hilos dentro de los procesos:

- Menos tiempo de creación, finalización y cambio de hilo vs un proceso.
- No hay necesidad de cambio de modo del OS a núcleo para realizar comunicación entre hilos del mismo proceso. Ver. Fig. 4.2
- Mejoran el rendimiento tanto en equipo monoprocesador como en multiprocesador.

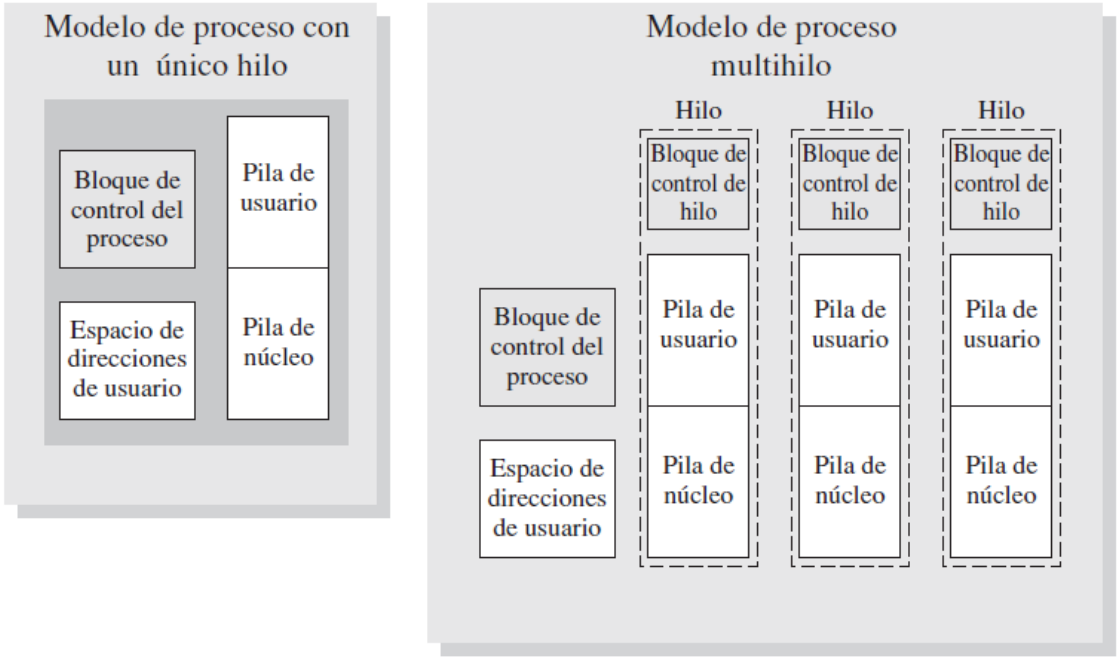


Figura 4.2. Modelos de proceso con un único hilo y multihilo.

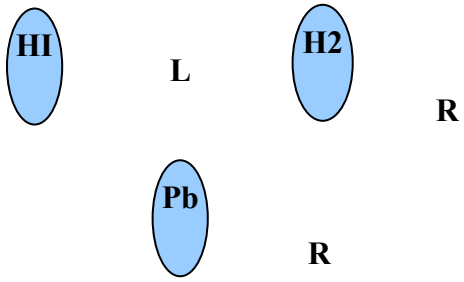
Los hilos en un monoprocesador se interconectan durante E/S o por tiempo de CPU.

Hilos a nivel de usuario – ULT	**Hilos a nivel de núcleo – KLT**
La aplicación administra los hilos; el núcleo no los reconoce, sino que lo planifica como una unidad. En la biblioteca se crea, destruye, planifica, se guarda y restaura y hay estructuras para el paso de mensajes y datos entre hilos.	El núcleo realiza toda la gestión. Caso Windows.
Ventajas:	Ventajas:

- menos carga al cambiar entre distintos hilos de un proceso porque no hay que pasar a modo núcleo; todo está en la biblioteca. - planificación puede ser específica de la aplicación dependiendo de las necesidades - pueden correr en cualquier OS. La librería de hilos es un conjunto de utilidades a nivel de aplicación compartidas por todas las aplicaciones.	- soluciona los problemas de hilos a nivel de usuario - el núcleo puede simultáneamente planificar múltiples hilos del mismo proceso en múltiples procesadores - si uno de los procesos está bloqueado, el núcleo puede planificar otro hilo del mismo proceso - las rutinas del núcleo pueden ser multihilos
Desventajas: - muchas llamadas al sistema son bloqueadoras, así que si un hilo hace una llamada al sistema bloqueará a los demás. (se resuelve con jacketing) - en una estrategia pura de hilos a nivel de usuario, sólo un hilo en un proceso puede estar en ejecución cada vez a pesar de contar con varios procesadores	Desventajas: - la transferencia de control de un hilo a otro dentro del mismo proceso requiere un cambio de modo a núcleo.

Jacketing: resuelve el bloqueo de hilos. La técnica convierte una llamada al sistema bloqueada a una no bloqueada, de manera que otros hilos puedan seguir en ejecución mientras un hilo hace una llamada al sistema.

1.

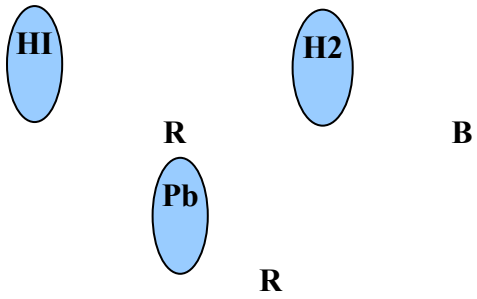


1. Se esta ejecutando el proceso B, hilo2.
2. Se esta ejecutando **H2,** hay una llamada al sistema que bloquea a **B;** el CPU pasa a otro proceso. Aunque B este Block o List, en la biblioteca H2 sigue R (aunque no este haciendo uso del CPU así lo ve la aplicación).

Obs.: Pb no esta en el CPU.

B

3. Interrupción de reloj para el proceso B, lo saca de ejecución y pasa a listo al proceso B.
4. H2 requiere de H1; H2= Block y H1=R.



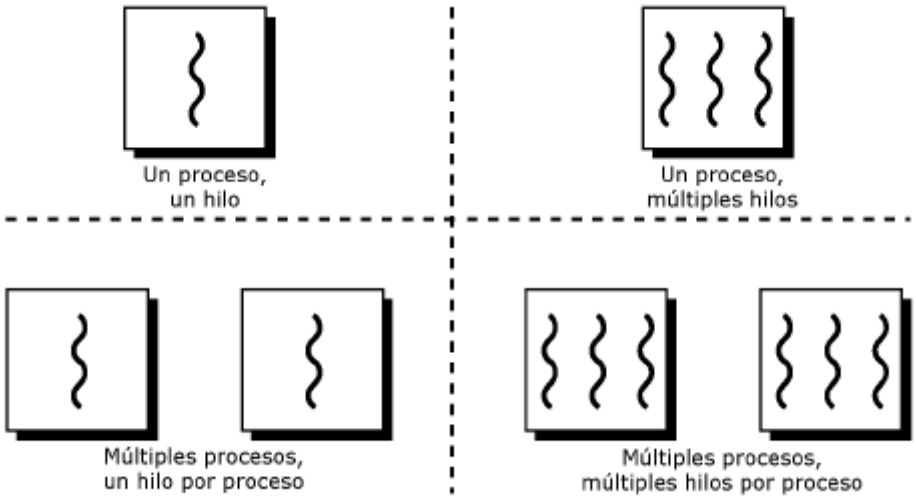
Obs.: Pb entra en ejecución así que sigue en H2 hasta que este se bloquea y se ejecuta.

3.1.2 Funcionalidad de los hilos.

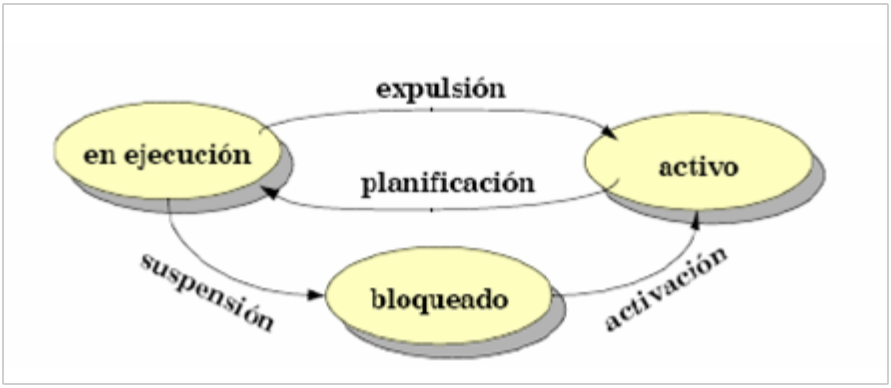
La funcionalidad de los hilos consiste en proporcionar un mecanismo eficiente para la comunicación y sincronización (comparten recursos) además evita en cierta medida los cambios de contextos.

Al igual que los procesos, los hilos poseen un estado de ejecución y pueden sincronizarse entre ellos para evitar problemas de compartimiento de recursos.

Generalmente, cada hilo tiene una tarea específica y determinada, como forma de aumentar la eficiencia del uso del procesador.



3.1.3 Estado de hilos



- **Ejecución, listo y bloqueado.**

Estados de un hilo en Unix

Un hilo puede ser en alguno de los siguientes estados:

- Listo: el hilo puede ser elegido para su ejecución.
- Standby: el hilo ha sido elegido para ser el siguiente en ejecutarse en el procesador.
- Ejecución: el hilo está siendo ejecutado.
- Espera: un hilo pasa a este estado cuando se bloquea por un suceso
- (E/S): se realiza una espera voluntaria de sincronización o alguien suspende al hilo.
- Transición: después de una espera el hilo pasa a este estado si está listo para ejecutar, pero alguno de sus recursos no está disponible aún.
- Terminado: un hilo llega a este estado cuando termina normalmente cuando su proceso padre ha terminado.

Un aspecto importante es si el bloqueo de un hilo implica el bloqueo del proceso completo. En otras palabras, si se bloquea un hilo de un proceso, ¿esto impide la ejecución de otro hilo del mismo proceso incluso si el otro hilo está en estado de Listo? Sin lugar a dudas, se pierde algo de la potencia y flexibilidad de los hilos si el hilo bloqueado bloquea al proceso entero.

Estados de un hilo en Linux

Linux no considera los hilos como tales. En Linux se crea un nuevo proceso copiando los atributos del proceso actual. Un nuevo proceso puede ser clonado para que comparta los recursos del actual, tales como archivos, gestores de la memoria virtual.

Cuando dos procesos comparten la memoria, operan en efecto como hilos dentro del mismo espacio, del mismo proceso.

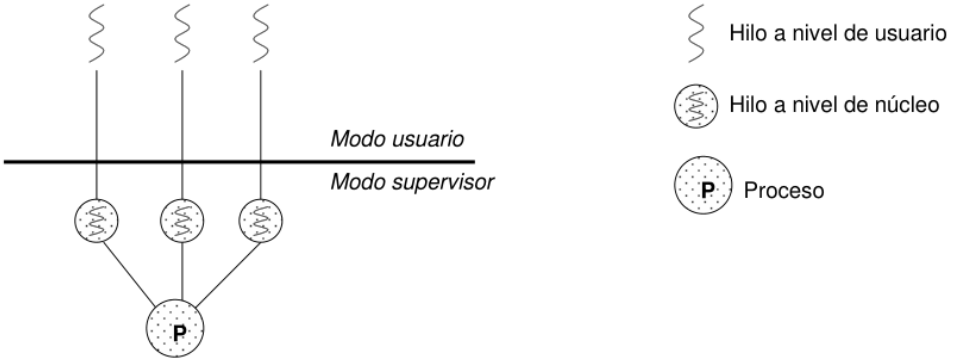
Sin embargo, no se manejan estructuras de datos para los hilos diferentes de las de los procesos, por lo que se puede argumentar que Linux no hace diferencias entre hilos y procesos.

3.1.4 Paralelismo.

Se refiere a la ejecución simultánea de varios procesos computacionales. Esto significa que se requieren varios medios de ejecución física: varios procesadores (o un procesador con varios núcleos) o varias computadoras (sistemas distribuidos) y la suficiente memoria para mantenerlos. Los procesos pueden estar relacionados entre ellos, para realizar una misma tarea, o no (Patricio, s.f.).

El paralelismo está relacionado con la capacidad del sistema en el que se ejecuta el programa, con sus recursos disponibles y que el software lo pueda aprovechar (Patricio, s.f.).

3.1.5 Hilos a nivel de usuario y de kernel



Hilos a nivel de usuario: son implementados en alguna librería. Estos hilos se gestionan sin soporte del SO, el cual solo reconoce un hilo de ejecución.

- En un entorno ULT puro, la aplicación gestiona todo el trabajo de los hilos y el núcleo no es consciente de la existencia de los mismos. La Figura 4.6 muestra el enfoque ULT. Cualquier aplicación puede programarse para ser multihilo a través del uso de una biblioteca de hilos, que es un paquete de rutinas para la gestión de ULT. La biblioteca de hilos contiene código para la creación y destrucción de hilos, para paso de mensajes y datos entre los hilos, para planificar la ejecución de los hilos, y para guardar y restaurar el contexto de los hilos.

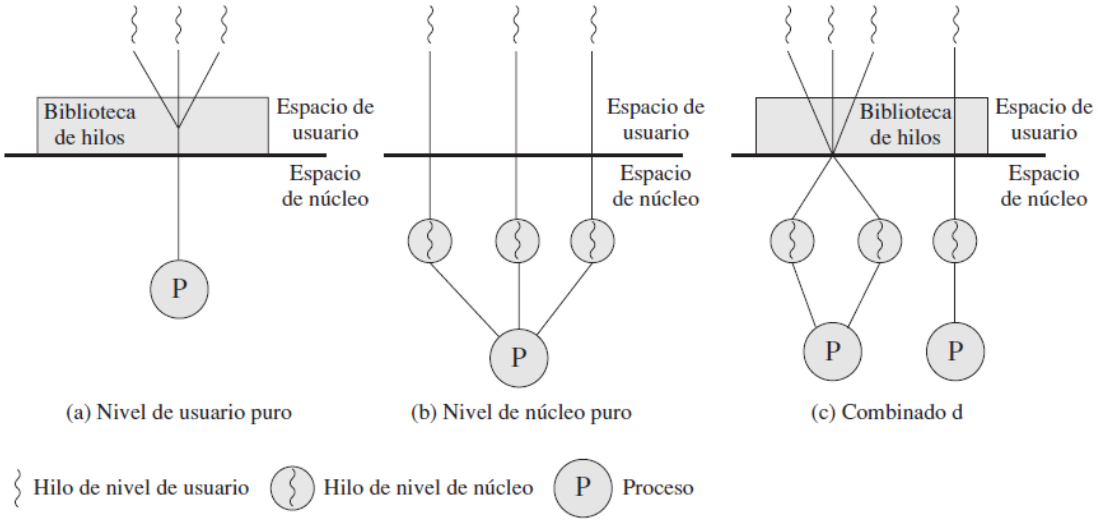


Figura 4.6. Hilos de nivel de usuario y de nivel de núcleo.

Hilos a nivel de kernel: el SO es quien crea, planifica y gestiona los hilos. Se reconocen tantos hilos como se hayan creado. Los hilos a nivel de usuario tienen como beneficio que su cambio de contexto es más sencillo que el cambio de contexto entre hilos de kernel. A demás, se pueden implementar aún si el SO no utiliza hilos a nivel de kernel. Otro de los beneficios consiste en poder planificar diferente a la estrategia del SO. Los hilos a nivel de kernel tienen como gran beneficio poder aprovechar mejor las arquitecturas multiprocesadores, y que proporcionan un mejor tiempo de respuesta, ya que, si un hilo se bloquea, los otros pueden seguir ejecutando.

- En un entorno KLT puro, el núcleo gestiona todo el trabajo de gestión de hilos. No hay código de gestión de hilos en la aplicación, solamente una interfaz de programación de aplicación (API) para acceder a las utilidades de hilos del núcleo. Windows es un ejemplo de este enfoque.

El uso de ULT en lugar de KLT, presenta las siguientes ventajas:

1. El cambio de hilo no requiere privilegios de modo núcleo porque todas las estructuras de datos de gestión de hilos están en el espacio de direcciones de usuario de un solo proceso. Por consiguiente, el proceso no cambia a modo núcleo para realizar la gestión de hilos. Esto ahorra la sobrecarga de dos cambios de modo (usuario a núcleo; núcleo a usuario).
2. La planificación puede especificarse por parte de la aplicación. Una aplicación se puede beneficiar de un simple algoritmo de planificación cíclico, mientras que otra se podría beneficiar de un algoritmo de planificación basado en prioridades. El algoritmo de planificación se puede hacer a medida sin tocar el planificador del sistema operativo.
3. Los ULT pueden ejecutar en cualquier sistema operativo. No se necesita ningún cambio en el nuevo núcleo para dar soporte a los ULT. La

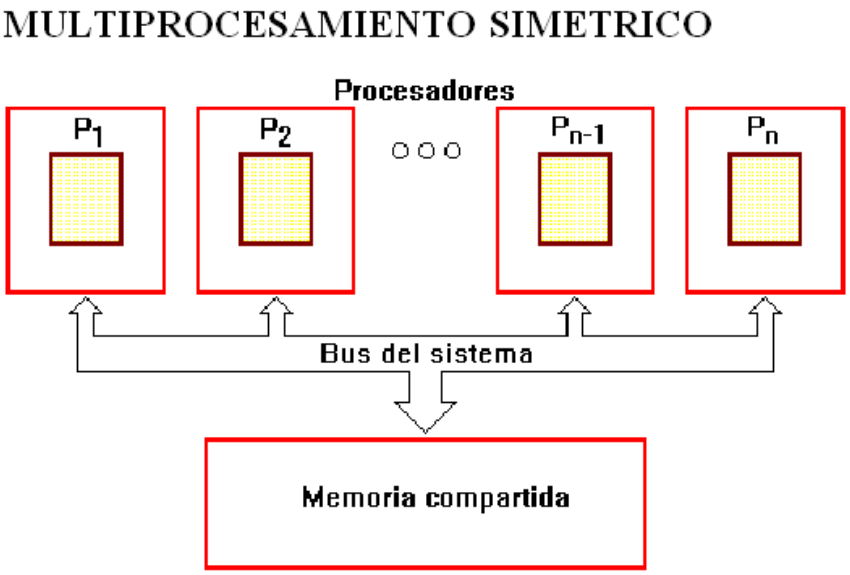
biblioteca de los hilos es un conjunto de utilidades a nivel de aplicación que comparten todas las aplicaciones.

Hay dos desventajas de los ULT en comparación con los KLT:

1. En un sistema operativo típico muchas llamadas al sistema son bloqueantes. Como resultado, cuando un ULT realiza una llamada al sistema, no sólo se bloquea ese hilo, sino que se bloquean todos los hilos del proceso.
2. En una estrategia pura ULT, una aplicación multihilo no puede sacar ventaja del multiproceso. El núcleo asigna el proceso a un solo procesador al mismo tiempo. Por consiguiente, en un determinado momento sólo puede ejecutar un hilo del proceso. En efecto, tenemos multiprogramación a nivel de aplicación con un solo proceso. Aunque esta multiprogramación puede dar lugar a una mejora significativa de la velocidad de la aplicación, hay aplicaciones que se podrían beneficiar de la habilidad de ejecutar porciones de código de forma concurrente.

Multiproceso simétrico

El multiprocesamiento simétrico se entiende como los sistemas que incluyen más de un procesador. Son capaces de ejecutar diversos procesos de forma simultánea y, además, comparten una misma memoria para el cumplimiento de sus funciones (Team, 2023).



Características del multiprocesamiento simétrico

- Dentro de las características y propiedades destacables del multiprocesamiento simétrico, se encuentra su utilidad para la adición de procesadores, memoria y otros recursos y componentes que permiten el incremento del rendimiento en los sistemas.

- De la misma forma, este tipo de multiprocesamiento se caracteriza porque cada procesador lleva a cabo la ejecución de las tareas en el sistema operativo. Así, se toman los procesos de una cola preparada en común o privada para cada uno de los procesadores.

- Como propiedad del multiprocesamiento simétrico aparece también que todo su procesador mantiene la misma arquitectura y que sus procesadores incluyen la capacidad de comunicarse con otros a través del recurso de memoria compartida.

Se refiere a la arquitectura hardware del sistema multiprocesador y al comportamiento del sistema operativo que utiliza dicha arquitectura. Un SMP es un computador con las siguientes características:
1) Tiene dos o más procesadores similares de capacidades comparables.
2) Los procesadores comparten la memoria principal y la E/S, y están interconectados mediante un bus u otro tipo de sistema de interconexión, de manera que el tiempo de acceso a memoria es aproximadamente el mismo para todos los procesadores.
3) Todos los procesadores comparten los dispositivos de E/S, pero pueden hacerlo bien a través de los mismos canales, o bien a través de otros caminos de acceso al mismo dispositivo.
4) Todos los procesadores pueden desempeñar las mismas funciones (de ahí el término simétrico).
5) El sistema está controlado por un sistema operativo que posibilita la interacción entre los procesadores y sus programas.

Objetivo de la especificación MP

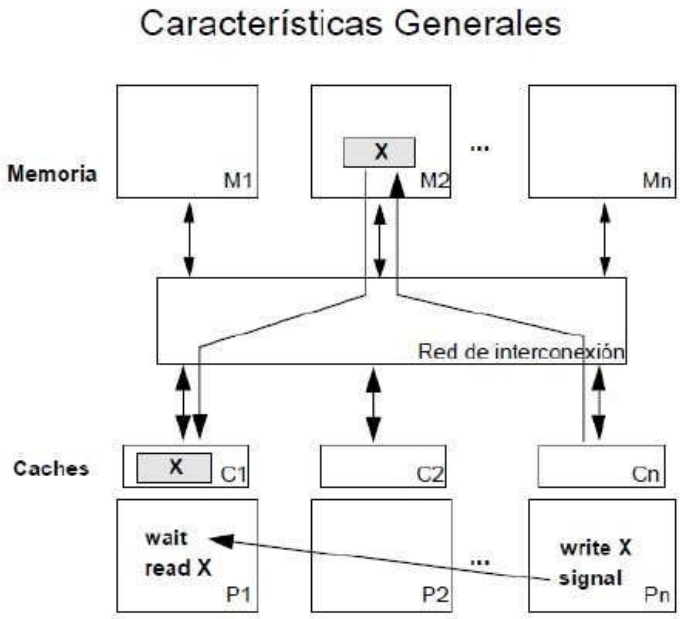
La meta más importante de esta norma es realizar una ampliación de la plataforma PC/AT mediante la cual sea posible diseñar equipos multiprocesador manteniendo una compatibilidad total con el hardware y el software existente (Galàn, 1996).

Mediante la consecución de este objetivo anteriormente mencionado, es posible continuar ejecutando todas las aplicaciones existentes para la plataforma PC/AT y, a través de la utilización de sistemas operativos y aplicaciones para multiproceso, conseguir importantes incrementos en la productividad de los equipos.

Para construir un sistema compatible con esta especificación es necesario, sin embargo, realizar modificaciones en elementos clave del ordenador, como la BIOS del equipo, que debe construir una tabla de datos que presente el hardware de una forma clara a los controladores de dispositivo o a la capa de abstracción de hardware del sistema operativo (Galàn, 1996).

3.2.1 Arquitectura SMP

La arquitectura SMP (Multi-procesamiento simétrico, también llamada UMA, de Uniform Memory Access), se caracteriza por el hecho de que varios microprocesadores comparten el acceso a la memoria. Todos los microprocesadores compiten en igualdad de condiciones por dicho acceso, de ahí la denominación "simétrico". Los sistemas SMP permiten que cualquier procesador trabaje en cualquier tarea sin importar su localización en memoria; con un propicio soporte del sistema operativo, estos sistemas pueden mover fácilmente tareas entre los procesadores para garantizar eficientemente el trabajo.



Una computadora SMP se compone de microprocesadores independientes que se comunican con la memoria a través de un bus compartido. Dicho bus es un recurso de uso común. Por tanto, debe ser arbitrado para que solamente un microprocesador lo use en cada instante de tiempo. Si las computadoras con un solo microprocesador tienden a gastar considerable tiempo esperando a que lleguen los datos desde la memoria, SMP empeora esta situación, ya que hay varios parados en espera de datos (Gustavo, s.f.).

La forma más común de categorizar estos sistemas es la taxonomía de sistemas de procesamiento paralelo introducida por Flynn [FLYN72]. Flynn propone las siguientes categorías de sistemas de computadores:

- **Única instrucción, único flujo de datos – Single instruction single data (SISD) stream.** Un solo procesador ejecuta una única instrucción que opera sobre datos almacenados en una sola memoria.

- **Única instrucción, múltiples flujos de datos – Single instruction multiple data (SIMD) stream.** Una única instrucción de máquina controla la ejecución simultánea de un número de elementos de proceso. Cada elemento de proceso tiene una memoria de datos asociada, de forma que cada instrucción se ejecuta en un conjunto de datos diferente a través de los diferentes procesadores. Los procesadores vectoriales y matriciales entran dentro de esta categoría.
- **Múltiples instrucciones, único flujo de datos – Multiple instruction single data (MISD) stream.** Se transmite una secuencia de datos a un conjunto de procesadores, cada uno de los cuales ejecuta una secuencia de instrucciones diferente. Esta estructura nunca se ha implementado.
- **Múltiples instrucciones, múltiples flujos de datos – Multiple instruction multiple data (MIMD) stream.** Un conjunto de procesadores ejecuta simultáneamente diferentes secuencias de instrucciones en diferentes conjuntos de datos.

En un **multiprocesador simétrico (Symmetric Multiprocessor, SMP)**, el núcleo puede ejecutar en cualquier procesador, y normalmente cada procesador realiza su propia planificación del conjunto disponible de procesos e hilos. El núcleo puede construirse como múltiples procesos o múltiples hilos, permitiéndose la ejecución de partes del núcleo en paralelo. El enfoque SMP complica al sistema operativo, ya que debe asegurar que dos procesadores no seleccionan un mismo proceso y que no se pierde ningún proceso de la cola. Se deben emplear técnicas para resolver y sincronizar el uso de los recursos.

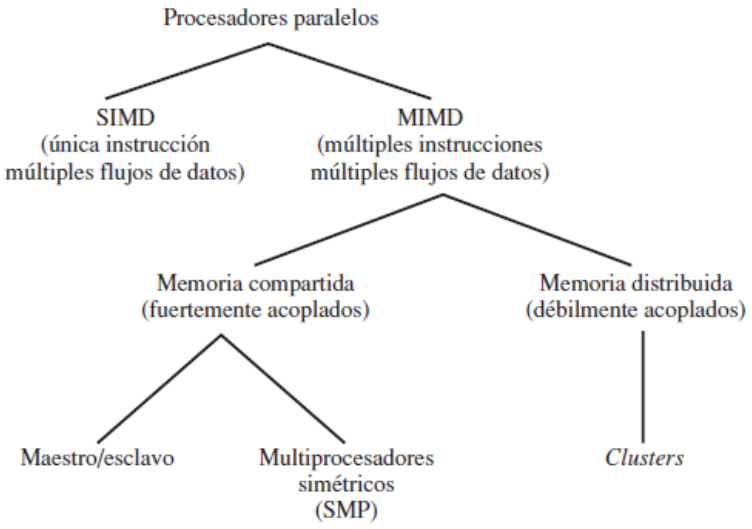


Figura 4.8. Arquitectura de procesadores paralelos.

El diseño de SMP y clusters es complejo, e involucra temas relativos a la organización física, estructuras de interconexión, comunicación entre procesadores, diseño del sistema operativo y técnicas de aplicaciones software.

3.2.2 Organización SMP

La Figura 4.9 muestra la organización general de un SMP. Existen múltiples procesadores, cada uno de los cuales contiene su propia unidad de control, unidad aritmético-lógica y registros. Cada procesador tiene acceso a una memoria principal compartida y dispositivos de E/S a través de algún mecanismo de interconexión; el bus compartido es común a todos los procesadores.

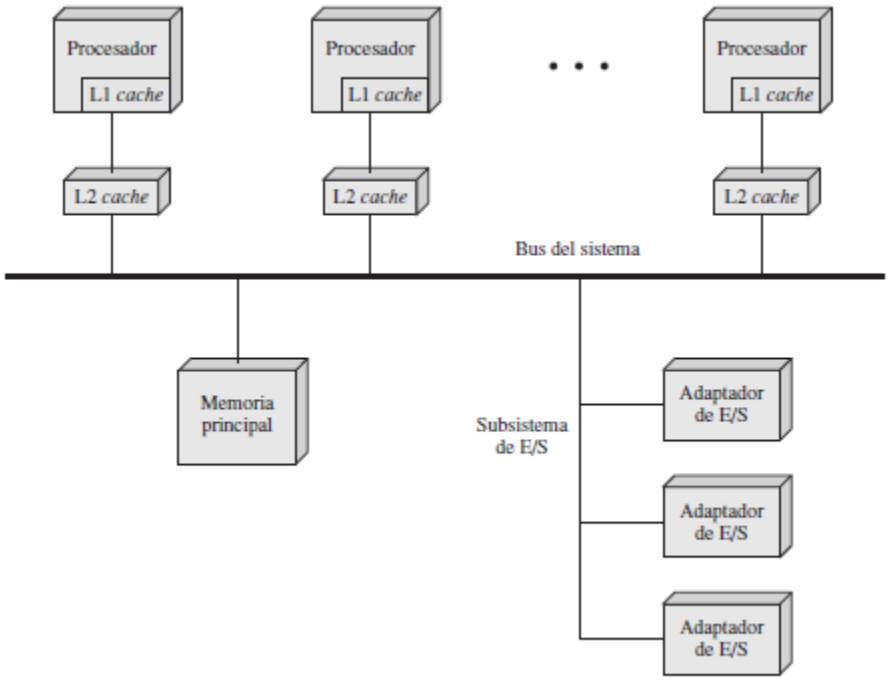
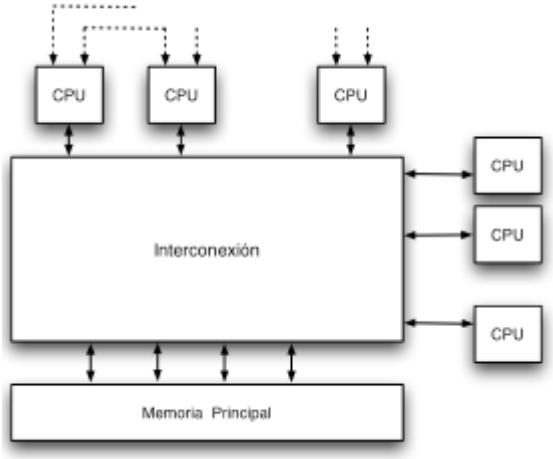


Figura 4.9. Organización de los multiprocesadores simétricos.

Los procesadores se pueden comunicar entre sí a través de la memoria (mensajes e información de estado dejados en espacios de memoria compartidos). Los procesadores han de poder intercambiarse señales directamente. A menudo la memoria está organizada de tal manera que se pueden realizar múltiples accesos simultáneos a bloques separados.



Existen dos o más CPUs, cada una de las cuales contiene:
a) unidad de control
b) ALU
c) registros
d) posiblemente caché.
Cada CPU tiene acceso a una memoria principal compartida y a los dispositivos I/O a través de un mecanismo de interconexión. Los procesadores pueden comunicarse entre ellos a través de la memoria (mensajes e información de estados almacenados en áreas comunes).

La organización de un sistema multiprocesador puede clasificarse de la siguiente forma:
• Tiempo compartido o Bus Común.
• Memoria Multipuerto.
• Unidad de Control Central.

Bus de tiempo compartido:

- Direccionamiento: Se pueden distinguir los módulos del bus para determinar la fuente y el destino de los datos.
- Arbitraje: Existe un mecanismo para arbitrar peticiones de control del bus, utilizando algún tipo de esquema de prioridades. Los módulos I/O también pueden funcionar temporalmente como master.
- Tiempo Compartido: Cuando un módulo está controlando el bus, los módulos restantes no están autorizados y deben suspender, si es necesario, la operación hasta que se les asigne el acceso al bus.

Las principales ventajas de esta estructura son:

• Simplicidad, ya que la estructura es la misma que en un sistema uniprocesador.
• Flexibilidad: Es fácil expandir el sistema añadiendo más CPUs.
• Fiabilidad: El bus es esencialmente un medio pasivo, por lo que en principio no debe producir fallos en el sistema.

Memoria multipuerto:

La aproximación de memoria multipuerto permite el acceso independiente y directo a los módulos de memoria principal por parte de cada CPU y módulo I/O. Se necesita una lógica asociada a la memoria para resolver conflictos de acceso. El método más utilizado es asignar permanentemente prioridades designadas a cada puerto de memoria.

Unidad Central Control:
La unidad central de control proporciona canales de datos separados para cada sentido entre módulos independientes: CPU, memoria y I/O. El controlador memoriza las peticiones e implementa funciones de arbitraje y temporización. Puede pasar también mensajes de control y estado entre CPUs, y alertar de modificación de cachés.

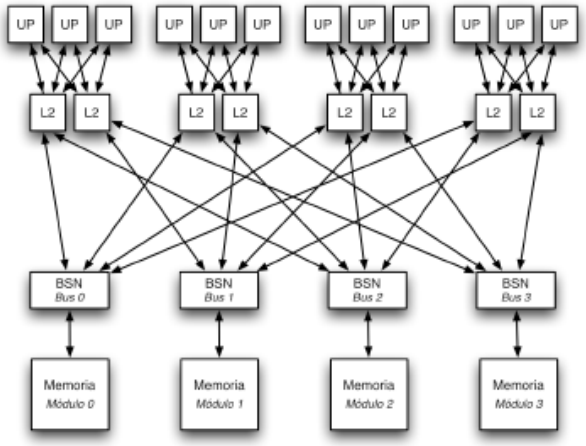
3.2.3 Consideraciones

Consideraciones de diseño de sistemas operativos multiprocesador

Las principales claves de diseño incluyen las siguientes características:

- **Proceso o hilo simultáneos concurrentes:** Las rutinas del núcleo necesitan ser reentrantes para permitir que varios procesadores ejecuten el mismo código del núcleo simultáneamente.
- **Planificación:** La planificación se puede realizar por cualquier procesador, por lo que se deben evitar los conflictos. Si se utiliza multihilo a nivel de núcleo, existe la posibilidad de planificar múltiples hilos del mismo proceso simultáneamente en múltiples procesadores.
- **Sincronización:** La sincronización es un servicio que fuerza la exclusión mutua y el orden de los eventos.
- **Gestión de memoria:** Además, el sistema operativo necesita explotar el paralelismo hardware existente, como las memorias multipuerto, para lograr el mejor rendimiento. Los mecanismos de paginación de los diferentes procesadores deben estar coordinados para asegurar la consistencia cuando varios procesadores comparten una página o segmento y para decidir sobre el reemplazo de una página.
- **Fiabilidad y tolerancia a fallos.** El sistema operativo no se debe degradar en caso de fallo de un procesador. El planificador y otras partes del sistema operativo deben darse cuenta de la pérdida de un procesador y reestructurar las tablas de gestión apropiadamente.

Grandes Computadores SMP (IBM S/390)
La mayoría de los PC y estaciones de trabajo de tipo SMP utilizan una estrategia de interconexión basada en bus. Sin embargo, existen otro tipo de organizaciones donde el mecanismo de interconexión no es un bus. Por ello, resulta ilustrativo analizar una aproximación alternativa al bus común, que se utiliza en las implementaciones más recientes de la familia de grandes computadores (mainframes) IBM S/390.



Muestra un esquema que corresponde a la organización general del SMP S/390. Esta familia de sistemas es escalable, e incluye, desde computadores monoprocesador con un módulo de memoria principal, en su configuración más básica, hasta sistemas con diez procesadores y cuatro módulos de memoria, en la gama alta.

Organización micronúcleo

Implementan en su núcleo únicamente la planificación de procesos, la gestión de interrupciones (la parte básica fundamental de la gestión de E/S que necesariamente se tiene que realizar en modo privilegiado) y la comunicación entre procesos. Por tanto, la administración de memoria principal, la gestión de la E/S y la gestión de ficheros se realiza en modo usuario. En este tipo de sistema operativo hay procesos especiales propios del sistema operativo que implementan dichas funcionalidades en modo usuario y se denominan proceso servidor.

Se caracterizan por disponer de un núcleo que implementa únicamente:

- Planificación de procesos
- Mecanismo de comunicación entre procesos
- Gestión de interrupciones

Además, existen procesos servidores que se ejecutan en modo no privilegiado del procesador - que, por supuesto, se ejecutan fuera del espacio del núcleo del sistema operativo - y que implementan los siguientes componentes:

- Administración de memoria principal
- Administración de ficheros
- Gestión de dispositivos de entrada/salida.

Siguiendo este esquema, cuando un proceso cualquiera solicita un servicio a través de una llamada al sistema, el micronúcleo canaliza la petición al proceso servidor correspondiente. Dicha comunicación se realiza mediante mensajería.
Los procesos no tienen que distinguir entre nivel de núcleo y usuario porque los servicios son provistos por el paso de mensajes.

La desventaja es que toma más tiempo en construir y enviar un mensaje por el micronúcleo y aceptar y decodificar la respuesta que hacer un simple servicio de llamada.

La filosofía existente en el micronúcleo es que solamente las funciones absolutamente esenciales del sistema operativo estén en el núcleo. Los servicios y aplicaciones menos esenciales se construyen sobre el micronúcleo y se ejecutan en modo usuario. Aunque la filosofía de qué hay dentro y qué hay fuera del micronúcleo varía de un diseño a otro, la característica general es que muchos servicios que tradicionalmente habían formado parte del sistema operativo ahora son subsistemas externos que interactúan con el núcleo y entre ellos mismos; algunos ejemplos son: manejadores de dispositivos, servidores de archivos, gestores de memoria virtual, sistemas de ventana y servicios de seguridad.

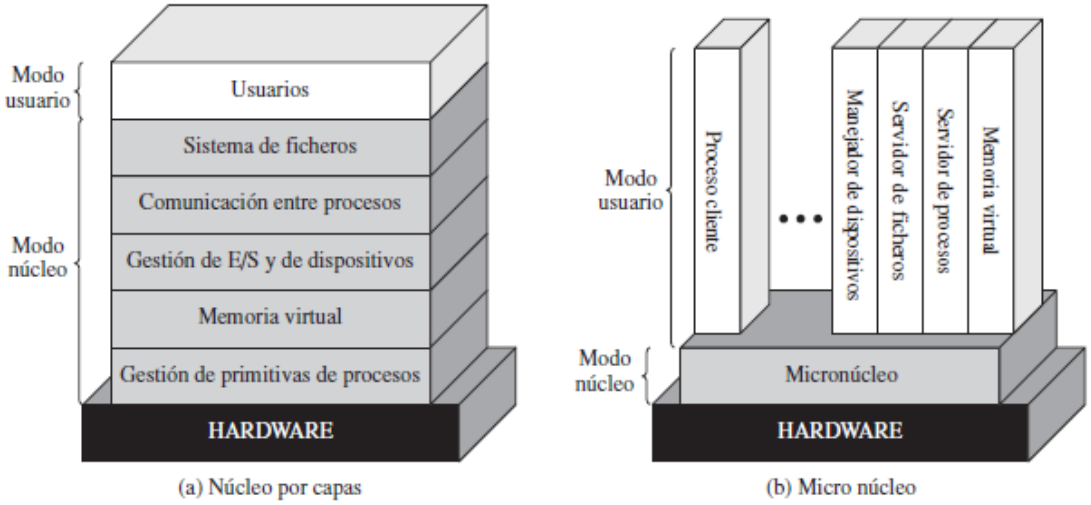


Figura 4.10. Arquitectura del núcleo.

La arquitectura del micronúcleo reemplaza la tradicional estructura vertical y estratificada en capas por una horizontal (Figura 4.10). Los componentes del sistema operativo externos al micronúcleo se implementan como servidores de procesos; interactúan entre ellos dos a dos, normalmente por paso de mensajes a través del micronúcleo. De esta forma, el micronúcleo funciona como un intercambiador de mensajes: válida mensajes, los pasa entre los componentes, y concede el acceso al hardware. El micronúcleo también realiza una función de protección; previene el paso de mensajes a no ser que el intercambio esté permitido.

Beneficios de una organización micronúcleo:

- Interfaces uniformes
- Extensibilidad
- Flexibilidad
- Portabilidad
- Fiabilidad

- Soporte de sistemas distribuidos
- Soporte de sistemas operativos orientados a objetos (OOOS)

Rendimiento del micronúcleo

Una potencial desventaja que se cita a menudo de los micronúcleos es la del rendimiento. Lleva más tiempo construir y enviar un mensaje a través del micronúcleo, y aceptar y decodificar la respuesta, que hacer una simple llamada a un servicio. Sin embargo, también son importantes otros factores, de forma que es difícil generalizar sobre la desventaja del rendimiento, si es que la hay.

Diseño del micronúcleo

El micronúcleo debe incluir aquellas funciones que dependen directamente del hardware y aquellas funciones necesarias para mantener a los servidores y aplicaciones operando en modo usuario. Estas funciones entran dentro de las categorías generales de gestión de memoria a bajo nivel, intercomunicación de procesos (IPC), y E/S y manejo de interrupciones.

- **Gestión de memoria a bajo nivel**. El micronúcleo tiene que controlar el concepto hardware de espacio de direcciones para hacer posible la implementación de protección a nivel de proceso. Con tal de que el micronúcleo se responsabilice de la asignación de cada página virtual a un marco físico, la parte principal de gestión de memoria, incluyendo la protección del espacio de memoria entre procesos, el algoritmo de reemplazo de páginas y otra lógica de paginación, pueden implementarse fuera del núcleo.

  [LIED95] recomienda un conjunto de tres operaciones de micronúcleo que pueden dar soporte a la paginación externa y a la gestión de memoria virtual:
  • **Conceder (Grant).** El propietario de un espacio de direcciones (un proceso) puede conceder alguna de sus páginas a otro proceso. El núcleo borra estas páginas del espacio de memoria del otorgante y se las asigna al proceso especificado.
  • **Proyectar (Map).** Un proceso puede proyectar cualquiera de sus páginas en el espacio de direcciones de otro proceso, de forma que ambos procesos tienen acceso a las páginas. Esto genera memoria compartida entre dos procesos. El núcleo mantiene la asignación de estas páginas al propietario inicial, pero proporciona una asociación que permite el acceso de otros procesos.
  • **Limpiar (Flush).** Un proceso puede reclamar cualquier página que fue concedida o asociada a otro proceso.

- **Gestión de E/S e interrupciones.** Con una arquitectura micronúcleo es posible manejar las interrupciones hardware como mensajes e incluir los puertos de E/S en los espacios de direcciones. El micronúcleo puede reconocer las interrupciones pero no las puede manejar. Más bien, genera un mensaje para el proceso a nivel de usuario que está actualmente asociado con esa interrupción. De esta forma, cuando se habilita una interrupción, se asigna un proceso de nivel de usuario a esa interrupción y el núcleo mantiene las asociaciones. La transformación de las interrupciones en mensajes las debe realizar el micronúcleo, pero el micronúcleo no está relacionado con el manejo de interrupciones específico de los dispositivos.

3.4 Hilos y Multiproceso Simétrico (SMP) en Linux.

Los hilos o threads son rutinas de código que corren de manera concurrente dentro de un mismo proceso.

Linux no considera los hilos como tales, se refiere a ellos como tareas más que como hilos. En Linux se crea un nuevo proceso copiando los atributos del proceso actual. Un nuevo proceso puede ser clonado para que comparta los recursos del actual, tales como archivos, gestores de señales o la memoria virtual.

Un proceso, o tarea, en Linux se representa por una estructura de datos task_struct, que contiene información de diversas categorías:

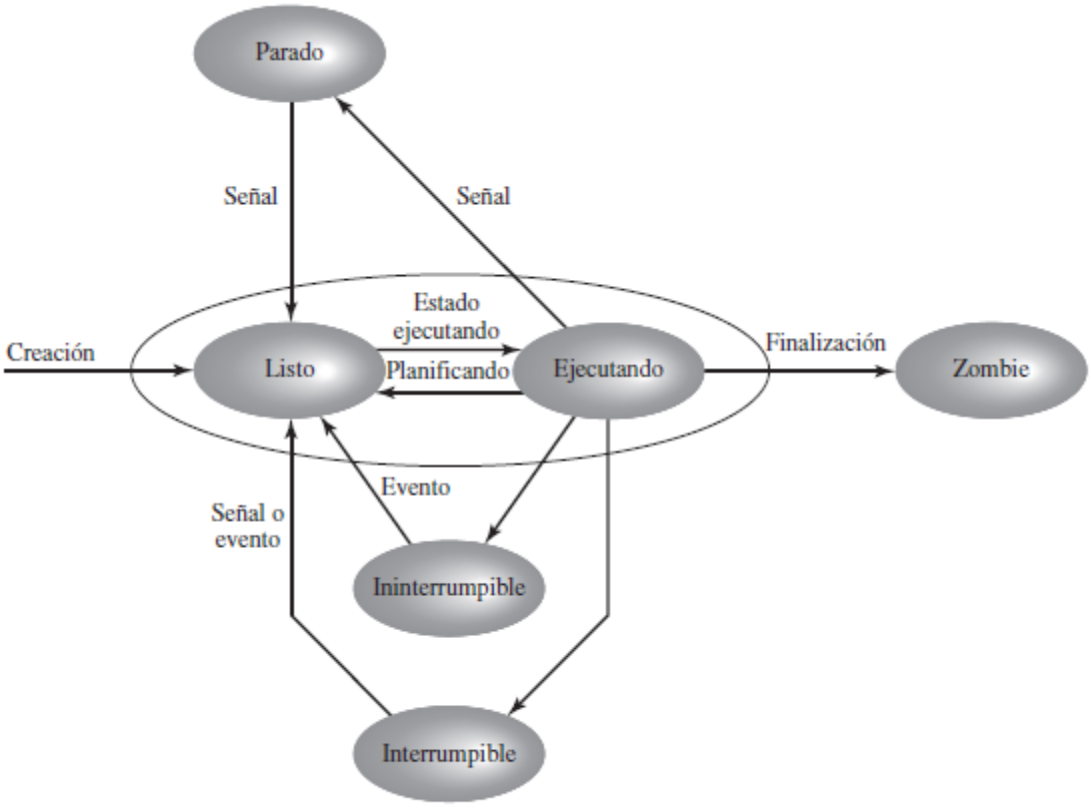


Figura 4.18. Modelo de procesos e hilos en Linux.

3.5 Hilos y SMP en UNIX.

Solaris hace uso de cuatros conceptos independientes relativos a hilos:

Proceso: este es el proceso UNIX convencional e incluye el espacio de direcciones de usuarios, la pila, y el bloque de control de procesos.

Hilos a nivel de usuario: implementados en el espacio de direcciones de un proceso por medio de una biblioteca de hilos, esos son invisibles para el sistema operativo. Los hilos a nivel de usuario (ULT) son la interfaz para el paralelismo de aplicaciones.

Procesos ligeros: un proceso ligero (LWP) puede verse como una correspondencia entre ULT e hilos del núcleo. Cada LWP soporta uno o más ULT y los hace corresponder con un hilo del núcleo. El núcleo planifica los LWP independientemente y puede ejecutar en paralelo sobre multiprocesadores.
Hilos del núcleo: son las entidades básicas de planificación y expedición en cada uno de los procesadores del sistema. Un LWP es visible para la aplicación dentro del proceso. De este modo, las estructuras de datos LWP existen dentro de los respectivos espacios de direcciones de los procesos. Al mismo tiempo, cada LWP este confinado a un único hilo del núcleo y la estructura de datos de este hilo del núcleo se mantiene dentro del espacio de direcciones del núcleo.

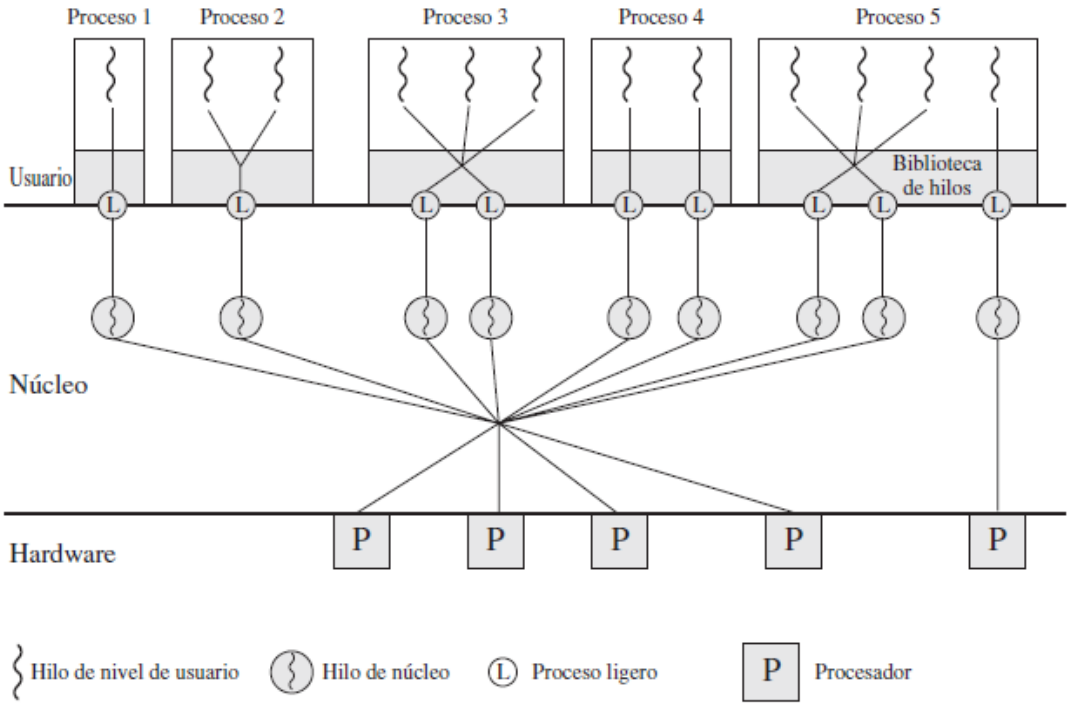


Figura 4.15. Ejemplo de arquitectura multihilo de Solaris.

La Figura 4.15 muestra la relación entre estas cuatro entidades. Nótese que hay siempre un hilo de núcleo por cada LWP. Un LWP es visible dentro de un proceso de la aplicación. De esta forma, las estructuras de datos LWP existen dentro del espacio de

direcciones del proceso respectivo. Al mismo tiempo, cada LWP está vinculado a un único hilo de núcleo activable, y la estructura de datos para ese hilo de núcleo se mantiene dentro del espacio de direcciones del núcleo.

Algunas referencias adicionales:
http://www.tau.org.ar/base/lara.pue.udlap.mx/sistoper/capitulo7.html
http://www.tau.org.ar/base/lara.pue.udlap.mx/sistoper/capitulo5.html
http://www.boost.org/doc/html/threads.html
http://www.flipcode.com/articles/article_multithreading.shtml

Lectura adicional (aplicativo):
Hilos en Windows
http://www.microsoft.com/spanish/msdn/comunidad/mtj.net/voices/art130.asp

Referencias

(s.f.). Obtenido de https://www.monografias.com/trabajos26/estados-proceso-hilos/estados-proceso-hilos

Galàn, S. (1 de enero de 1996). *Multiproceso simétrico*. Obtenido de https://www.dealerworld.es/archive/multiproceso-simetrico

Gustavo, L. C. (s.f.). *La arquitectura SMP*. Obtenido de https://es.scribd.com/document/106857115/La-Arquitectura-SMP#

Patricio, H. (s.f.). *La diferencia entre concurrencia y paralelismo*. Obtenido de https://blog.thedojo.mx/2019/04/17/la-diferencia-entre-concurrencia-y-paralelismo.html#:~:text=Se%20refiere%20a%20la%20ejecuci%C3%B3n,la%20suficiente%20memoria%20para%20mantenerlos.

Team, k. (1 de marzo de 2023). *¿Qué es el multiprocesamiento simétrico?* Obtenido de https://keepcoding.io/blog/que-es-multiprocesamiento-simetrico/#:~:text=El%20multiprocesamiento%20sim%C3%A9trico%20se%20entiende,el%20cumplimiento%20de%20sus%20funciones.

Vasquez, C. (13 de julio de 2016). *Funcionalidad de los hilos*. Obtenido de https://sistemasoperativos05blog.wordpress.com/2016/07/13/funcionalidad-de-los-hilos/

Planificación de proceso

Temas:

9.1. Tipos de planificación del procesador

9.2. Algoritmos de planificación

10.1. Planificación multiprocesador

10.2. Planificación de tiempo real

9.1. Tipos de planificación del procesador

El objetivo de la planificación de procesos es asignar procesos a ser ejecutados por el procesador o procesadores a lo largo del tiempo, de forma que se cumplan los objetivos del sistema tales como el tiempo de respuesta, el rendimiento y la eficiencia del procesador.

Tipos:

- **Planificación a largo plazo:** La decisión de añadir un proceso al conjunto de procesos a ser ejecutados.

  Determina qué programas se admiten en el sistema para su procesamiento. De esta forma, se controla el grado de multiprogramación. Una vez admitido, un trabajo o programa de usuario se convierte en un proceso y se añade a la cola del planificador a corto plazo. En algunos sistemas, un proceso de reciente creación comienza en la zona de intercambio, en cuyo caso se añaden a la cola del planificador a medio plazo.

- **Planificación a medio plazo:** La decisión de añadir un proceso al número de procesos que están parcialmente o totalmente en la memoria principal.

  Con frecuencia, la decisión de intercambio se basa en la necesidad de gestionar el grado de multiprogramación.

- **Planificación a corto plazo:** La decisión por la que un proceso disponible será ejecutado por el procesador.

  Se ejecuta más frecuentemente para tomar decisiones de intercambio. Se invoca siempre que ocurre un evento que puede conllevar el bloqueo del proceso actual y que puede proporcionar la oportunidad de expulsar al proceso actualmente en ejecución en favor de otro.

  Algunos ejemplos de estos eventos son:

  - Interrupciones de reloj.
  - Interrupciones de E/S.
  - Llamadas al sistema.
  - Señales (por ejemplo, semáforos).

La planificación expulsiva se utiliza cuando un proceso cambia del estado de ejecución al estado listo o del estado de espera al estado listo. Los recursos (principalmente ciclos de CPU) se asignan al proceso durante un período de tiempo limitado y luego se retiran, y el proceso se vuelve a colocar en la cola lista si aún le queda tiempo de ráfaga de CPU. Ese proceso permanece en la cola de espera hasta que tenga la próxima oportunidad de ejecutarse. Algoritmos con este mecanismo son Round Robin, Shortest Remaining Time Next, Priority (Preemptive version), etc.

La planificación no expulsiva se utiliza cuando finaliza un proceso o cuando un proceso cambia del estado de ejecución al estado de espera. En esta programación, una vez que los recursos (ciclos de la CPU) se asignan a un proceso, el proceso retiene la CPU hasta que finaliza o alcanza un estado de espera. En el caso de la programación no expulsiva, no interrumpe un proceso que se ejecuta en la CPU en medio de la ejecución. En cambio, espera hasta que el proceso complete su tiempo de ráfaga de CPU y luego puede asignar la CPU a otro proceso. Algoritmos con este mecanismo son Shortest Process Next, Priority (Non preemptive version).

- **Planificación de la E/S:** La decisión por la que un proceso que está pendiente de una petición de E/S será atendido por un dispositivo de E/S disponible.

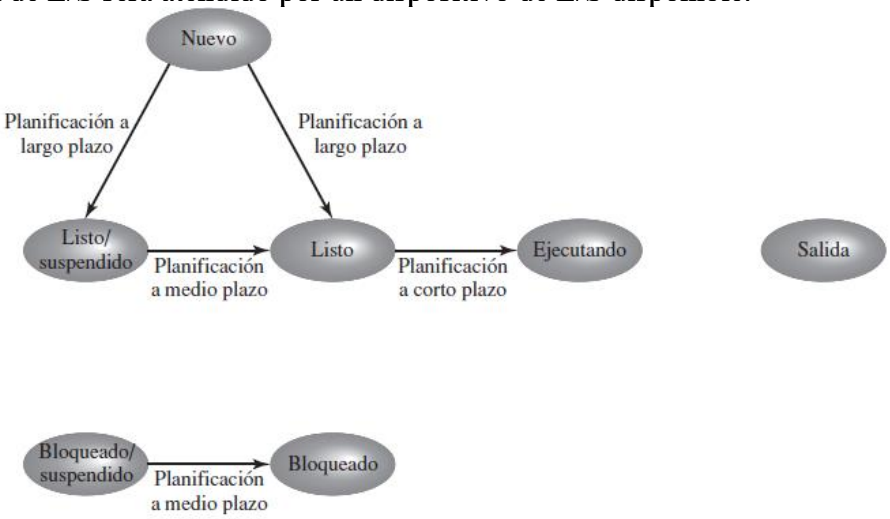


Figura 9.1. Planificación y transiciones de estado de los procesos.

9.2 Algoritmos de planificación

Criterios de la planificación a corto plazo

- **Tiempo de estancia (*turnaround time*)** Tiempo transcurrido desde que se lanza un proceso hasta que finaliza.
- **Tiempo de respuesta (*response time*)** Para un proceso interactivo, es el tiempo que trascurre desde que se lanza una petición hasta que se comienza a recibir la respuesta. La planificación debe intentar lograr bajos tiempos de respuesta y maximizar el número de usuarios interactivos con tiempos de respuesta aceptables.

- **Fecha tope (*deadlines*)** Cuando se puede especificar la fecha tope de un proceso, el planificador debe subordinar otros objetivos al de maximizar el porcentaje de fechas tope conseguidas.

Orientados al usuario, otros

- **Previsibilidad** Un trabajo dado debería ejecutarse aproximadamente en el mismo tiempo y con el mismo coste a pesar de la carga del sistema.

Orientados al sistema, relacionados con las prestaciones

- **Rendimiento** La política de planificación debería intentar maximizar el número de procesos completados por unidad de tiempo. Es una medida de cuánto trabajo está siendo realizado. Esta medida depende claramente de la longitud media de los procesos, pero está influenciada por la política de planificación, que puede afectar a la utilización.
- **Utilización del procesador** Es el porcentaje de tiempo que el procesador está ocupado. Para un sistema compartido costoso, es un criterio significativo. En un sistema de un solo usuario y en otros sistemas, tales como los sistemas de tiempo real, este criterio es menos importante que algunos otros.

Orientados al sistema, otros

- **Equidad** En ausencia de orientación de los usuarios o de orientación proporcionada por otro sistema, los procesos deben ser tratados de la misma manera, y ningún proceso debe sufrir inanición.
- **Imposición de prioridades** Cuando se asignan prioridades a los procesos, la política del planificador debería favorecer a los procesos con prioridades más altas.
- **Equilibrado de recursos** La política del planificador debería mantener ocupados los recursos del sistema. Los procesos que utilicen poco los recursos que en un determinado momento están sobre utilizados, deberían ser favorecidos. Este criterio también implica planificación a medio plazo y a largo plazo.

El uso de prioridades

Un problema de los esquemas de planificación con prioridades es que los procesos con prioridad más baja pueden sufrir inanición. Esto sucederá si hay siempre un conjunto de procesos de mayor prioridad listos para ejecutar.

Políticas de planificación alternativas

A medida que se describan las políticas de planificación se utilizará el conjunto de procesos de la Tabla 9.4 como ejemplo de ejecución. Podemos pensar en estos procesos como trabajos por lotes, con un tiempo de servicio igual al tiempo de ejecución requerido. También los podemos considerar como procesos en curso que requieren un uso alternativo del procesador y de la E/S de forma repetitiva. En este último caso, los tiempos de servicio representan el tiempo de procesador requerido en un ciclo. En cualquier caso, en términos de un modelo de colas, esta cantidad se corresponde con el tiempo de servicio.

Tabla 9.4. Ejemplo de planificación de procesos.

Proceso	Tiempo de llegada	Tiempo de servicio
A	0	3
B	2	6
C	4	4
D	6	5
E	8	2

Primero en llegar, primero en servirse (first-come-first-served).

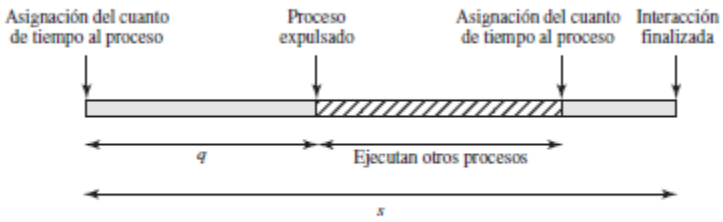
En el momento en que un proceso pasa al estado de listo, se une a la cola de listos.
FCFS no es una alternativa atractiva por sí misma para un sistema uniprocesador. Sin embargo, a menudo se combina con esquemas de prioridades para proporcionar una planificación eficaz. De esta forma, el planificador puede mantener varias colas, una por cada nivel de prioridad, y despachar dentro de cada cola usando primero en llegar primero en servirse.
FCFS funciona mucho mejor para procesos largos que para procesos cortos. Considérese el siguiente ejemplo, basado en [FINK88]:

Proceso	Tiempo de Llegada	Tiempo de Servicio (T_s)	Tiempo de Comienzo	Tiempo de Finalización	Tiempo de Estancia (T_r)	T_r/T_s
W	0	1	0	1	1	1
X	1	100	1	101	100	1
Y	2	1	101	102	100	100
Z	3	100	102	202	199	1,99
Media					100	26

Turno rotatorio (round robin)

Esta técnica es también conocida como cortar el tiempo (time slicing), porque a cada proceso se le da una rodaja de tiempo antes de ser expulsado.

El tema clave de diseño es la longitud del quantum de tiempo, o rodaja, a ser utilizada. Si el quantum es muy pequeño, el proceso se moverá por el sistema relativamente rápido. Por otra parte, existe una sobrecarga de procesamiento debido al manejo de la interrupción de reloj y por las funciones de planificación y activación. De esta forma, se deben evitar los quantums de tiempo muy pequeños.



(b) Cuanto de tiempo menor que la interacción típica

Figura 9.6. Efecto del tamaño del *quantum* de tiempo de expulsión.

Primero el proceso más corto (shortest process next)

Es una política no expulsiva en la que se selecciona el proceso con el tiempo de procesamiento más corto esperado. De esta forma un proceso corto se situará a la cabeza de la cola, por delante de los procesos más largos.

Un problema de la política SPN es la necesidad de saber, o al menos de estimar, el tiempo de procesamiento requerido por cada proceso.
La forma más sencilla de cálculo podría ser la siguiente:

$$S_{n+1} = \frac{1}{n}\sum_{i=1}^{n} T_i$$

donde,
Ti = tiempo de ejecución del procesador para la instancia *i*-ésima de este proceso (tiempo total de ejecución para los trabajos por lotes; tiempo de ráfaga de procesador para los trabajos interactivos)
Si = valor predicho para la instancia *i*-ésima
S1 = valor predicho para la primera instancia; no calculado

Para evitar volver a calcular la suma completa cada vez, podemos reescribir la ecuación como

$$S_{n+1} = \frac{1}{n}T_n + \frac{n-1}{n}S_n$$

Menor tiempo restante (shortest remaining time)

La política del menor tiempo restante (SRT) es una versión expulsiva de SPN. En este caso, a medida que entran procesos nuevos, el planificador escoge el proceso que tiene el menor tiempo restante por ejecución.

Primero el de mayor tasa de respuesta (highest response ratio next)

No podemos saber por adelantado el tiempo de servicio de los procesos, pero lo podemos aproximar, basándonos en la historia anterior, en alguna entrada de usuario o en un gestor de configuración. Considerar la siguiente tasa:

$$R = \frac{(w+s)}{s}$$

donde
R = tasa de respuesta
w = tiempo invertido esperando por el procesador
s = tiempo de servicio esperado

Si se despacha inmediatamente al proceso con este valor, R es igual al tiempo de estancia normalizado. Observar que el valor mínimo de R es 1,0, que sucede cuando un proceso acaba de entrar en el sistema.

La planificación contribución justa (fair-share scheduling)
El término *contribución justa* indica la filosofía que hay detrás de este planificador. A cada usuario se le asigna una prima de algún tipo que define la participación del usuario en los recursos del sistema como una fracción del uso total de dichos recursos. En particular, a cada usuario se le asigna una rodaja del procesador. Este esquema debería operar, más o menos, de forma lineal. Así, si un usuario A tiene el doble de prima que un usuario B, en una ejecución larga, el usuario A debería ser capaz de hacer el doble de trabajo que el usuario B.

El objetivo del planificador contribución justa es controlar el uso para dar menores recursos a usuarios que se han excedido de su contribución justa y mayores recursos a los que no han llegado.

10.1. Planificación multiprocesador

Podemos clasificar los sistemas multiprocesador como sigue:

- **Débilmente acoplado o multiprocesador distribuido, o cluster.** Consiste en una colección de sistemas relativamente autónomos; cada procesador tiene su propia memoria principal y canales de E/S.
- **Procesadores de funcionalidad especializada.** Un ejemplo es un procesador de E/S. En este caso, hay un procesador de propósito general maestro y procesadores especializados que son controlados por el procesador maestro y que le proporcionan servicios.
- **Procesamiento fuertemente acoplado.** Consiste en un conjunto de procesadores que comparten la memoria principal y están bajo el control integrado de un único sistema operativo.

Granularidad

Una buena manera de caracterizar los multiprocesadores y de situarlos en contexto respecto de otras arquitecturas, es considerar la granularidad de sincronización, o frecuencia de

sincronización entre los procesos del sistema. Podemos distinguir cinco categorías de paralelismo que difieren en el grado de granularidad.

Tabla 10.1. Granularidad de sincronización y procesos.

Tamaño del Grano	Descripción	Intervalo de sincronización (Instrucciones)
Fino	Paralelismo inherente en un único flujo de instrucciones	<20
Medio	Procesamiento paralelo o multitarea dentro de una única aplicación	20-200
Grueso	Multiprocesamiento de procesos concurrentes en un entorno multiprogramado	200-2000
Muy grueso	Procesamiento distribuido entre nodos de una red para conformar un único entorno de computación	2000-1M
Independiente	Múltiples procesos no relacionados	(N/D)

Planificación de procesos

En los sistemas multiprocesador más tradicionales, los procesos no se vinculan a los procesadores. En cambio, hay una única cola para todos los procesadores o, si se utiliza algún tipo de esquema basado en prioridades, hay múltiples colas basadas en prioridad, alimentando a un único colectivo de procesadores. En cualquier caso, el sistema puede verse como una arquitectura de colas multiservidor.

Planificación de hilos

En un monoprocesador, los hilos pueden usarse como una ayuda a la estructuración de programas y para solapar E/S con el procesamiento. Dada la mínima penalización por realizar un cambio de hilo comparado con un cambio de proceso, los beneficios se obtienen con poco coste.

Entre las muchas propuestas para la planificación multiprocesador de hilos y la asignación a procesadores, destacan cuatro enfoques generales:

- **Compartición de carga.** Los procesos no se asignan a un procesador particular. Se mantiene una cola global de hilos listos, y cada procesador, cuando está ocioso, selecciona un hilo de la cola.
- **Planificación en pandilla.** Un conjunto de hilos relacionados que se planifica para ejecutar sobre un conjunto de procesadores al mismo tiempo, en una relación uno-a-uno.
- **Asignación de procesador dedicado.** Esto es lo opuesto al enfoque de compartición de carga y proporciona una planificación implícita definida por la asignación de hilos a procesadores.

- **Planificación dinámica**. El número de hilos de un proceso puede cambiar durante el curso de su ejecución.

Planificación en pandilla

El concepto de planificar un conjunto de procesos simultáneamente sobre un conjunto de procesadores es anterior al uso de hilos.

- Si se ejecutan en paralelo procesos estrechamente relacionados, puede reducirse el bloqueo por sincronización, pueden necesitarse menos cambios de proceso y las prestaciones aumentarán.
- La sobrecarga de planificación puede reducirse dado que una decisión única afecta a varios procesadores y procesos a la vez.

Asignación de procesador dedicado

Pueden realizarse dos observaciones en defensa de esta estrategia:

- En un sistema altamente paralelo, con decenas o cientos de procesadores, cada uno de los cuales representa una pequeña fracción del coste del sistema, la utilización del procesador deja de ser tan importante como medida de la eficacia o rendimiento.
- Evitar totalmente el cambio de proceso durante la vida de un programa debe llevar a una sustancial mejora de velocidad de ese programa.

Planificación dinámica

Cuando un trabajo solicita uno o más procesadores (bien cuando el trabajo llega por primera vez o porque sus requisitos cambian),

- Si hay procesadores ociosos, utilizarlos para satisfacer la solicitud.
- En otro caso, si el trabajo que realiza la solicitud acaba de llegar, ubicarlo en un único procesador quitándoselo a cualquier trabajo que actualmente tenga más de un procesador.
- Si no puede satisfacerse cualquier parte de la solicitud, mantenerla pendiente hasta que un procesador pase a estar disponible, o el trabajo rescinda la solicitud (por ejemplo, si dejan de ser necesarios los procesadores extra). Cuando se libere uno o más procesadores (incluyendo la terminación de un trabajo),
- Examinar la cola actual de solicitudes de procesador no satisfechas. Asignar un único procesador a cada trabajo en la lista que no tenga actualmente procesadores (por ejemplo, a todos los recién llegados en espera). Luego volver a examinar la lista, volviendo a asignar el resto de los procesadores siguiendo la estrategia FCFS.

Compartición de carga

La compartición de carga es posiblemente el enfoque más simple y que se surge más directamente de un entorno monoprocesador.

[LEUT90] analiza tres versiones diferentes de compartición de carga:

- **Primero en llegar, primero en ser servido (FCFS).** Cuando llega un trabajo, cada uno de sus hilos se disponen consecutivamente al final de la cola compartida. Cuando un procesador pasa a estar ocioso, coge el siguiente hilo listo, que ejecuta hasta que se completa o se bloquea.
- **Menor número de hilos primero.** La cola compartida de listos se organiza como una cola de prioridad, con la mayor prioridad para los hilos de los trabajos con el menor número de hilos no planificados. Los trabajos con igual prioridad se ordenan de acuerdo con qué trabajo llega primero. Al igual que con FCFS, el hilo planificado ejecuta hasta que se completa o se bloquea.
- **Menor número de hilos primero con expulsión.** Se les da mayor prioridad a los trabajos con el menor número de hilos no planificados. Si llega un trabajo con menor número de hilos que un trabajo en ejecución se expulsarán los hilos pertenecientes al trabajo planificado.

10.2. Planificación de tiempo real

Características de los sistemas operativos de tiempo real

Los sistemas operativos de tiempo real pueden ser caracterizados por tener requisitos únicos en cinco áreas generales [MORG92]:

- Determinismo
- Reactividad
- Control del usuario
- Fiabilidad
- Operación de fallo suave

Para cumplir los requisitos precedentes, los sistemas operativos de tiempo real incluyen de forma representativa las siguientes características [STAN89]:

- Cambio de proceso o hilo rápido.
- Pequeño tamaño (que está asociado con funcionalidades mínimas).
- Capacidad para responder rápidamente a interrupciones externas.
- Multitarea con herramientas para la comunicación entre procesos como semáforos, señales y eventos.

- Utilización de ficheros secuenciales especiales que pueden acumular datos a alta velocidad.
- Planificación expulsiva basada en prioridades.
- Minimización de los intervalos durante los cuales se deshabilitan las interrupciones.
- Primitivas para retardar tareas durante una cantidad dada de tiempo y para parar/retomar tareas.
- Alarmas y temporizaciones especiales.

Planificación de tiempo real

En base a estas consideraciones los autores identifican las siguientes clases de algoritmos:

- **Enfoques estáticos dirigidos por tabla.** En éstos se realiza un análisis estático de la factibilidad de la planificación. El resultado del análisis es una planificación que determina cuando, en tiempo de ejecución, debe comenzar a ejecutarse cada tarea.
- **Enfoques estáticos expulsivos dirigidos por prioridad.** También se realiza un análisis estático, pero no se obtiene una planificación. En cambio, el análisis se utiliza para asignar prioridades a las tareas, y así puede utilizarse un planificador expulsivo tradicional basado en prioridades.
- **Enfoques dinámicos basados en un plan.** La factibilidad se determina en tiempo de ejecución (dinámicamente) en vez de antes de comenzar la ejecución (estáticamente). Una nueva tarea será aceptada como ejecutable sólo si es posible satisfacer sus restricciones de tiempo. Uno de los resultados del análisis de factibilidad es un plan que se usará para decidir cuándo poner en marcha la tarea.
- **Enfoques dinámicos de mejor esfuerzo.** No se realiza análisis de factibilidad. El sistema intenta cumplir todos los plazos y aborta la ejecución de cualquier proceso cuyo plazo haya fallado.

Planificación por plazos

Generalmente, las aplicaciones de tiempo-real no se preocupan tanto de la velocidad de ejecución como de completar (o comenzar) sus tareas en los momentos más adecuados, ni muy pronto ni muy tarde, a pesar de la demanda dinámica de recursos u otros conflictos, sobrecarga de procesamiento y fallos hardware o software.

En su forma más general, puede utilizarse la siguiente información de cada tarea:

- **Tiempo de activación.** Momento en el cual la tarea pasa a estar lista para ejecutar. En el caso de una tarea repetitiva o periódica se tratará de una secuencia de tiempos conocida de antemano.

- **Plazo de comienzo.** Momento en el cual la tarea debe comenzar.

- **Plazo de conclusión.** Momento para el cual la tarea debe estar completada. Las aplicaciones de tiempo real típicas tendrán plazos de comienzo o bien plazos de conclusión, pero no ambos.

- **Tiempo de proceso.** Tiempo necesario para ejecutar la tarea hasta su conclusión.

- **Recursos requeridos.** Conjunto de recursos (distintos del procesador) que la tarea necesita
- durante su ejecución.

- **Prioridad.** Mide la importancia relativa de la tarea. Las tareas de tiempo real duro pueden tener una prioridad «absoluta», provocando que el sistema falle si algún plazo no se cumple.

- **Estructura de subtareas.** Una tarea puede ser descompuesta en una subtarea obligatoria y una subtarea opcional. Sólo la subtarea obligatoria posee un plazo duro.

Planificación de tasa monótona

Uno de los métodos más prometedores para resolver los conflictos de planificación multitarea para tareas periódicas es la planificación de tasa monótona (RMS). El esquema fue propuesto por primera vez en [LIU73] pero sólo recientemente ha ganado popularidad [BRIA99, SHA94]. RMS asigna prioridades a las tareas en base a sus periodos.

Para el RMS la tarea de mayor prioridad es aquélla con el periodo más breve, la segunda tarea de mayor prioridad es aquélla con el segundo periodo más breve, y así sucesivamente. Cuando hay más de una tarea disponible para su ejecución, aquella con el periodo más breve se sirve primero.

Bibliografía

Stallings, W. (s.f.). Sistema Operativo 5 edición. En Pearson, *Aspectos internos y principio de diseño.*

https://www.geeksforgeeks.org/preemptive-and-non-preemptive-scheduling/