
1.7 ESTRUCTURA DEL SISTEMA OPERATIVO

Ahora que hemos estudiado el aspecto externo de los sistemas operativos (o sea, la interfaz del programador), es momento de dar un vistazo al interior. En las secciones que siguen examinaremos cinco estructuras que se han probado, a fin de tener una idea de la gama de posibilidades. Desde luego, no es una muestra exhaustiva, pero da una idea de los diseños que se han probado en la práctica. Los cinco diseños son: sistemas monolíticos, sistemas en capas, máquinas virtuales, exokernels y sistemas cliente-servidor.

1.7.1 Sistemas monolíticos

Esta organización, que por mucho es la más común, bien podría calificarse como “El Gran Embrollo”. La estructura consiste en que no hay estructura. El sistema operativo se escribe como una colección de procedimientos, cada uno de los cuales puede invocar a cualquiera de los otros cuando lo necesita. Si se utiliza esta técnica, cada procedimiento del sistema tiene una interfaz bien definida desde el punto de vista de parámetros y resultados, y cada una está en libertad de invocar a cualquier otra, si ésta realiza alguna operación útil que la primera necesita.

Para construir el programa objeto del sistema operativo cuando se adopta este enfoque, lo primero que se hace es compilar todos los procedimientos individuales, o archivos que contienen los procedimientos, y luego unirlos todos en un solo archivo objeto, utilizando el enlazador del sistema. Respecto al ocultamiento de la información, éste prácticamente no existe: cualquier procedimiento puede ver a cualquier otro (en contraposición con una estructura que contiene módulos o paquetes, en la que gran parte de la información queda oculta dentro de módulos, y desde afuera del módulo sólo se pueden invocar los puntos de ingreso oficialmente designados).

Incluso en los sistemas monolíticos, empero, es posible tener al menos un poco de estructura. Los servicios (llamadas al sistema) que presta el sistema operativo se solicitan colocando los parámetros en un lugar bien definido (la pila), y ejecutando después una instrucción TRAP. Esta instrucción cambia la máquina de modo de usuario a modo de *kernel* y transfiere el control al sistema operativo, lo cual se muestra como paso 6 en la figura 1-17. Entonces, el sistema operativo obtiene los parámetros y determina cuál llamada al sistema debe ejecutarse. Después usa el número de llamada k como índice para buscar en una tabla un apuntador al procedimiento que la realiza (paso 7 en la figura 1-17).

Esta organización sugiere una estructura básica para el sistema operativo:

1. Un programa principal que invoca el procedimiento de servicio solicitado.

2. Un conjunto de procedimientos de servicio que ejecutan las llamadas al sistema.
3. Un conjunto de procedimientos utilitarios que apoyan a los procedimientos de servicio.

En este modelo, por cada llamada al sistema hay un procedimiento de servicio que se encarga de ella. Los procedimientos utilitarios hacen cosas que varios procedimientos de servicio necesitan, como obtener datos de los programas de usuario. Esta división de los procedimientos en tres capas se muestra en la figura 1-24.

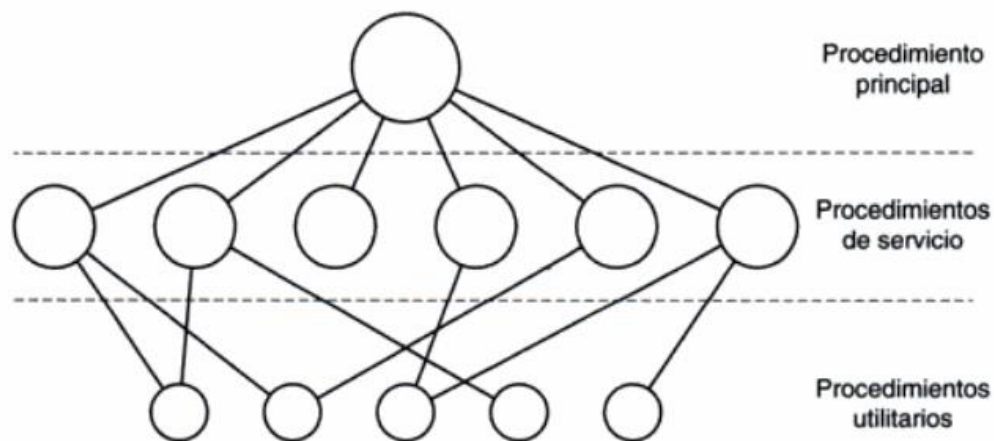


Figura 1-24. Modelo de estructuración simple para un sistema monolítico.

1.7.2 Sistemas en capas

Una generalización del enfoque de la figura 1-24 consiste en organizar el sistema operativo en una jerarquía de capas, cada una cimentada en la que está abajo. El primer sistema construido de esta manera fue THE, creado en la Technische Hogeschool Eindhoven en los Países Bajos por E. W. Dijkstra (1968) y sus estudiantes. El sistema THE era un sencillo sistema por lotes para una computadora holandesa, la Electrologica X8, que tenía 32K de palabras de 27 bits (los bits eran costosos en ese entonces).

El sistema tenía seis capas, como se muestra en la figura 1-25. La capa 0 se ocupaba de la asignación del procesador, conmutando entre procesos al presentarse interrupciones o expirar temporizadores. Arriba de la capa 0, el sistema consistía en procesos secuenciales, cada uno de los cuales podía programarse sin tener que preocuparse por el hecho de que varios procesos se estuvieran ejecutando en un solo procesador. En otras palabras, la capa 0 hacía posible la multiprogramación básica de la CPU.

| Capa | Función |
|------|--|
| 5 | El operador |
| 4 | Programas de usuario |
| 3 | Administración de entrada/salida |
| 2 | Comunicación operador-proceso |
| 1 | Administración de memoria y tambor |
| 0 | Asignación de procesador y multiprogramación |

Figura 1-25. Estructura del sistema operativo THE.

La capa 1 se encargaba de la administración de memoria: repartía espacio para los procesos en la memoria principal y en un tambor de palabras de 512K en el que se guardaban las partes de los procesos (páginas) que no cabían en la memoria principal. Arriba de la capa 1, los procesos no tenían que preocuparse por saber si estaban en la memoria o en el tambor; el software de la capa 1 se encargaba de que las páginas se transfirieran a la memoria si se les necesitaba.

La capa 2 manejaba la comunicación entre cada proceso y la consola del operador. Arriba de esta capa era como si cada proceso tuviera su propia consola de operador. La capa 3 se encargaba de administrar los dispositivos de E/S y de colocar en búferes los flujos de información hacia y desde ellos. Arriba de la capa 3 cada proceso podía tratar con dispositivos de E/S abstractos con propiedades ligeras, en lugar de dispositivos reales con muchas peculiaridades. En la capa 4 estaban los programas de usuario, que no tenían que preocuparse por la administración de procesos, memoria, consola o E/S. El proceso del operador del sistema se ubicaba en la capa 5.

Una generalización adicional del concepto de capas sucedió en el sistema MULTICS. En lugar de capas, el sistema MULTICS tenía una serie de anillos concéntricos, siendo los interiores más privilegiados que los exteriores (lo cual es efectivamente lo mismo). Cuando un procedimiento de un anillo exterior quería invocar a uno de un anillo interior, tenía que emitir el equivalente de una llamada al sistema, es decir, una instrucción TRAP cuyos parámetros se verificaban cuidadosamente para comprobar que fueran válidos, antes de permitir que se efectuara la llamada. Aunque en MULTICS todo el sistema operativo formaba parte del espacio de direcciones de cada proceso de usuario, el hardware permitía designar procedimientos individuales (en realidad, segmentos de memoria) como protegidos contra lectura, escritura o ejecución.

Si bien el esquema de capas de THE no era más que una ayuda para el diseño, porque en última instancia todas las partes del sistema se enlazaban en un solo programa objeto, en MULTICS el mecanismo de anillos sí estaba muy presente en el momento de la ejecución y el hardware hacía que se respetara. La ventaja del mecanismo de anillos es que puede extenderse con facilidad para estructurar los subsistemas de usuario. Por ejemplo, un profesor podría escribir un programa para probar y calificar los programas de los estudiantes y ejecutarlo en el anillo n , mientras que los programas de usuario se ejecutarían en el anillo $n + 1$ para que no pudieran alterar sus calificaciones.

Fuente:

Tanenbaum, A. (2003). *Sistemas Operativos Modernos*. México: Pearson Educación. pp. 56-59.