1. INTRODUCCIÓN A LOS SISTEMAS OPERATIVOS

1.1 CONCEPTO Y FUNCIONES DEL SISTEMA OPERATIVO

En los años sesenta, un sistema operativo se podría haber definido como el software que controla al hardware. Sin embargo, actualmente existe una tendencia significativa a la transferencia de las funciones del software al firmware, es decir, microcódigo. Dicha tendencia se ha pronunciado tanto que es probable que en algunos sistemas las funciones codificadas en firmware sobrepasen pronto a aquéllas codificadas en software.

Es evidente que se necesita una nueva definición de sistema operativo. Se puede imaginar un sistema operativo como los programas, instalados en el software o el firmware, que hacen utilizable el hardware.

Un sistema operativo es un conjunto de programas que administran eficientemente los recursos de un sistema de cómputo, controlando la ejecución de programas de aplicación y actuando como una interfaz entre el usuario y la computadora (el hardware). Puede considerarse que un sistema operativo lleva a cabo tres funciones principales:

?? ***Comodidad*****:** Un sistema operativo hace que una computadora sea más fácil de utilizar.
?? ***Eficiencia:*** Un sistema operativo permite que los recursos de un sistema informático se aprovechen de una manera más eficiente.
?? ***Capacidad de evolución:*** Un sistema operativo deberá construirse de modo que permita el desarrollo efectivo, la verificación y la introducción de nuevas funciones en el sistema y a la vez, no interferir en los servicios que brinda.

Los sistemas operativos son ante todo administradores de recursos; el principal recurso que administran es el hardware de la computadora: el procesador, los medios de almacenamiento, los dispositivos de E/S, los dispositivos de comunicación y los datos. Los sistemas operativos realizan muchas funciones, como proporcionar la interfaz con el usuario, permitir que los usuarios compartan entre sí el hardware y los datos, evitar que los usuarios se interfieran recíprocamente, planificar la distribución de los recursos entre los usuarios, facilitar la entrada y salida, recuperarse de los errores, contabilizar el uso de los recursos, facilitar las operaciones en paralelo, organizar los datos para lograr un acceso rápido y seguro, y manejar las comunicaciones en red.

Como sea que se decida definir los sistemas operativos, lo importante es no olvidar que constituyen una parte integral del ambiente de cómputo y, por tanto,

necesitan ser comprendidos en alguna medida por todos los usuarios de computadoras.

1.2 ESTRUCTURA DE UN SISTEMA OPERATIVO

El esquema que suele usarse para el estudio de los sistemas operativos recibe el nombre de “modelo de cebolla”, debido a que esta formado por capas concéntricas al rededor del núcleo (ver figura 1.1). La parte interna del conjunto jerárquico de programas que forman un sistema operativo recibe el nombre de núcleo o kernel. Las otras capas se encargan del manejo de la memoria, el procesador, los dispositivos de E/S y los archivos.

Figura 1.1 Un modelo de estudio para los sistemas operativos

1.3 HISTORIA Y EVOLUCIÓN DE LOS SISTEMAS OPERATIVOS

Para intentar comprender la manera de funcionar de un sistema operativo y la razón de ser de algunas de sus características principales resulta útil considerar la manera cómo han evolucionado a lo largo de los años.

Proceso en serie
En las primeras computadoras de finales de los 40 hasta mediados de los 50, el programador interactuaba directamente con el hardware; no había sistema operativo.

La operación con la máquina se efectuaba desde una consola consistente en unos indicadores luminosos, unos conmutadores, algún tipo de dispositivo de entrada y una impresora. Los programas en código máquina se cargaban a través del dispositivo de entrada (un lector de tarjetas, por ejemplo). Si se detenía el programa por un error, la condición de error se indicaba mediante los indicadores luminosos. El programador podía examinar los registros y la memoria principal para determinar la causa del error.

Si el programa continuaba hasta su culminación normal, la salida aparecía en la impresora. *Estos primeros sistemas presentaban dos problemas principales:*

Planificación: La mayoría de las instalaciones empleaban un formulario de reserva de tiempo de máquina. Normalmente un usuario podía reservar bloques de tiempo en múltiplos de media hora o algo por el estilo. Un usuario podía reservar una hora y terminar a los 45 minutos; esto daba como resultado un desperdicio del tiempo de la computadora. Por el contrario, el usuario podía tener dificultades, no terminar en el tiempo asignado y verse forzado a parar sin haber resuelto el problema.

Tiempo de preparación: Un programa sencillo, llamado trabajo, cargaba un compilador y un programa en lenguaje de alto nivel (programa fuente) en la memoria, salvaba el programa compilado (programa objeto) y luego montaba y cargaba el programa objeto junto con las funciones comunes. Cada uno de estos pasos podía implicar montar y desmontar cintas o preparar paquetes de tarjetas. Si se producía un error, el infortunado usuario tenía que volver al inicio de este proceso de preparación. De este modo, se perdía un tiempo considerable en preparar un programa para su ejecución.

Este modo de operación podría denominarse proceso en serie porque refleja el hecho de que los usuarios tenían que acceder al computador en serie. Con el paso del tiempo se desarrollaron varias herramientas de software de sistemas para intentar hacer más eficiente este proceso en serie. Entre éstas se incluían

bibliotecas de funciones comunes, montadores, cargadores y rutinas de manejo de E/S que estaban disponibles como un software común para todos los usuarios.

Sistemas simples de procesos por lotes

Las primeras máquinas eran muy caras y, por tanto, era importante maximizar la utilización de las mismas. El tiempo desperdiciado por la planificación y la preparación era inaceptable.

Para mejorar el uso, se desarrolló el concepto de sistema operativo por lotes (batch). El primer sistema operativo por lotes fue desarrollado a mediados de los 50 por la General Motors para usar en un IBM 701. Este concepto fue refinado posteriormente e implementado en un IBM 704 por una serie de clientes de IBM. A principios de los 60, un conjunto de constructores ya habían desarrollado sistemas operativos por lotes para sus computadores. IBSYS, el sistema operativo de IBM para los computadores 7090/7094, es particularmente notable por su amplia influencia en otros sistemas.

La idea central que está detrás del esquema sencillo de proceso por lotes es el uso de un elemento de software conocido como monitor. Con el uso de esta clase de sistema operativo, los usuarios ya no tenían acceso directo a la máquina. En su lugar, el usuario debía entregar los trabajos en tarjetas o en cinta al operador de la computadora, quien agrupaba secuencial mente los trabajos por lotes y ubicaba los lotes enteros en un dispositivo de entrada para su empleo por parte del monitor. Cada programa se construía de modo tal que volviera al monitor al terminar su procesamiento y, en ese momento, el monitor comenzaba a cargar automáticamente el siguiente programa.

Para entender como funciona este esquema, se va a ver desde dos puntos de vista: el del monitor y el del procesador. Desde el punto de vistas del monitor, él es quien controla la secuencia de sucesos. Para que esto sea posible, gran parte del monitor debe estar en memoria principal y disponible para su ejecución figura 1.2. Esta parte del monitor se conoce como monitor residente. El resto del monitor consta de utilidades y funciones comunes que se cargan como subrutinas en los programas de los usuarios al comienzo de cualquier trabajo que las necesite. El monitor lee los trabajos uno a uno del dispositivo de entrada (normalmente, un lector de tarjetas o una unidad de cinta magnética). A medida que lo lee, el trabajo actual se ubica en la zona del programa de usuario y el control pasa al trabajo. Cuando el trabajo termina, se devuelve el control al monitor, quien lee inmediatamente un nuevo trabajo. Los resultados de cada trabajo se imprimen y entregan al usuario.

Figura 1.2 Disposición de la memoria con un monitor residente

Considere esta secuencia desde el punto de vista del procesador. En un cierto momento, el procesador estará ejecutando instrucciones de la zona de memoria principal que contiene al monitor. Estas instrucciones hacen que el trabajo siguiente sea leído en otra zona de la memoria principal. Una vez que el trabajo se ha leído, el procesador encuentra en el monitor una instrucción de desvío que ordena al procesador continuar la ejecución en el inicio del programa de usuario. El procesador ejecuta entonces las instrucciones del programa de usuario hasta que encuentre una condición de finalización de error. Cualquiera de estos dos sucesos provocan que el procesador vaya por la instrucción siguiente del programa monitor. De este modo, la frase “el control se le pasa al trabajo” quiere decir simplemente que el procesador pasa a leer y a ejecutar instrucciones del programa de usuario, que la frase “el control vuelve al monitor” quiere decir que el procesador pasa ahora a leer y ejecutar las instrucciones del programa monitor.

Debe quedar claro que es el monitor el que gestiona el problema de la planificación. Se pone en cola un lote de trabajos y éstos son ejecutados tan rápido como es posible, sin que haya tiempo alguno de desocupación.

¿Qué ocurre con la preparación de los trabajos? El monitor también se encarga de esto. Con cada trabajo, se incluyen instrucciones de una forma primitiva de lenguaje control de trabajos (JCL, Job Control Language), que es un tipo especial de lenguaje de programación empleado para dar instrucciones al monitor. La figura 1.3 muestra un ejemplo sencillo con entradas de trabajos desde tarjetas. En este ejemplo, el usuario envía un programa escrito en FORTRAN junto a unos datos que se utilizarán en el programa. Además de las tarjetas de FORTRAN y de datos, el paquete incluye instrucciones de control de trabajos, que se denotan mediante un signo dólar ($) al comienzo.

Para ejecutar el trabajo, el monitor lee la tarjeta $FTN y carga el compilador adecuado desde el dispositivo de almacenamiento masivo (generalmente una cinta). El compilador traduce el programa de usuario en código objeto, que se almacena en memoria o en el dispositivo de almacenamiento. Si se carga en memoria, la operación es conocida como “compilar, cargar y arrancar” (compile, load, and go). Si se almacena en cinta, entonces se requiere la tarjeta &LOAD. Esta tarjeta es leída por el monitor, quien retoma el control después de la operación de compilación. El monitor llama al cargador, que carga el programa objeto en memoria en lugar del compilador y le transfiere el control. De esta manera, un segmento grande de memoria se puede compartir entre diferentes subsistemas, aunque en cada momento sólo uno de ellos tiene que estar presente y ejecutándose.

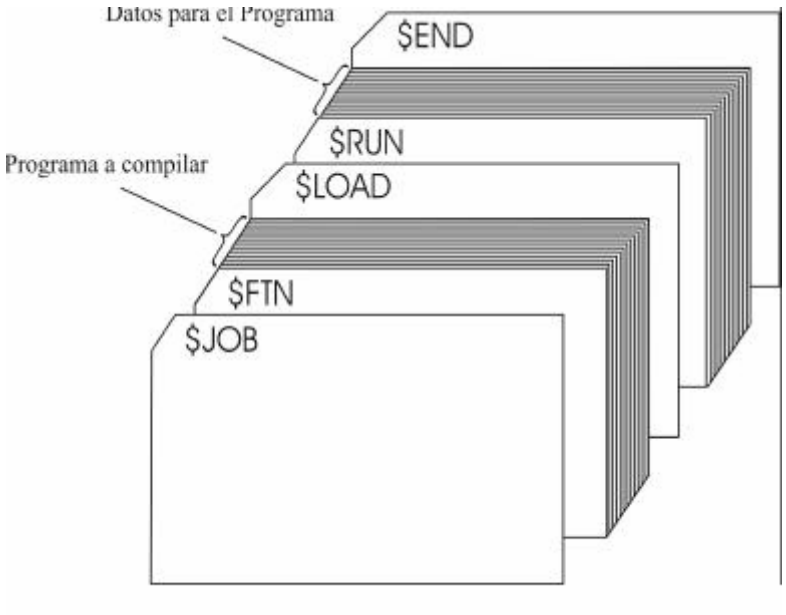

Figura 1.3 Paquete de tarjetas para un sistema sencillo por lotes

Durante la ejecución del programa de usuario, cada instrucción de entrada origina la lectura de una tarjeta de datos. La instrucción de entrada en el programa de usuario hace que se invoque una rutina de entrada, que forma parte del sistema operativo. La rutina de entrada se asegura de que el programa de usuario no ha leído accidentalmente una tarjeta JCL. Si esto sucede, se produce un error y el control se transfiere al monitor. Al terminar un trabajo, con o sin éxito, el monitor recorre las tarjetas de entrada hasta encontrar la próxima tarjeta JCL. De este modo, el sistema se protege contra un programa que tenga tarjetas de datos de más o de menos.

Se comprobará que el monitor o el sistema de proceso por lotes es simplemente un programa de computadora. Se basa en la capacidad del procesador para traer y ejecutar instrucciones desde varias zonas de la memoria principal y así apoderarse y ceder el control de forma alterna. Para esto serían convenientes algunas otras características del hardware, entre las que se encuentran las siguientes:

?? *Protección de memoria:* Mientras el programa de usuario este ejecutándose, no debe modificar la zona de memoria en la que está el monitor. Si se hace un intento tal, el hardware del procesador deberá detectar el error y transferir el control al monitor. El monitor abortara entonces el trabajo, imprimirá el mensaje de error y cargará el siguiente trabajo.

?? *Temporizador:* Se utiliza un temporizador para impedir que un solo trabajo monopolice el sistema. El temporizador se lanza al comenzar cada trabajo. Si expira el tiempo, se producirá una interrupción y el control volverá al monitor.

?? *Instrucciones Privilegiadas:* Ciertas instrucciones son consideradas como privilegiadas y pueden ser ejecutadas sólo por el monitor. Si el procesador encuentra una instrucción tal, cuando está ejecutando el programa del usuario, se producirá una interrupción de error. Entre las instrucciones privilegiadas se encuentran las instrucciones de E/S, de forma que el monitor retenga el control de todos los dispositivos de E/S. Esto impide por ejemplo, que un programa de usuario lea accidentalmente instrucciones de control que son del trabajo siguiente.

Si un programa de usuario desea realizar una E/S, debe solicitar al monitor que haga la operación por él. Si el procesador encuentra una instrucción privilegiada cuando esta ejecutando un programa de usuario, el hardware del procesador la considera como un error y transfiere el control al monitor.

?? *Interrupciones:* los primeros modelos de computadores no tenían esta capacidad.

Esta característica aporta al sistema operativo más flexibilidad para ceder y retomar el control de los programas usuarios.

Naturalmente se puede construir un sistema operativo sin esta característica, pero los fabricantes de computadores comprobaron rápidamente que los resultados eran caóticos y, por tanto, incluso los sistemas operativos por lotes más primitivos ya disponían de esta características en el hardware. Por otro lado, hay que decir que el sistema operativo más utilizado del mundo, el MS-DOS, no dispone de protección de memoria ni de instrucciones privilegiadas de E/S. Sin embargo, como este sistema está destinado a computadoras personales de un solo usuario, los problemas que se puedan originar son menos graves.

En un sistema operativo por lotes, el tiempo de máquina se reparte entre la ejecución de programas de usuarios y la ejecución del monitor. Así se tienen dos pérdidas: se entrega al monitor cierta cantidad de memoria principal y éste consume cierto tiempo de la máquina. Ambas pérdidas son una forma de sobrecarga. Aún con esta sobrecarga, los sistemas operativos sencillos por lotes mejoran el uso de la computadora.

Sistemas por lotes multiprogramados

Aún con el secuenciamiento automático de los trabajos ofrecido por un sistema operativo sencillo por lotes, el procesador esta desocupado a menudo. El problema es que los dispositivos de E/S son lentos comparados con el procesador. La figura 1.4 detalla un cálculo representativo. Los números corresponden a un programa que procesa un archivo de registros y ejecuta, en promedio, 100 instrucciones de máquina por cada registro. En este ejemplo, la computadora gasta más del 96% del tiempo esperado a que los dispositivos de E/S terminen de transferir sus datos. La figura 1.5ª ilustra esta situación. El procesador gasta parte del tiempo ejecutando hasta que encuentra una instrucción de E/S. Entonces debe esperar a que concluya la instrucción de E/S antes de continuar.

Leer un registro 0,0015 segundos
Ejecutar 100 instrucciones 0,0001 segundos
Escribir un registro 0,0015 segundos
TOTAL 0,0031 segundos
Porcentaje de utilización de la CPU = 0,0001/0,0031 = 0,032 = 3,2%

Figura 1.4 Ejemplo de utilización del sistema

Esta ineficiencia no es necesaria. Se sabe que hay memoria suficiente para almacenar el sistema operativo (el monitor residente) y un programa de usuario.

Supóngase que hay espacio suficiente para el sistema operativo y dos programas de usuario. Ahora, cuando un trabajo necesite esperar una E/S, el procesador puede cambiar al otro trabajo, que probablemente no estará esperando a la E/S (figura 1.5b).

Además, se podría ampliar la memoria para almacenar tres, cuatro o más programas y conmutar entre todos ellos (figura 1.5c). Este proceso es conocido como multiprogramación multitarea. Este es el punto central de los sistemas operativos modernos.

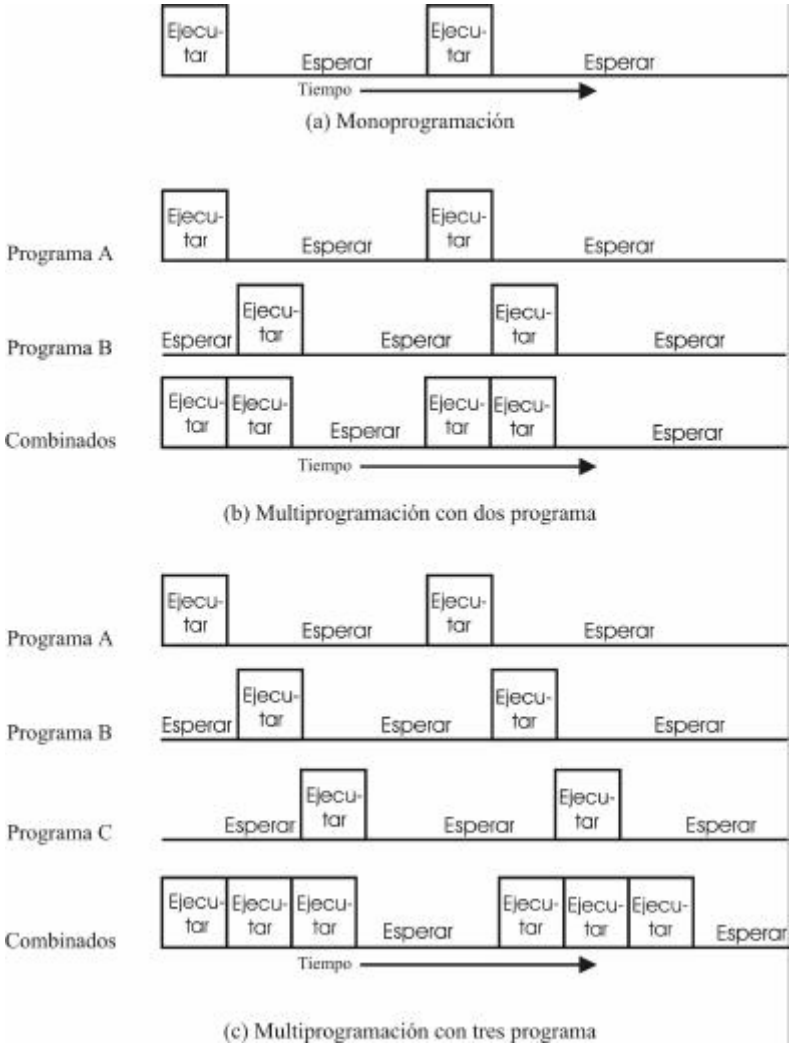

Figura 1.5 Multiprogramación

Sistemas de tiempo compartido

Con el uso de multiprogramación, el tratamiento por lotes puede llegar a ser bastante eficiente. Sin embargo, para muchas tareas, es conveniente suministrar un modo en que el usuario interactúe con la computadora. De hecho, para algunos trabajos, tales como el proceso de transacciones, este modo interactivo es fundamental.

Hoy en día, los requisitos de un servicio de computación interactiva pueden y suelen llevarse a cabo con el empleo de una computadora dedicada. Esta opción no estaba disponible en los años 60, cuando la mayoría de los computadores eran grandes y costosas. En su lugar, se desarrollaron las técnicas de tiempo compartido.

Al igual que la multiprogramación permite al procesador manejar varias tareas por lotes al mismo tiempo, la multiprogramación puede también utilizarse para manejar varias tareas interactivas. En este último caso, la técnica se conoce como tiempo compartido, por que refleja el hecho de que el tiempo del procesador es compartido entre los diversos usuarios. La técnica básica de un sistema de tiempo compartido es tener varios usuarios utilizando simultáneamente el sistema mediante terminales, mediante que el sistema operativo intercala la ejecución de cada programa de usuario en ráfagas cortas de cómputo o *quantum*. De esta manera si hay N usuarios que solicitan servicio a la vez, cada usuario solo dispondrá, en medio, de 1/N de la atención efectiva de la computadora, sin contar con la sobrecarga del sistema operativo. Sin embargo, dado el tiempo de reacción relativamente lento que tiene el ser humano, el tiempo de respuesta en un sistema correctamente diseñado debería ser comparable al de una computadora dedicada.

Tanto la multiprogramación por lotes como el tiempo compartido utilizan multiprogramación. Las diferencias básicas se enumeran en la tabla 1.1. Multiprogramación por lotes Tiempo compartido

Objetivo principal
Origen de la instrucción
Maximizar la utilización del procesador
Instrucciones de un lenguaje de control de trabajos incluidas junto con el trabajo
Minimizar el tiempo de Respuesta
Ordenes dadas en el terminal

Tabla 1.1 Multiprogramación por lotes frente a tiempo compartido

1.4 HARDWARE, SOFTWARE Y FIRMWARE

HARDWARE

El *hardware* consiste en los dispositivos de un sistema de cómputo: su procesador, sus almacenamientos, sus dispositivos de entrada/salida y sus conexiones de comunicación.

SOFTWARE

El *software* se compone de los programas con instrucciones en lenguaje de máquina y los datos que son interpretados por el hardware. Algunos tipos comunes de software son los compiladores, ensambladores, cargadores, editores de enlace, cargadores de enlace, programas de aplicación, sistemas manejadores de base de datos, sistemas de comunicación de datos y sistemas operativos.

El *software* esta constituido por los programas de instrucciones y los datos que definen para el hardware los algoritmos de resolución de problemas. Los sistemas operativos y otro software de sistemas son proporcionados generalmente por los proveedores de hardware. La gran mayoría del software de aplicación es proporcionada por *proveedores independientes* de *software.*

FIRMWARE

El *firmware* consiste en programas ejecutables almacenados en un circuito integrado programable de sólo lectura (PROM). Instrucción o programa informativo (tal como una microprograma) de uso frecuente que se encuentra almacenado en la ROM, en lugar de estar incluido en el software básico. Se utiliza a menudo en computadoras que controlan procesos de producción.

Firmware, en general, software de sistema guardado de manera permanente en la memoria de solo lectura (ROM) o en otra parte de la circuiteria de la computadora, como en los chips del sistema básico de entrada-salida (BIOS) de las computadoras compatibles con la PC de IBM.