Sistemas Operativos

Diego Enrique Fontán, CosasDePuma

2017-18

Índice

1. Niveles de software **1**
1.1. Concepto de Sistema Operativo 1
1.1.1. Características del Sistema Operativo 1
1.1.2. Funciones principales del Sistema Operativo 2

2. Tipos de Sistemas **3**
2.1. Sistemas monolíticos 3
2.2. Sistemas en estratos 3
2.3. Máquinas virtuales 4
2.4. Modelo Cliente-Servidor 4
2.5. Estructura orientada al objeto 5
2.6. Sistemas híbridos 5

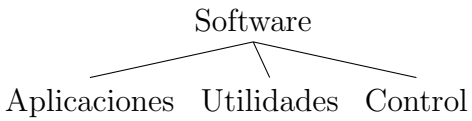
3. Procesos **6**
3.1. Definiciones básicas 6
3.2. Concurrencia. Programa Concurrente 8
3.2.1. Tipos de concurrencia 8
3.3. Comunicación entre procesos 9
3.3.1. Exclusión mutua 9
3.3.2. Sincronización 10
3.4. Interbloqueos o Deadlocks 11
3.4.1. Condiciones necesarias para el interbloqueo 11
3.4.2. Prevención del interbloqueo 12
3.5. Estados de un proceso 13
3.5.1. Bloque de Control de Procesos (PCB) 14
3.5.2. Bloque de Control del Sistema (SCB) 14
3.5.3. Operaciones básicas sobre procesos 14

4. El núcleo del Sistema Operativo. El kernel **15**
4.1. Funciones básicas del kernel 15

5. Memoria Virtual: Estructura y organización **16**
5.1. Paginación 16

1. Niveles de software

Se denomina **soporte lógico** o **software** a todo aquel conjunto de programas asociados a un ordenador.



Niveles de software

1.1. Concepto de Sistema Operativo

Un **Sistema Operativo** es un programa (o conjunto de subprogramas o módulos) de control que tiene como finalidad facilitar el uso del ordenador y conseguir que se utilice eficientemente.

1.1.1. Características del Sistema Operativo

- Actúa como interfaz entre el usuario y la máquina física.

- Controla la ejecución de otros programas.

- Gestiona y mantiene ficheros.

- Contabiliza la utilización de los recursos.

- Protege los datos y los programas.

- Gestiona y asigna directamente los recursos hardware.

⋮

1.1.2. Funciones principales del Sistema Operativo

El Sistema Operativo debe inicializar la máquina y preparar el ordenador para su funcionamiento mediante una **inicialización total** (Initial Program Loading, Bootstrapping) o mediante una **inicialización parcial**.

También servirá de **máquina extendida o virtual** con el fin de ocultar los detalles del hardware al usuario y proporcionar un entorno más cómodo. De esta forma se logran varios objetivos:

- Evitar que la ejecución de los programas se interfieran unos entre otros, proporcionando así **seguridad**.

- Construir recursos virtuales de alto nivel a partir de recursos físicos de de más bajo nivel.

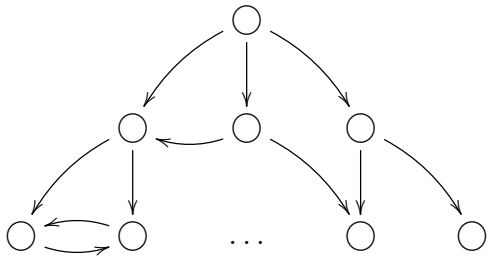
Otra de las funciones principales de un Sistema Operativo es la de **administrar los recursos para su funcionamiento**, ya sea *asignandole* todos los que requiera un programa para su ejecución o *controlando* el uso corrector de estos.

Además, es importante que un Sistema Operativo sea **determinista**, para que un mismo programa ejecutado con los mismo datos de los mismos resultados en cualquier momento y en cualquier ejecución, y que sea **indeterminista**, para que pueda responder a circunstancias que ocurren de manera impredecible.

2. Tipos de Sistemas

2.1. Sistemas monolíticos

No tienen una estructrura definida. Se componen de un conjunto de procedimientos donde cada uno de ellos puede llamar a todos los demás.



2.2. Sistemas en estratos

Se organiza en una jerarquía de estratos, estando construído cada uno de ellos sobre el otro que tiene menor jerarquía que él.

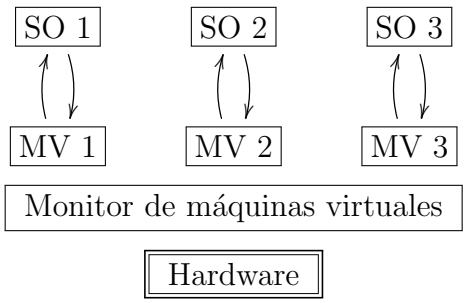
Operador del SO
Programas de usuario
Administración de E/S
Comunicación operador-procesos
Administración de memoria y tambor
Distribución del procesador y multiprogramación

Ejemplo de sistema THE

2.3. Máquinas virtuales

Máquinas similares a la máquina real pero de carácter virtual.

El programa de control se ejecuta sobre el propio hardware y ofreec al nivel superior varias máquinas virtuales.



2.4. Modelo Cliente-Servidor

Su objetivo es minimizar el kernel desplazando el código de todos sus servicios a estratos lo más superiores posibles.

Para ello, la mayoría de sus funciones se implementan como procesos del servidor, denominados **procesos servidores**, de forma que cuando un proceso de usuario, llamado **proceso cliente**, necesita un servicio del SO, lo que hace es enviar un mensaje al servidor correspondiente (el cual realiza el trabajo y devuelve la respuesta).

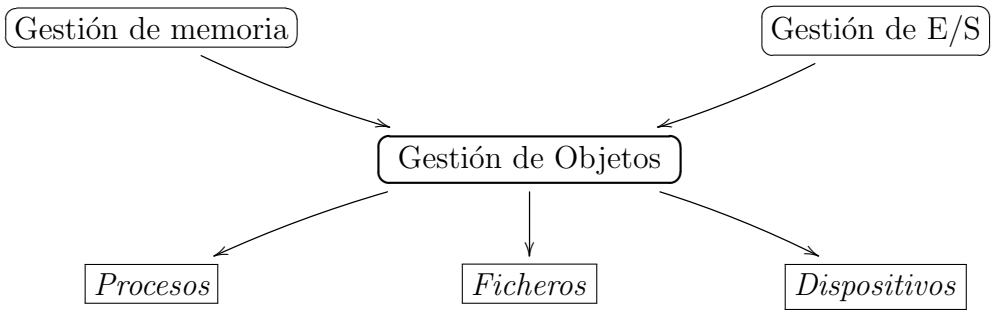
El kernel. lo único que hace es implementar la comunicación entre cliente-servidor y entre servidor-hardware.

2.5. Estructura orientada al objeto

Se basan en una colección de objetos, donde las funciones del sistema son ficheros, dispositivos, etc.

La interacción entre dichos objetos viene determinada por las capacidades que cada uno tenga para actuar con el otro.

El kerner es el responsable del mantenimiento de las definiciones de los tipos de objetos soportados y del control de los privilegios de acceso de los mismos.



2.6. Sistemas híbridos

Son similares a los sistemas cliente-servidor, aunque añaden ciertas funcionalidades al kernel para que se ejecute más rápido que si permanecen en el espacio de usuario.

Se les llama híbridos porque usan mecanismos o conceptos de arquitectura de los sistemas monolíticos y de los sistemas cliente-servidor.

3. Procesos

3.1. Definiciones básicas

Multiprogramación: Característica que permite que más de un programa resida en la memoria principal al mismo tiempo.

Al número de programas almacenados simultáneamente en memoria en un momento dado se denomina grado de multiprogramación.

Programa: Secuencia de instrucciones o acciones definidas mediante un lenguaje de programación y que pueden ser ejecutadas por parte de un procesador.

Proceso: Secuencia de acciones derivadas de la ejecución de una serie de instrucciones definidas a priori.

Destacar que un proceso puede requqerir la ejecución de uno o varios programas y, por contrapartida, un programa puede formar parte de más de un proceso.

Un programa siempre es el mismo, es decir, es una entidad estática y pasiva, pero un proceso puede variar bastante según esté el sistema en cada momento (entidad dinámica y activa).

El programa o programas asociados a un proceso no tienen porqué estar implementados en el software.

Procesador: Agente que lleva a cabo un proceso que ejecuta un programa asociado.

Concurrencia: Solapamiento en el tiempo de la ejecución de varias actividades.

Contexto o entorno volátil: Estado actual de un proceso (valor de registros de la CPU, memoria asignada a procesos, etc.).

Conmutación de la CPU: Pérdida del control de la CPU por parte de un proceso para que ésta se le asigne a otro proceso.

Esto implica guardar el entorno volátil del proceso que pierde la CPU y restaurar el contexto del proceso al que se le asigna.

Overhed, sobrecarga o carga adicional: Tiempo de CPU empleado en la ejecución de los procesos del sistema.

Procesamiento Concurrente: Situación obtenida al hacer una instantánea del sistema. En este caso encontraremos varios procesos que se encuentran en un estado intermedio entre su estado inicial y final.

Programación Concurrente: Conjunto de notaciones que se usan para expresar paralelismo y conjunto de técnicas que se usan para resolver posibles conflictos entre procesos.

3.2. Concurrencia. Programa Concurrente

Dos procesos, P1 y P2, se ejecutan concurrentemente si la primera instrucción de P1 se ejecuta entre la primera y la última instrucción de P2.

3.2.1. Tipos de concurrencia

Si el número de procesadores es mayor o igual al número de procesos, entonces se da concurrencia real, también llamada paralelismo.

Figura 1: Real o paralela

Si el número de procesadores es menor al número de procesos, entonces se da concurrencia aparente o simulada, también llamada pseudoparalelismo.

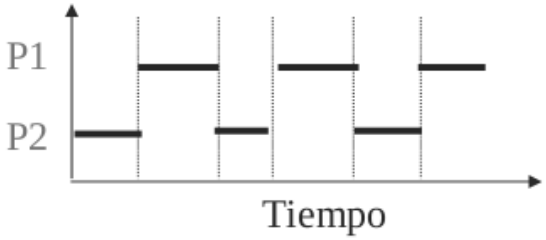


Figura 2: Aparente, simulada o pseudoparalela

3.3. Comunicación entre procesos

Los procesos de un sistema no actúan de forma aislada.

A veces tiene que cooperar para alcanzar un objetivo común. A este tipo de comunicación se le llama **sincronización**.

Por contrapartida, a veces tienen que competir por el uso de recursos. Esto es lo que se denomina **exclusión mutua**.

3.3.1. Exclusión mutua

Los tipos de recursos que existen en exclusión mutua son:

- **Compartibles:** Pueden ser usados por varios procesos de forma concurrente.

- **No compartibles:** Su uso está restringido a un sólo proceso a la vez hasta que el proceso que está usando dicho recurso finaliza su utilización.

Es por ello que la exclusión mutua se da cuando dos o más procesos se están ejecutando en paralelo y necesitan a la vez el uso de un recurso no compartible.

En este caso, el recurso no compartible se le asigna sólamente a un proceso y el resto queda en espera hasta que el primero finalice de usarlo. Cuando esto ocurra, el recurso será asignado a uno de los procesos en espera. Con ello se asegura *el correcto uso del recurso*.

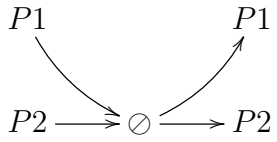
El trozo de código donde un proceso hace uso de un recurso no compartible se denomina **sección crítica o región crítica** y deben ser ejecutadas lo más rápido posible y cuidadosamente codificadas.

3.3.2. Sincronización

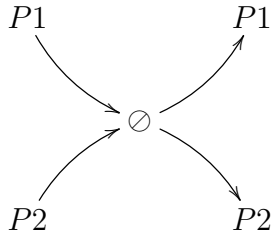
La sincronización es la comunicación requerida entre dos o más procesos con el fin de sincronizar sus actividades.

Existen dos tipos de sincronización generales:

- **Sincronización simple:** Un proceso, P1, llegado a cierto punto clave, no puede proseguir con su ejecución, hasta que otro proceso, P2, haya llegado a otro punto clave de ejecución.



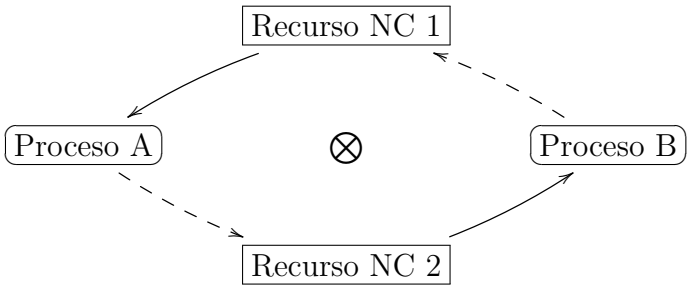
- **Sincronización múltiple:** Un proceso, P1, llegado a cierto punto clave de su ejecución no puede proseguir hasta que otro proceso, P2, no haya llegado a dicho punto, y viceversa.



3.4. Interbloqueos o Deadlocks

Un proceso está **interbloqueado** si está esperando por un evento determinado que nunca va a ocurrir.

Un conjunto de procesos está **interbloqueado** cuando cada uno de ellos está esperando un suceso que sólo puede ser causado por otro proceso del mismo conjunto y que nunca se producirá porque están bloqueados.



Interbloque entre dos procesos por el uso de recursos no compartibles

Un proceso está **postergado indefinidamente** cuando espera por un evento que puede ocurrir pero no se sabe cuando.

Un proceso puede quedar esperando indefinidamente si no puede proseguir su ejecución debido al continuo procesamiento de otro proceso por parte del Sistema Operativo.

3.4.1. Condiciones necesarias para el interbloqueo

- **Exclusión mutua:** Los procesos reclaman control excesivo de los recursos que piden.

- **Esperar por:** Los procesos mantienen los recursos que ya les han asignado mientras esperan por recursos adicionales.

-**No apropiatividad:** Los recursos no pueden ser extraídos de los procesos que los mantienen hasta su completa utilización.

- **Espera circular:** Existe una cadena circular de procesos en la cual uno de ellos mantiene uno o más recursos que son requeridos por el siguiente proceso de la cadena.

3.4.2. Prevención del interbloqueo

Para prevenir el interbloqueo, podemos realizar tres acciones:

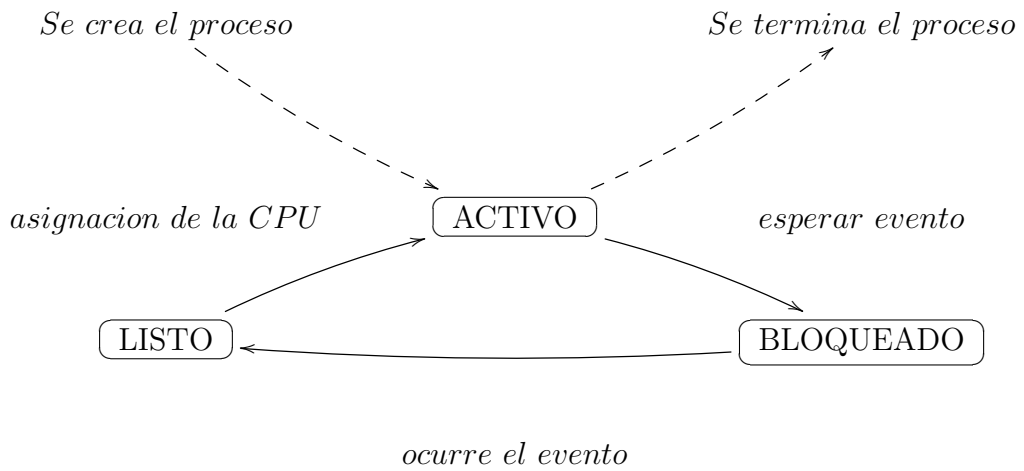
- **Negar la condición de *esperar por*:** Cada proceso debe pedir todos los recursos que va a necesitar de golpe. Si el conjunto de todos ellos está disponible, se le asigna todos. En caso contrario, no se le asigna a ningún proceso y tendrá que esperar hasta que todos estén disponibles.

- **Negar la condición de *no apropiatividad*:** Cuando un proceso que tiene recursos le es negada una petición de recursos adicionales, deberá liberar sus recursos y, si es necesario, pedirlos de nuevo junto a los recursos adicionales.

- **Negar la condición de *espera circular*:** Cuando se intala un recurso, se le asigna un número exclusivo, de forma que los procesos deben solicitar los recursos en orden ascendente.

Con esto podemos prevenirlo **evitando** el interbloqueo condicionando al sistema para que esquive determinadas situaciones propensas a ello, **detectando** y determinando si existe o no dicho interbloqueo identificando qué procesos y recursos están implicados en él o **recuperarnos** de uno retirando uno o más condiciones necesarias.

3.5. Estados de un proceso



Un proceso está **activo** cuando tiene el control de un procesador.

Un proceso está **bloqueado** cuando está esperando a que ocurra un evento.

Un proceso está **listo** cuando está esperando para usar un procesador.

3.5.1. Bloque de Control de Procesos (PCB)

El Bloque de Control de Procesos o PCB es una estructura de datos que contiene toda la información de un proceso que el Sistema Operativo necesita para su control.

Guarda parámetros de estado volátil, como el estado actual, identificación, punteros a memoria, punteros de recursos, etc.

3.5.2. Bloque de Control del Sistema (SCB)

El Bloque de Control del Sistema o SCB es una estructura de datos que gestiona el Sistema Operativo para controlar todos los PCB's de los procesos. Reside en memoria principal por su alta frecuencia de consulta.

Guarda parámetros como la lista de PCB's, punteros a los PCB's activos, puntero a la lista de procesos listos, puntero a la lista de procesos bloqueados, etc.

3.5.3. Operaciones básicas sobre procesos

- **Crear un proceso**: dar nombre al proceso, insertarlo en la lista de procesos en estado listo, determinar su prioridad inicial, crear el PCB del proceso y asignarles los recursos iniciales del proceso.

- **Destruir un proceso**: devolver al sistema los recursos que tiene asignado, eliminarlo de todas las listas del sistema y borrar su PCB.

- **Cambiar la prioridad de un proceso**.

- **Bloquear un proceso**.

- **Despertar un proceso**.

- **Despachar un proceso**.

4. El núcleo del Sistema Operativo. El kernel

El kernel se ejecuta con las interrupciones desactivadas (**modo privilegiado**) y es la parte del Sistema Operativo más cercana al hardware, por ello suele estar codificado en ensamblador o en lenguaje máquina y reside en la memoria principal.

4.1. Funciones básicas del kernel

a) Manipulación de interrupciones.

b) Inhabilitación y habilitación de interrupciones.

c) Creación y destrucción de procesos.

d) Cambio de estado de un proceso.

e) Despachar un proceso.

f) Comunicación entre procesos.

g) Manipulación de los PCB's.

h) Soporte para servicios de más alto nivel.

5. Memoria Virtual: Estructura y organización

5.1. Paginación

La paginación de memoria virtual en dividir el espacio virtual de direcciones de un trabajo en trozos del mismo tamaño, denominados **páginas** o **páginas lógicas**.

También consiste en dividir la memoria principal en bloques del mismo tamaño que el tamaño máximo de las páginas lógicas, a lo cual denominamos **marcos de páginas** o **páginas físicas**.

Memoria Principal		Memoria Secundaria	
Página 0			Marco 3
Página 1		Página 2	Marco 2
Página 2			Marco 1
Página 0		Página 0	Marco 0

Esta organización tiene las siguientes características:

- Los marcos de páginas van a empezar en direcciones de almacenamiento real que son múltiplos enteros del tamaño del marco de página. Los marcos ocupan:

De **posicion*tamano** *hasta* **(posicion+1)*tamano-1**

- El intercambio de información se hace a nivel de páginas. Es decir, se copian páginas completas a cualquier marco de página disponible.

- Las páginas consecutivas de un trabajo en el espacio virtual de direcciones no tiene por qué ocupar marcos de páginas consecutivas en la memoria principal.

Para realizar la traducción dinámica de direcciones, cada trabajo tiene asociado una **tabla de mapas de páginas**, con tantos elementos como el número de páginas en las que se ha dividido el trabajo.

El formato típico de de un elemento de la tabla de páginas sería:

- t_p **:** bit de residencia de la página en memoria principal (normalmente $t_p = 0$ si la página no reside y $t_p = 1$ si la página reside).

- m_p **:** dirección donde está la página en la memoria secundaria.

- p' **:** dirección real del comienzo del marco de página (a veces no contiene dicha dirección, sino el número de marco de página).