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.*

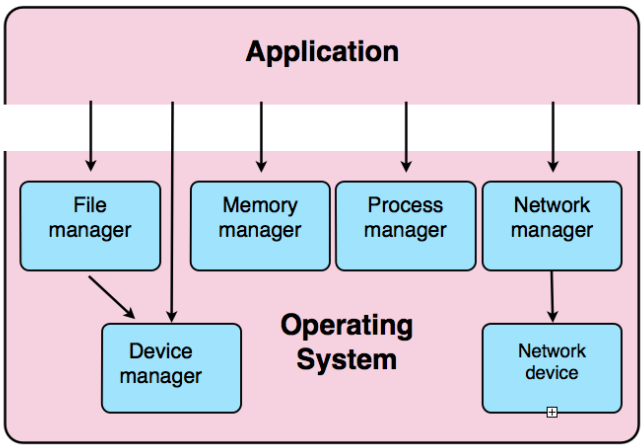
    - *Problema: sincronización de los módulos que componen el micronúcleo y su acceso a la memoria e integración con aplicaciones.*

    - *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.

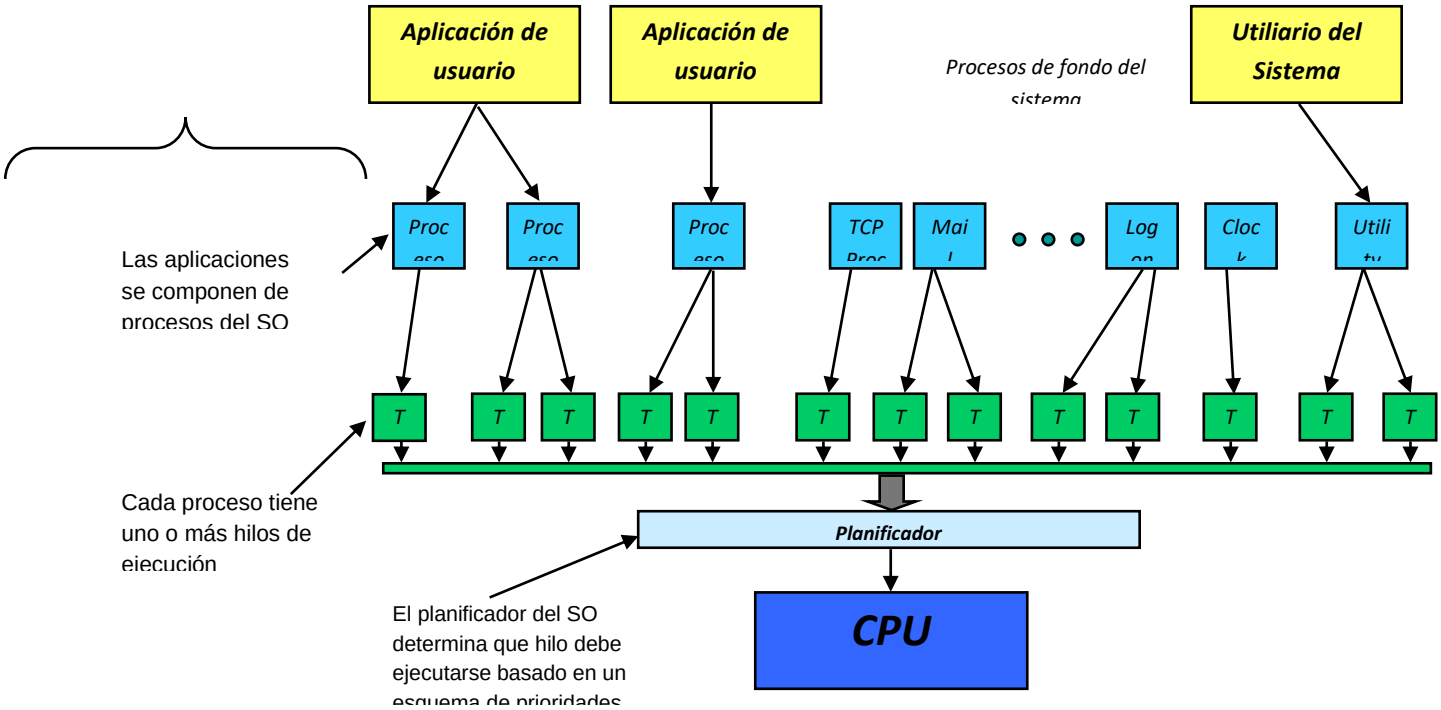
Cuál es la diferencia entre un procesador y un proceso?

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.*

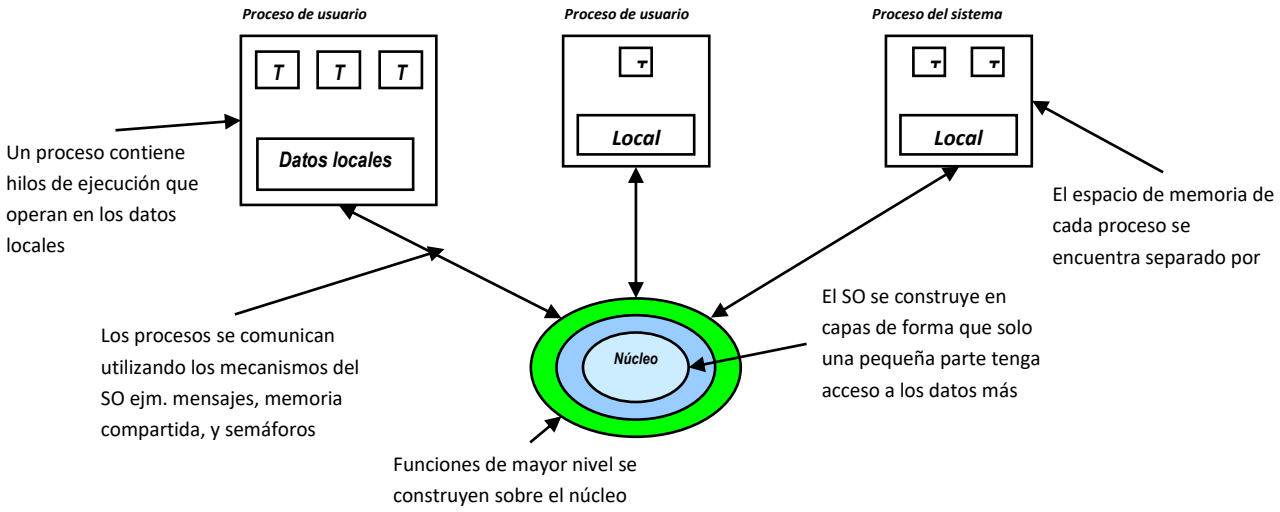
En un sistema monotarea el manejo de las interrupciones es bastante complicado debido a que solo puede ejecutar una tarea a la vez, por lo que, para procesar la interrupción, el sistema operativo tiene que intercambiar entre miles de procesos, miles de veces en un segundo. Este proceso se describe a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

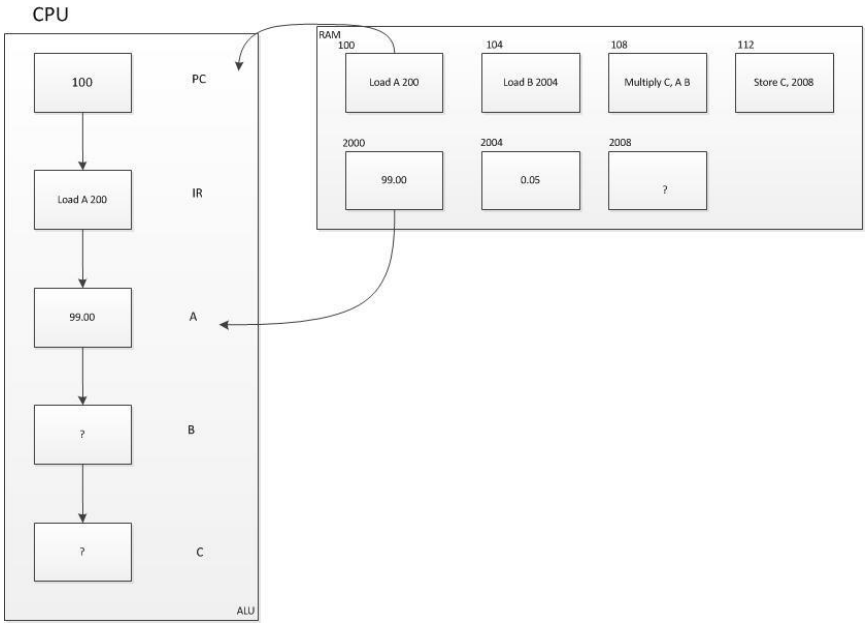


La figura muestra la interacción de los procesos con el Sistema Operativo.

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.*

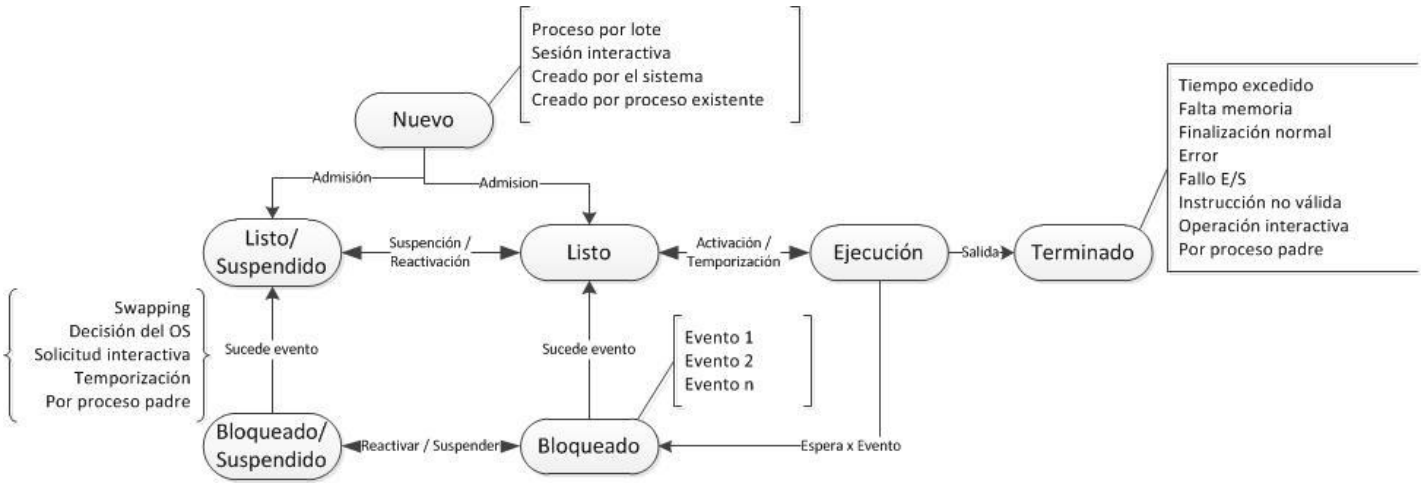
11. *Se incrementa el valor de PC a la siguiente instrucción.*

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**:*

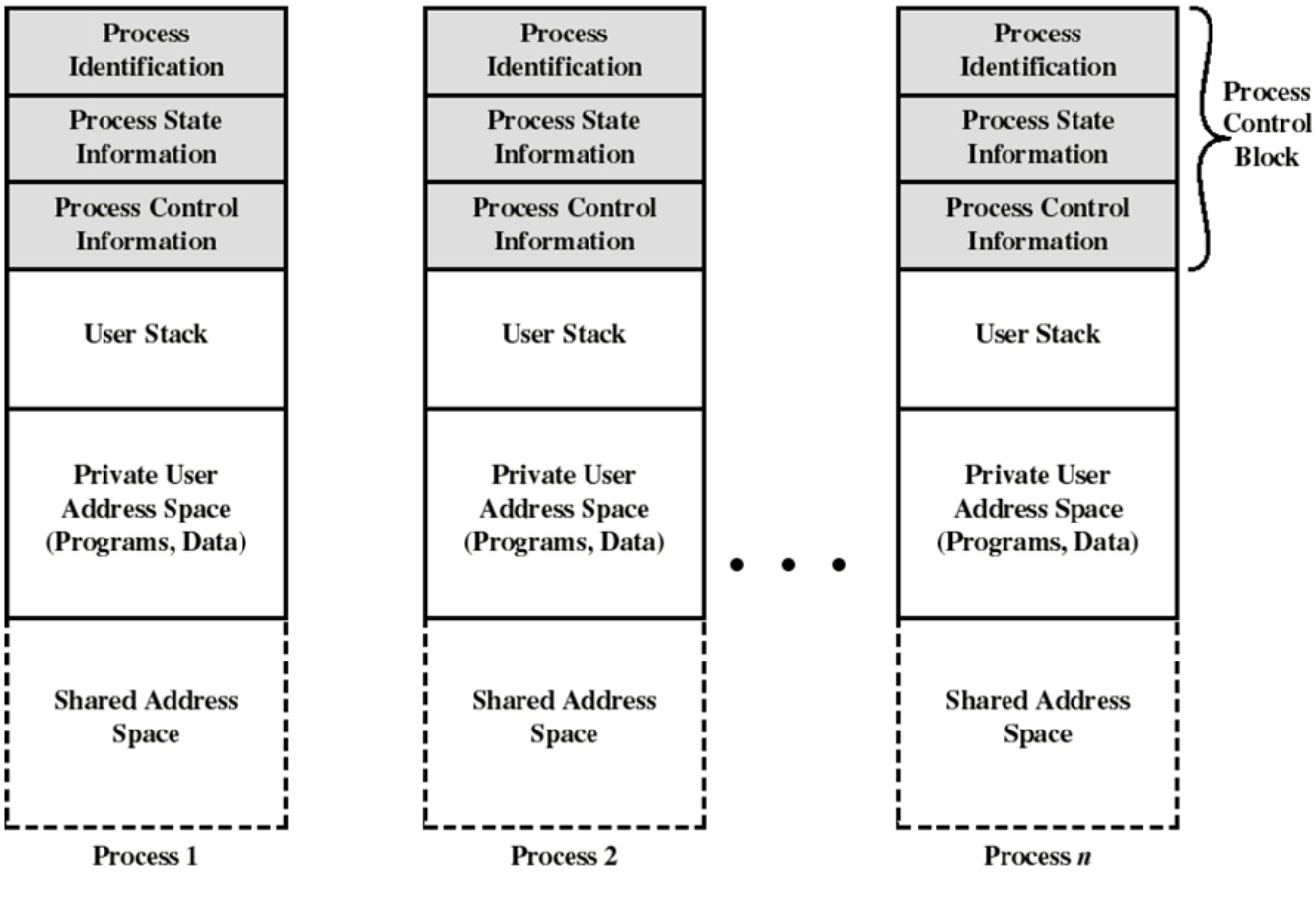
- *Bloqueado / suspendido: los procesos están en memoria secundaria, esperando por un evento.*
- *Listo / suspendido: los procesos están en memoria secundaria, pero está disponible para ejecución tan pronto como sea cargado en memoria principal, cuando se libere algún espacio.*

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.

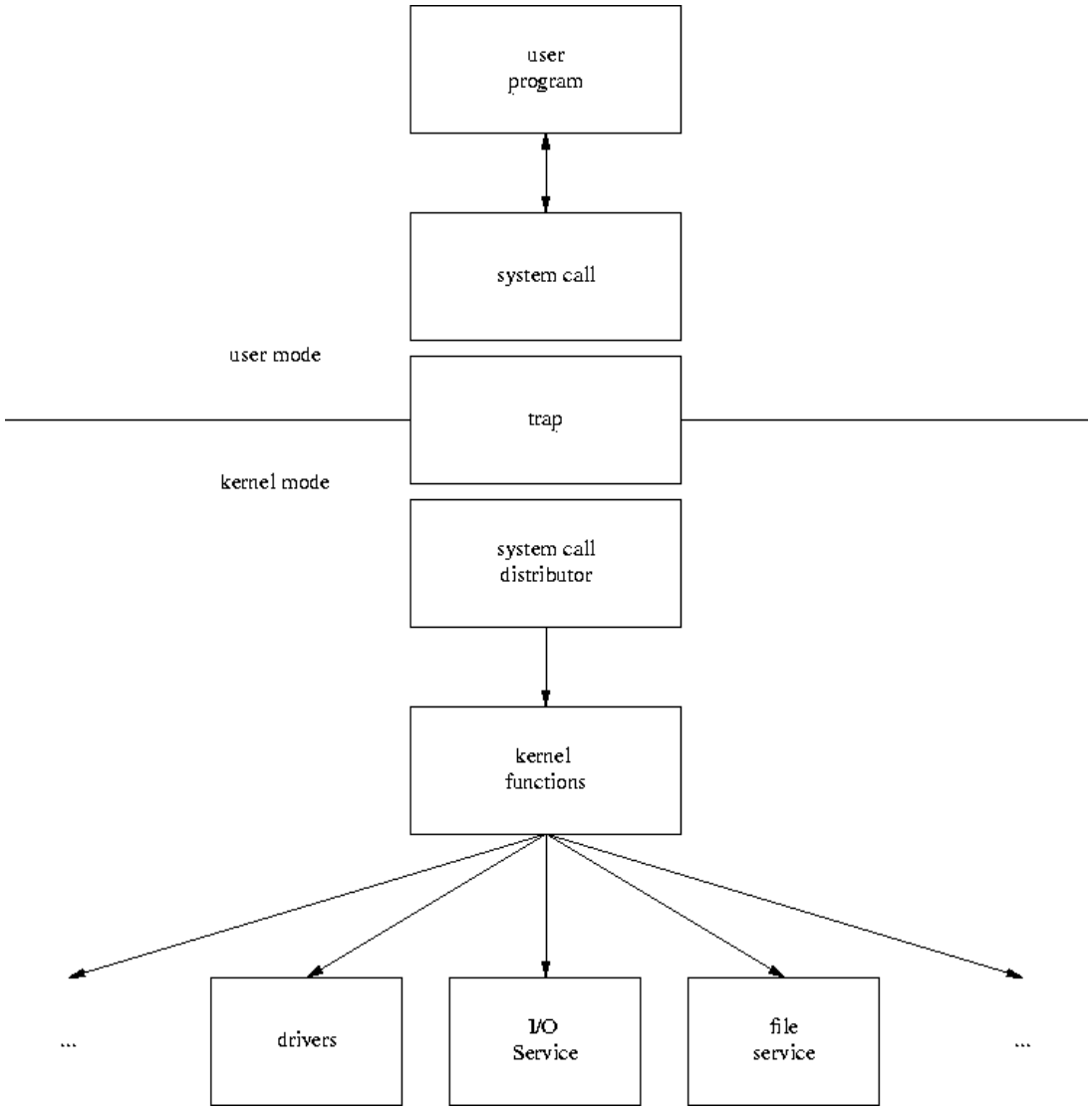
Cambio de Modo vs Cambio de Proceso

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.

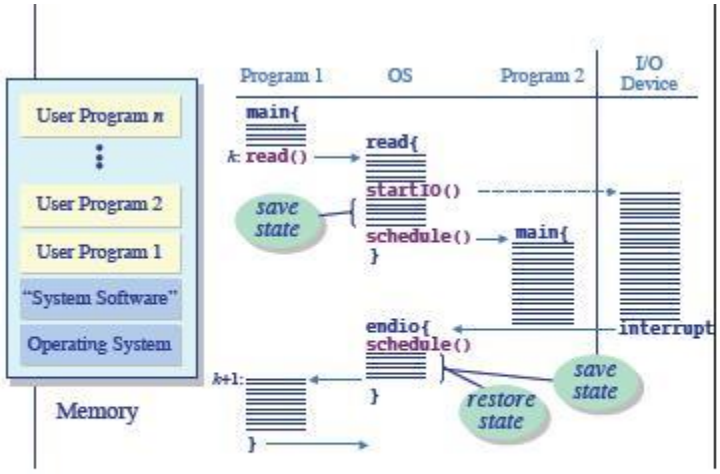
El cambio de proceso involucra:

- *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.*

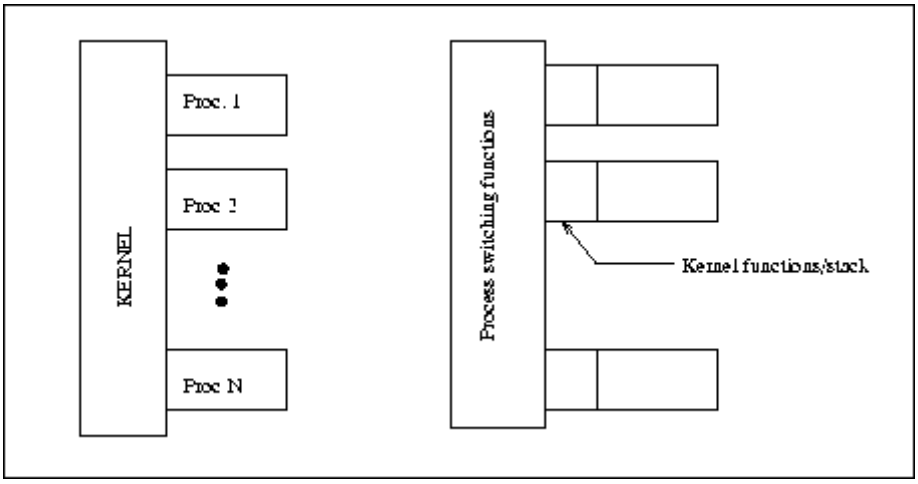
Gestión de memoria: *asegurar consistencia cuando varios procesadores comparten una página o segmento y por el reemplazo de los mismos.*

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.*



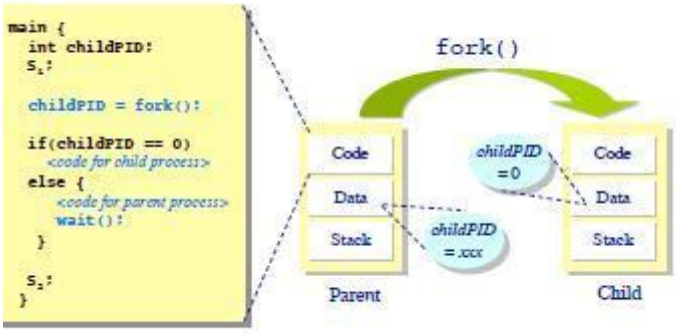
Fuente: Ejecución del OS [11]

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*