**Capitulo 2**

**FUNCIONES Y OBJETIVOS DE LOS SISTEMAS OPERATIVOS**

un sistema operativo tiene tres objetivos o lleva a cabo tres funciones:

*• Comodidad:* Un sistema operativo hace que un computador sea más cómoda 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 debe construirse de modo que permita el desarrollo efectivo, la verificación y la introducción de nuevas funciones en el sistema

**El Sistema Operativo como Interfaz Usuario/Computadora**

El hardware y el software que se utilizan para proveer de aplicaciones a los usuarios pueden contemplarse de forma estratificada o jerárquica. Al usuario de estas aplicaciones se le llama *usuario*. el usuario final ve al sistema informático en términos de aplicaciones. Las aplicaciones pueden construirse con un lenguaje de programación y son desarrolladas por programadores de aplicaciones. Algunos de estos programas se denominan *utilidades* e implementan funciones que ayudan a la creación de los programas, la gestión de los archivos y el control de los dispositivos de E/S. El sistema operativo oculta al programador los detalles del hardware y le proporciona una interfaz cómoda para utilizar el sistema. Actúa como mediador, facilitándole al programador y a los programas de aplicación el acceso y uso de todas esas características y servicios.

un sistema operativo ofrece servicios en las áreas siguientes:

• *Creación de programas:* El sistema operativo ofrece una variedad de características y servicios, tales como los editores y los depuradores *(debuggers),* para ayudar al programador en la creación de programas.

• *Ejecución de programas:* Para ejecutar un programa se necesita un cierto número de tareas. Las instrucciones y los datos se deben cargar en la memoria principal, los archivos y los dispositivos de E/S se deben inicializar y se deben preparar otros recursos.

• *Acceso a los dispositivos de E/S:* Cada dispositivo de E/S requiere un conjunto propio y peculiar de instrucciones o de señales de control para su funcionamiento.

• *Acceso controlado a los archivos:* el control debe incluir una comprensión, del formato de los archivos y del medio de almacenamiento. Una vez más, es el sistema operativo el que se encarga de los detalles.

• *Acceso al sistema:* un sistema compartido o público, el sistema operativo controla el acceso al sistema como un todo y a los recursos específicos del sistema.

• *Detección y respuesta a errores:* el sistema operativo debe dar una respuesta que elimine la condición de error con el menor impacto posible sobre las aplicaciones que están en ejecución. La respuesta puede ser desde terminar el programa que produjo el error, hasta reintentar la operación o, simplemente, informar del error a la aplicación.

• *Contabilidad:* Un buen sistema operativo debe recoger estadísticas de utilización de los diversos recursos y supervisar los parámetros como el tiempo de respuesta. Para cualquier sistema, esta información es útil para anticiparse a la necesidad de mejoras futuras y así mejorar su rendimiento

.

**El sistema operativo como administrador de recursos**

Un computador es un conjunto de recursos para el traslado, almacenamiento y proceso de datos y para el control de estas funciones. El sistema operativo es el responsable de la gestión de estos recursos.

Este no es el caso de un sistema operativo, que no es habitual como mecanismo de control en dos aspectos:

• El sistema operativo funciona de la misma manera que el software normal de un computador, es decir, es un programa ejecutado por el procesador.

• El sistema operativo abandona con frecuencia el control y debe depender del procesador para recuperarlo.

El sistema operativo es nada más que un programa del computador. El sistema operativo dirige al procesador en el empleo de otros recursos del sistema y en el control del tiempo de ejecución de otros programas. el sistema operativo cede el control al procesador para hacer algún trabajo "útil" y luego lo retoma durante el tiempo suficiente para preparar el procesador para llevar a cabo la siguiente parte del trabajo.

El sistema operativo decide cuándo puede utilizarse un dispositivo de E/S por parte de un programa en ejecución y controla el acceso y la utilización de los archivos. El procesador es, en si mismo, un recurso y es el sistema operativo el que debe determinar cuánto tiempo del procesador debe dedicarse a la ejecución de un programa de usuario en particular. En el caso de sistemas multiprocesador, la decisión debe distribuirse entre todos los procesadores.

**Facilidad de evolución de un sistema operativo**

Un sistema operativo importante evolucionará en el tiempo por una serie de razones:

*• Actualizaciones del hardware* y *nuevos tipos de hardware:* el empleo de terminales gráficos y terminales de pantalla completa, en lugar de los terminales de líneas, pueden influir en el diseño de los sistemas operativos. Esto necesita un soporte más sofisticado en el sistema operativo.

*• Nuevos servicios:* Como respuesta a Las demandas del usuario o a las necesidades de los ad-ministradores del sistema, el sistema operativo ampliará su oferta de servicios. Por ejemplo, si se determina que es difícil de mantener un buen rendimiento para los usuarios con las he-rramientas existentes, se deben añadir nuevas medidas y herramientas de control al sistema operativo. Otro ejemplo es el de las nuevas aplicaciones que exigen el uso de ventanas en la pantalla. Esta característica requiere actualizaciones mayores en el sistema operativo.

*• Correcciones:* el sistema operativo tiene fallos que se descubrirán con el curso del tiempo y que es necesario corregir.

La necesidad de hacer cambios en un sistema operativo de forma regular introduce ciertos requisitos en el diseño. Una afirmación obvia es que el sistema debe tener una construcción modular, con interfaces bien definidas entre los módulos y debe estar bien documentado. Para programas grandes, como normalmente son los sistemas operativos actuales, no es adecuado lo que podría denominarse modularización elemental

**EVOLUCION DE LOS SISTEMAS OPERATIVOS**

**Proceso en serie**

En los primeros computadores, el programador interactuaba directamente con el hardware; no había sistema operativo. Los programas en código máquina se cargaban a través del dispositivo de entrada. Si se detiene el programa por un error, la condición de error se indicaba mediante los indicadores luminosos.

Estos primeros sistemas presentaban dos problemas principales:

• *Planificación:* Un usuario podía reservar una hora y terminar a los 45 minutos; esto daba como resultado un desperdicio del tiempo del computador. Por el contrario, el usuario podía tener dificultades, no terminar en el tiempo asignado y verse forzado a parar sin haber solucionado el problema.

• *Tiempo de preparación:* esto implica 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.

**Sistemas sencillos de proceso por lotes**

Las primeras máquinas eran muy caras El tiempo desperdiciado por la planificación y la preparación era inaceptable. se desarrolló el concepto de sistema operativo por lotes *(batch).*

La idea central que está detrás del esquema sencillo de proceso por lotes es el uso de un elemento de software conocido como monitor el usuario debía entregar los trabajos en tarjetas o en cinta al operador del computador, Cada programa se construía de modo tal que volviera al monitor al terminar su procesamiento

Desde el punto de vista del monitor, él es quien controla la se-cuencia de sucesos. gran parte del monitor debe estar siempre en memoria principal y disponible para su ejecución se conoce como monitor residente.

ahora 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. 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 una condición de finalización o de error.

es el monitor el que gestiona el problema de la planificación con cada trabajo, se incluyen instrucciones de una forma primitiva de lenguaje de control de trabajos (JCL, *Job Control Lenguaje),* que es un tipo especial de lenguaje de programación empleado para dar instrucciones al monitor.

Para ejecutar el trabajo, el monitor lee la tarjeta $FTN y carga el compilador adecuado desde el dispositivo de almacenamiento masivo 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 el lugar del compilador y le transfiere el control.

cada instrucción de entrada origina la lectura de una tarjeta de datos. La instrucción de entrada en el programa del usuario hace que se invoque una rutina de entrada, La rutina de entrada se asegura de que el programa de usuario no ha leído accidentalmente una tarjeta JCL. el monitor recorre las tarjetas de entrada hasta encontrar la próxima tarjeta JCL. el monitor o el sistema de proceso por lotes es simplemente un programa de computador. Se basa en la capacidad del procesador para traer y ejecutar instrucciones desde varias zonas de la memoria principal otras características del hardware, entre las que se encuentran las siguientes:

*• Protección de memoria:* Mientras el programa de usuario esté ejecutándose, no debe mo-dificar la zona de memoria en la que está el monitor.

*• Temporizador:* Se utiliza para impedir que un sólo trabajo monopolice el sistema.

*• Instrucciones Privilegiadas:* Ciertas instrucciones son designadas como privilegiadas y pueden ser ejecutadas solo por el monitor. Si el procesador encuentra una instrucción tal, se producirá una interrupción de error. instrucciones privilegiadas se encuentran las instrucciones de E/S, Si un programa de usuario desea realizar una *E/S,* debe solicitarse al monitor que haga la operación por él.

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

sin estas características los fabricantes de computadores comprobaron rápidamente que los resultados eran caóticos los sistemas operativos por lotes más primitivos ya disponían de estas ca-racterísticas en el hardware.

En un sistema operativo por lotes, el tiempo de máquina se reparte entre la ejecución de programas de usuario 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.

**Sistemas por lotes con multiprogramación**

sistema operativo sencillo por lotes, el procesador está desocupado a menudo, el computador gasta más tiempo esperando a que los dispositivos de E/S terminen de transferir sus datos

Supóngase que hay espacio suficiente para el sistema operativo y dos programas usuarios cuando un trabajo necesite esperar una E/S, el procesador puede cambiar al otro trabajo, Además, se podría ampliar la memoria para almacenar tres, cuatro o más programas y conmutar entre todos ellos Este proceso es conocido como multiprogramador o multitarea. Éste es el punto central de los sistemas operativos modernos.

un sistema por lotes con multiprogramación tiene que depender de ciertas características del hardware del computador. La característica adicional es que el hardware respalde las interrupciones de E/S y el DMAel procesador puede enviar una orden de E/S para un trabajo y continuar con la ejecución de otro, mientras la E/S es efectuada por el controlador del dispositivo.

Los sistemas operativos con multiprogramación son bastante más sofisticados en comparación con los sistemas de monoprogramación o de un solo programa. Para tener varios trabajos listos para ejecutar, éstos deben mantenerse en la memoria principal.

**Sistemas de tiempo compartido**

Con el uso de la 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 directamente con el computador. De hecho, para algunos trabajos, tales como el proceso de transacciones, este modo interactivo es fundamental.

Esta opción no estaba disponible en los años 60, cuando la mayoría de los computadores eran grandes y costosas. se desarrollaron la técnica se conoce como tiempo compartido, porque 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 a varios usuarios utilizando simultáneamente el sistema mediante terminales, mientras que el sistema operativo intercala la ejecución de cada programa de usuario en ráfagas cortas de cómputo o cuantos *(quantum).*

Tanto la multiprogramación por lotes como el tiempo compartido utilizan multiprogramación.

El tiempo compartido y la multiprogramación plantean una multitud de problemas nuevos *para el sistema operativo. Si hay varios trabajos en memoria, entonces* el sistema de archivos debe protegerse de forma que sólo los usuarios autorizados puedan tener acceso a un archivo en particular.

**LOGROS PRINCIPALES**

se han obtenido cuatro logros intelectuales significativos en el desarrollo de los sistemas operativos:

• Los procesos

• La gestión de memoria

• La seguridad y la protección de la información

• La planificación y la gestión de recursos

• La estructura del sistema

**Procesos**

*es* fundamental en la estructura de los sistemas operativos. Este término fue acuñado por primera vez por los diseñadores de Multics en los años 60. Es un término algo más general que el de *trabajo.*

• Un programa en ejecución

• El "espíritu animado" de un programa

• La entidad que puede ser asignada al procesador y ejecutada por él.

.

Tres líneas principales en el desarrollo de los sistemas informáticos la operación por lotes con multiprogramación, el tiempo compartido y los sistemas de transacciones en tiempo real. Como se ha visto, la multiprogramación fue diseñada para mantener ocupados a la vez tanto procesador como los dispositivos de E/S, incluyendo los dispositivos de almacenamiento, de modo que se alcance la mayor eficiencia posible. La clave de este mecanismo es que, como respuesta a las señales que indiquen que ha terminado una transacción de E/S los sistemas de tiempo compartido de propósito general el objetivo clave del diseño es que el sistema sea sensible a las necesidades del usuario individual y que, además, por razones de coste, pueda dar soporte simultáneo a muchos usuarios.

transacciones en tiempo real, La diferencia clave entre un sistema de proceso de transacciones y un sistema de tiempo compartido es que el primero está limitado a una o pocas aplicaciones, mientras que los usuarios de un sistema de tiempo compartido pueden dedicarse al desarrollo de un programa, En ambos casos, el tiempo de respuesta del sistema es primordial.

La herramienta principal disponible para los programadores de sistemas en el desarrollo de los primeros sistemas interactivos multiusuario y de multiprogramación fue la interrupción.

El diseño del software del sistema para coordinar estas diversas actividades resultó extraordinariamente difícil.

En ausencia de un medio sistemático de coordinación y cooperación entre las actividades, los programadores recurrían a métodos *ad hoc* basados en su comprensión del entorno que el sistema operativo tenía que controlar. Estos esfuerzos estaban expuestos a errores sutiles de programación Estos eran muy difíciles de diagnosticar, porque era necesario poder distinguirlos de los errores del software de aplicación y de los del hardware. había cuatro causas principales de error:

• *Sincronización incorrecta:* Es frecuente el caso en el que una rutina debe ser suspendida a la espera de un suceso en cualquier lugar del sistema. Un diseño incorrecto del mecanismo de señalización puede dar como resultado la pérdida de señales o la recepción de señales duplicadas.

• *Fallos de exclusión mutua:* Es habitual el caso en que más de un usuario o programa in-tentan a la vez hacer uso de un recurso compartido.Debe existir algún tipo de mecanismo de exclusión mutua que permita que sólo una rutina a la vez pueda realizar una transacción sobre una determinada parte de los datos. Verificar la corrección de la implementación de dicha exclusión mutua bajo todas las secuencias posibles de sucesos es difícil.

• *Funcionamiento no determinista del programa:* Los resultados de un programa en particular deben depender sólo de la entrada del programa y no de las actividades de otros programas en un sistema compartido. el orden en que se organiza la ejecución de varios programas puede influir en los resultados de un programa en particular.

• *Interbloqueos:* Es posible que dos o más programas estén suspendidos a la espera uno del otro,puede depender del ritmo imprevisto de la asignación y la liberación de recursos.

Se puede considerar que un proceso está formado por las tres componentes siguientes:

• Un programa ejecutable

• Los datos asociados necesarios para el programa (variables, espacio de trabajo, buffers, etc.)

• El contexto de ejecución del programa

El contexto de ejecución incluye toda la información que el sistema operativo necesita para administrar el proceso y que el procesador necesita para ejecutar correctamente el proceso.

La lista de procesos contiene una entrada para cada proceso, la cual dispone de un puntero a la posición del bloque de memoria que contiene al proceso. El registro de índice del proceso contiene el índice, dentro de la lista de procesos, del proceso que está actualmente controlando al procesador. El contador de programa apunta a la próxima instrucción del proceso que se ejecutará. Los registros de base y de límite definen la región de memoria ocupada por el proceso y no deben exceder el valor del registro de límite. Esto impide las interferencias entre procesos.

El "estado" entero del proceso está contenido en su contexto. Esta estructura permite el desarrollo de técnicas potentes que aseguran la coordinación y la cooperación entre procesos. Se pueden diseñar e incorporar nuevas características al sistema operativo mediante la ampliación del contexto para incluir cualquier nueva información que sea necesaria para dar soporte al nuevo atributo.

Gestión de memoria

el sistema operativo tiene cinco responsabilidades principales en la gestión del almacenamiento que son:

• *Aislamiento del proceso:* El sistema operativo debe procurar que cada proceso independiente no interfiera con los datos y la memoria de ningún otro.

• *Asignación y gestión automáticas:* A los programas se les debe asignar memoria dinámicamente en la jerarquía de memoria, según la vayan necesitando. Este proceso debe ser transparente para el programador.

*• Soporte para la programación modular:* Los programadores deben ser capaces de definir módulos de programa y de crear, destruir y alterar el tamaño de los módulos dinámicamente.

*• Protección y control de acceso:* Compartir la memoria en algún nivel de la jerarquía de memoria origina la posibilidad de que un programa pueda direccionar el espacio de memoria de otro programa. Algunas veces, esto es conveniente, sobre todo cuando se necesita compartición en una aplicación en particular. En otros casos, esto amenaza la integridad de los programas y del mismo sistema operativo. El sistema operativo debe permitir que las secciones de memoria estén accesibles de varias maneras para los diversos usuarios.

*• Almacenamiento a largo plaza:* Muchos usuarios y aplicaciones necesitan medios para almacenar información por largos periodos de tiempo.

La *memoria virtual* es un servicio que permite a los programas direccionar la memoria desde un punto de vista lógico, sin depender del tamaño de la memoria principal física disponible. Cuando se está ejecutando, solo una parte del programa y de los datos pueden estar realmente en memoria principal. Las partes restantes del programa y de los datos se mantienen en bloques en el disco. El sistema de archivos da cuenta del almacenamiento a largo plazo, almacenándose la información en unos objetos con nombre denominados *archivos.* El archivo es un concepto práctico para el programador y es una unidad útil de control de acceso y de protección en el sistema operativo.

las instrucciones del lenguaje de programación pueden hacer referencia a los programas y las posiciones de los datos en la memoria virtual. El aislamiento de los procesos se puede lograr dándole a cada proceso una única memoria virtual que no se solape con otra. Los archivos se mantienen en un almacén permanente. Los archivos o una parte de los mismos pueden copiarse en la memoria virtual para su manipulación por parte de los programas.

El almacenamiento consta de una memoria principal directamente direccionable y una memoria auxiliar, cargando los bloques en la memoria principal. Se coloca un hardware de traducción de direcciones *(mapper)* entre el procesador y la memoria. Los programas hacen referencia a las posiciones utilizando direcciones virtuales, que son traducidas a direcciones reales de la memoria principal. Si se hace una referencia a una dirección virtual que no está en la memoria real, entonces una parte del contenido de la memoria real se expulsa hacia la memoría auxiliar, intercambiándose con el bloque de memoria deseado. Es tarea del diseñador construir un mecanismo de traducción de direcciones que genere poca sobrecarga.

**Seguridad y protección de la información**

El crecimiento de la utilización de los sistemas de tiempo compartido y, más recientemente, las redes de computadores, ha traído consigo un aumento de las preocupaciones por la protección de la información.

1. Intentos organizados y deliberados de obtener información económica y mercantil de las organizaciones competitivas del sector privado.

2. Intentos organizados y deliberados de obtener información económica de las oficinas del gobierno.

3. Adquisición inadvertida de información económica o mercantil.

4. Adquisición inadvertida de información sobre las personas.

5. Fraude intencional a través del acceso ilegal a bancos de datos en computadores, con énfasis, en orden decreciente de importancia.

6. Intromisión del gobierno en los derechos individuales.

7. Atropello de los derechos individuales por la comunidad.

La naturaleza de las amenazas que conciernen a una organización pueden variar enormemente de un conjunto de circunstancias a otro. Solo interesan los problemas de control de acceso a los sistemas informáticos y a la información almacenada en ellos. Se han identificado cuatro clases de políticas generales de protección, en orden creciente de dificultad:

• *No compartición:* los procesos están aislados uno del otro y cada proceso tiene control exclusivo sobre los recursos que le fueron asignados estática o dinámicamente.

• *Compartiendo los originales de los programas o archivos de datos:* Con el uso de código reentrante, una única copia física de un programa puede aparecer en varios espacios de memoria virtual como archivos de sólo lectura.

• *Subsistemas confinados o sin memoria:* El servidor se protegerá de que el cliente descubra el algoritmo con el cual lleva a cabo su trabajo y el cliente se protegerá de que el servidor retenga alguna información sobre el trabajo que está llevando a cabo.

• *Diseminación controlada de la información:* A los usuarios y a las aplicaciones se les dan credenciales de seguridad de un cierto nivel, mientras que a los datos y a otros recursos se les dota de clasificaciones de seguridad. La política de seguridad hace cumplir las restricciones relativas a qué usuarios tienen acceso a qué clasificaciones.

• *Control de acceso:* Tiene que ver con la regulación del acceso del usuario al sistema com-pleto, a los subsistemas y a los datos, así como a regular el acceso de los procesos a los recursos y objetos del sistema.

• *Control del flujo de información:* Regula el flujo de datos dentro del sistema y su distribu-ción a los usuarios.

• *Certificación:* Es relativa a la demostración de que el acceso y los mecanismos de control del flujo se llevan a cabo de acuerdo a las especificaciones y a que estas cumplen las políticas de protección y seguridad deseadas.

**Planificación y gestión de recursos**

Una tarea clave del sistema operativo es administrar los recursos que tiene disponibles y planificar su utilización por parte de los diferentes procesos en activo. Cualquier política de asignación de recursos y de planificación debe tener en cuenta los tres factores siguientes:

• *Equidad:* los procesos que compiten por el uso de un determinado recurso les sea otorgado un acceso al recurso que sea aproximadamente igualitario y equitativo. Esto es especialmente así para los trabajos de una misma clase

• *Sensibilidades diferenciales:* El sistema operativo debe intentar tomar decisiones de asignación y planificación que satisfagan la totalidad de los requisitos. El sistema operativo debe contemplar estas decisiones dinámicamente.

• *Eficiencia:* el sistema operativo debe intentar maximizar la productividad, minimizar el tiempo de respuesta y, en el caso de tiempo compartido, alojar a tantos usuarios como sea posible.

La tarea de planificación y gestión de recursos es básicamente un problema de investigación operativa, así que se pueden aplicar los resultados matemáticos de esta disciplina.

El sistema operativo mantiene una serie de colas, cada una de las cuales no es más que una lista de procesos esperando a algún recurso. La cola a corto plazo está formada por procesos que están en memoria principal (o que, por *lo* menos, una parte mínima básica está en memoria principal) y están listos para ejecutar. Alguno de estos procesos podría ser el siguiente en usar el procesador. Depende del planificador a corto plazo o distribuidor *(dispatcher)* el escoger a uno.

Todos los procesos esperando por el uso de un determinado dispositivo están alineados en la cola de dicho dispositivo. Una vez más, es el sistema operativo el que debe determinar a qué proceso se le debe asignar un dispositivo de E/S cuando esté disponible.

. Un proceso puede invocar algún servicio del sistema operativo, como un manejador de dispositivos de E/S, una vez que la interrupción o la petición de servicio es atendida, se invoca al planificador a corto plazo para que seleccione un proceso para su ejecución.

**Estructura del sistema**

En la medida en que se añaden más características a los sistemas operativos y en que el hardware se hace más complejo y versátil, el tamaño y la complejidad de los sistemas operativos ha ido creciendo. El tamaño de un sistema operativo completo y la dificultad de las tareas que lleva a cabo plantean tres problemas desafortunados pero demasiado habituales. Primero, los sistemas operativos, cuando se entregan, ya están cronológicamente retrasados. Segundo, los sistemas tienen fallos latentes que se manifiestan en el terreno y que deben ser detectados y corregidos. Y, por último, su rendimiento no es a menudo el que se esperaba.

El software debe ser modular. Esto ayuda a organizar el proceso de desarrollo de software y reduce las tareas de diagnóstico y detección de errores. Los módulos tienen que tener interfaces bien definidas entre sí y estas interfaces deben ser tan simples como sea posible.

La estructura jerárquica de un sistema operativo moderno separa sus funciones de acuerdo a su complejidad, su escala característica de tiempo y su nivel de abstracción. Se puede contemplar al sistema como una serie de niveles. Cada nivel lleva a cabo un determinado subconjunto de funciones requeridas por el sistema operativo.

Es útil tomar un sistema propuesto por Brown y sus colegas y por Denning y Brown, aunque no corresponde a ningún sistema operativo en particular. El modelo está definido en la tabla 2.4 y consta de los siguientes niveles:

• Nivel 1: Consta de circuitos electrónicos, donde los objetos que se tratan son registros, cel-das de memoria y puertas lógicas.

• Nivel 2: Es el conjunto de instrucciones del procesador.

• Nivel 3: Añade el concepto de procedimiento o subrutina, así como las operaciones de llamada y retomo.

• Nivel 4: Introduce las interrupciones, las cuales hacen que el procesador salve el contexto actual e invoque a una rutina de tratamiento de la interrupción.

• Nivel 5: En este nivel se introduce la noción de proceso como un programa en ejecución.

• *Nivel 6:* Tiene que ver con los dispositivos de almacenamiento secundario del computador.

• *Nivel 7:* Crea un espacio de direcciones lógicas para los procesos.

• *Nivel 8:* Se dedica a la comunicación de información y mensajes entre los procesos.

• *Nivel 9:* Da soporte al almacenamiento a largo plazo de los archivos con nombre.

• *Nivel 10:* Es el que proporciona acceso a los dispositivos externos mediante interfaces es- tandarizadas.

• *Nivel 11:* Es responsable de mantener la asociación entre los identificadores externos e in- ternos de los recursos y objetos del sistema.

• *Nivel 12:* Proporciona servicios completos de soporte a los procesos.

• *Nivel 13:* Ofrece al usuario una interfaz con el sistema operativo.

Este modelo hipotético de un sistema operativo proporciona una descripción útil de la estructura, a la vez que sirve como guía de implementación. Se puede volver a esta estructura en el transcurso de este libro para observar el contexto de cualquier aspecto particular de diseño que esté en discusión.

**SISTEMAS DE EJEMPLO**

Para ilustrar los conceptos y aso-ciarlos con las decisiones reales de diseño que se deben tomar, se han seleccionado tres sis- temas operativos actuales:

• *Windows NT:* Un sistema operativo monousuario y multitarea diseñado para que pueda ejecutar sobre una amplia variedad de PC y estaciones de trabajo

• *UNIX:* un sistema operativo multíusuario dirigido originalmente a minicomputadores pero implementado en un amplio rango de máquinas, desde potentes minicomputadores hasta supercomputadores.

• *MVS (Múltiple Virtual Storage):* Es el sistema operativo situado en la cima de la línea de grandes sistemas de IBM y uno de los sistemas operativos más complejos que se han desa- rrollado.

En esta sección se dará una breve descripción e historia de cada uno de estos sistemas operativos.

**WINDOWS NT**

**Historia**

El antepasado más distante de Windows NT es un sistema operativo desarrollado por Microsoft para los primeros computadores personales de IBM y conocido como MS-DOS o PC-DOS. La versión inicial, el DOS 1.0, fue lanzada en Agosto de 1981. Esta constaba de 4000 líneas de código fuente en lenguaje de ensamblador y ejecutaba en 8K de memoria utilizando el microprocesador Intel 8086.

Cuando IBM desarrolló su computador personal basada en disco duro, el PC XT, Micro- soft desarrolló el DOS 2.0, lanzado en 1983. La nueva versión permitía que los directorios pudieran tener subdirectorios, así como archivos. También contenía un conjunto más completo de órdenes incluidas en el sistema operativo, que brindaban funciones que en la versión 1 tenían que llevarse a cabo con programas externos. Entre las capacidades que se le añadieron estaban algunas características del tipo de las de UNIX, tales como el redireccionamiento de E/S, que es la capacidad de cambiar la identidad de la entrada o la salida de una aplicación y la impresión de fondo.

Cuando IBM anunció el PC AT en 1984, Microsoft introdujo el DOS 3.0. El AT incorporaba el procesador Intel 80286, que estaba provisto con un direccionamiento ampliado y con recursos de protección de memoria. Estos no fueron utilizados por el DOS. Aún así, los requisitos de memoria aumentaron a 36KB. Hubo varias actualizaciones notables en la versión 3.0. El DOS 3.1, lanzado en 1984, tenía ya soporte para redes de PC. El tamaño de la parte residente no cambió; esto se consiguió aumentando el volumen del sistema operativo que podía intercambiarse. El DOS 3.3, lanzado en 1987, ya daba soporte para la nueva línea de máquinas IBM, los PS/2. Una vez más, esta versión no sacaba ventajas de las capacidades del procesador del PS/2, provisto con el 80286 y el 80386, de 32 bits. La parte residente del sistema había crecido hasta un mínimo de 46KB.

el DOS estaba siendo utilizado en un entorno que iba más allá de sus capacidades. La introducción del 80486 y del Pentium de Intel introdujeron potencia y características que no pueden ser explotadas por el ingenuo DOS. Mientras tanto, a comienzos de los 80, Microsoft había comenzado a desarrollar una interfaz gráfica de usuario que podía colocarse entre el usuario y el DOS. Hacia 1990, Microsoft tenía una versión de GUI, conocida como Windows 3.0, que se parecía a la interfaz de usuario del Macintosh.

Después de una tentativa frustrada por parte de Microsoft de desarrollar conjuntamente con IBM un sistema operativo de nueva generación, que aprovecharía la potencia de los nuevos microprocesadores y que incorporaría las características de facilidad de uso de Windows, Microsoft continuó por su cuenta y desarrolló Windows NT. Windows NT puede aparecer ante los usuarios como si fuera Windows 3.1, pero está basado en un concepto radical- mente diferente. Windows NT aprovecha la potencia de los microprocesadores actuales y ofrece una multitarea completa en un entorno monousuario.

**Multitarea monousuario**

Windows NT es, quizá, el ejemplo más importante de lo que se ha convertido en la nueva ola de los sistemas operativos de computadores personales.se guió por la necesidad de aprovechar la tremenda potencia de los microprocesadores de 32 bits de hoy en día, los cuales rivalizan con las grandes computadores y las minicomputadores de hace unos pocos años, tanto en velocidad como en sofisticación del hardware y en capacidad de la memoria.

Una de las características más significativas de estos nuevos sistemas operativos es que, aunque siguen estando orientados a dar soporte a un sólo usuario interactivo, son sistemas operativos multitarea. Dos desarrollos principales han disparado la necesidad de la multitarea en los computadores personales. Primero, con el aumento de la velocidad y de la capacidad de memoria de los microprocesadores, junto con el a poyo de la memoria virtual, las aplicaciones se han hecho más complejas e interrelacionadas.

En la medida en que los servicios y las capacidades disponibles para los usuarios se hacen más potentes y variadas, el entorno monotarea se hace más molesto y poco amistoso. En un entorno de multitarea, el usuario abre cada aplicación según lo necesita y *la deja abierta.* La información se puede mover con facilidad entre las distintas aplicaciones. Cada aplicación tiene una o más ventanas abiertas y una interfaz gráfica con un dispositivo que sirve de apuntador, como puede ser un ratón, que permite al usuario navegar con facilidad en este entorno.

Una segunda motivación para la multitarea es el crecimiento del proceso cliente/servidor. Con el proceso cliente/servidor, un computador personal o una estación de trabajo (el cliente) y un sistema anfitrión o *host* (el servidor) se unen para llevar a cabo una aplicación particular. Las dos están conectadas entre sí y a cada una se le asigna la parte del trabajo que esté acorde con sus capacidades. El cliente/servidor se puede lograr en un red local de computadores personales y servidores, o por mediación de un enlace entre un sistema de usuario y un *host* grande, como puede ser un computador central.

**Descripción**

Muchos elementos influyeron en el diseño de Windows NT. Este ofrece el mismo tipo de GUI que los productos anteriores de Windows, incluyendo el uso de ventanas, menús y la interacción "señalar y seleccionar" *(point and click).* Su estructura interna está inspirada en el sistema operativo Mach, el cual está basado, a su vez, en UNIX.

Como casi todos los sistemas operativos, NT diferencia el software de aplicación del soft- ware del sistema operativo. Este último ejecuta en lo que se denomina *modo privilegiado o modo núcleo.* El software en modo núcleo tiene acceso a los datos del sistema y al hardware.

El software restante, que ejecuta en modo usuario, tiene acceso limitado a los datos del sistema. El software en este modo núcleo se denomina ejecutor de NT.

Tanto si está ejecutando sobre un monoprocesador como sobre un multíprocesador, sobre un sistema CISC o uno RISC, la mayor parte de NT tiene la misma visión del hardware sub- yacente. Para alcanzar esta independencia, el sistema operativo consta de cuatro niveles:

• *Capa de Abstracción de hardware:* Establece una correspondencia entre las órdenes y respuestas genéricas del hardware y aquellas que son pro- pias de una plataforma específica, La HAL hace que el bus del sistema de cada máquina, el controlador de DMA, el controlador de interrupciones, los relojes de sistema y el módulo de memoria parezcan los mismos para el núcleo.

• *Núcleo:* Consta de las componentes más usadas y fundamentales del sistema operativo. El núcleo administra la planificación y el cambio de contexto, la gestión de excepciones e interrupciones y la sincronización de multiprocesadores.

• *Subsistemas:* Incluyen varios módulos con funciones específicas que hacen uso de los ser- vicios básicos proporcionados por el núcleo.

• *Servicios del sistema:* Ofrece una interfaz al software en modo usuario.

La potencia de Windows NT proviene de su habilidad para dar soporte a aplicaciones escritas para otros sistemas operativos. La forma en que Windows NT da este soporte con un ejecutor único y compacto es a través de los subsistemas protegidos. Los subsistemas protegidos ofrecen al usuario una interfaz gráfica o de líneas de órdenes que definen la apariencia del sistema operativo para un usuario. cada subsistema protegido proporciona la interfaz para los programas de aplicación del entorno particular de operación. Esto significa que las aplicaciones creadas para un entorno particular de operación pueden ejecutarse sin modificaciones sobre NT, porque la interfaz del sistema operativo que ven es la misma que para la que fueron escritas. De modo que, por ejemplo, las aplicaciones basadas en OS/2 pueden ejecutarse en NT sin modificaciones. Más aún, puesto que el sistema NT está en sí mismo diseñado para que sea independiente de la plataforma, a través del uso de la HAL.

Cada servidor se implementa con uno o más procesos. Cada proceso espera de un cliente la solicitud de un servicio, como por ejemplo, servicios de memoria, de creación de proce- sos o de planificación del procesador. Un cliente, que puede ser un programa de aplicación u otro módulo del sistema operativo, solicita un servicio enviando un mensaje. El mensaje es encaminado a través del ejecutor hasta el servidor apropiado. El servidor lleva a cabo las operaciones solicitadas y devuelve los resultados o la información de estado por medio de otro mensaje, que es encaminado a través del ejecutor de regreso al cliente.

[CLJST93] señala las ventajas siguientes en una arquitectura cliente/servidor:

• Simplifica el sistema operativo de base, el ejecutor de NT. Puesto que el ejecutor no ofrece una API, es posible construir varias API sin ningún conflicto o duplicación en el ejecutor. Se pueden añadir fácilmente nuevas API.

• Mejora la fiabilidad. Cada servidor ejecuta un proceso independiente, con su propia partición de memoria, protegido de otros servidores. Más aún, los servidores no pueden acceder directamente al hardware o modificar la memoria en la que se almacena el ejecutor. Un servidor puede fallar sin que se hunda o corrompa el resto del sistema operativo.

• Proporciona una base natural para el proceso distribuido. En NT, un servidor local puede pasar un mensaje a otro remoto para su procesamiento en nombre de las aplicaciones locales clientes. Los clientes no necesitan saber si una solicitud es atendida por un servidor local o remoto. El hecho de que una solicitud sea atendida de forma local o remota puede cambiar dinámicamente.

**Hilos**

Un aspecto importante de Windows NT es su soporte de hilos dentro de los procesos. Los hi- los incorporan algunas de las funciones asociadas tradicionalmente con los procesos. Se puede hacer la siguiente distinción:

• *Hilo:* Una unidad de trabajo que se puede expedir para ejecución. Es lo que se ejecuta se- cuencialmente y es interrumpible para que el procesador puede pasar a otro hilo.

• *Proceso:* Una colección de uno o más hilos y recursos del sistema asociados. Esto se acerca mucho al concepto de programa en ejecución.

**Multiproceso simétrico**

Hasta hace poco, casi todos los computadores personales y las estaciones de trabajo tenían un único microprocesador de propósito general. A medida que las demandas de rendimiento aumentan y el coste de los microprocesadores continúa bajando, esta imagen cambia. Los fabricantes están introduciendo sistemas con varios microprocesadores. una máxima eficiencia y fiabilidad, es conveniente un modo de operación conocido como *multiproceso simétrico* (SMP), cualquier hilo o proceso puede ser asignado a cualquier procesador. Esto incluye a los procesos e hilos del sistema operativo.

Uno de los requisitos clave en el diseño de los sistemas operativos contemporáneos es la capacidad de explotar el SMP. [CUT93] enumera las siguientes características de Windows NT que dan soporte al uso de SMP:

• Las rutinas del sistema operativo se pueden ejecutar en cualquier procesador disponible y rutinas diferentes pueden ejecutarse simultáneamente en diferentes procesadores.

• NT permite el uso de varios hilos de ejecución dentro de un solo proceso. En un mismo proceso pueden ejecutarse varios hilos en diferentes procesadores simultáneamente.

• Los procesos servidores pueden utilizar varios hilos para procesar solicitudes de más de un cliente simultáneamente.

• NT brinda los mecanismos oportunos para compartir datos y recursos entre los procesos, así como capacidades flexibles de comunicación entre procesos.

**Objetos de Windows NT**

Windows NT se sustenta fuertemente en los conceptos del diseño orientado a objetos. Este enfoque facilita la compartición de recursos y datos entre los procesos y la protección de los recursos ante usuarios no autorizados. Entre los conceptos clave empleados por NT están los siguientes:

• *Encapsulamiento:* Un objeto consta de uno o más elementos de datos, llamados *atributos* y uno o más procedimientos que pueden actuar sobre los datos, llamados *servicios.* La única manera de acceder a los datos de un objeto es invocando a uno de los servicios del objeto. Así pues, los datos en el objeto pueden ser protegidos fácilmente ante usos no autorizados e incorrectos (por ejemplo, tratar de ejecutar un elemento de datos no ejecutable).

• *Clases e instancias:* Una clase de objeto es una plantilla que enumera los atributos y los servicios de un objeto y define ciertas características del objeto. El sistema operativo puede crear instancias específicas de una clase de objeto cada vez que lo necesite. Por ejemplo, hay una clase de objeto para procesos simples y un objeto proceso para cada pro- ceso en activo. Este enfoque simplifica la creación y gestión de objetos.

En NT, los objetos pueden tener nombre o no. Cuando un proceso crea un objeto sin nombre, el administrador de objetos devuelve un descriptor del objeto, que es la única forma de hacer referencia al objeto. Los objetos con nombre disponen de un nombre que pueden usar otros procesos para obtener un descriptor del objeto. Por ejemplo, si el pro- ceso A desea sincronizarse con el proceso B, podría crear un objeto suceso con nombre y pasarle a B el nombre del suceso. El proceso B podría abrir y usar este objeto suceso. Sin embargo, si A sólo desea utilizar el suceso para sincronizar dos hilos dentro de sí mismo, entonces podría crear un objeto suceso sin nombre, porque no hay necesidad de poder usar el suceso para otros procesos.

Una distinción importante entre los objetos con nombre y sin nombre es que los primeros siempre tienen información de seguridad asociada a ellos, en forma de una señal de acceso *(access token).* Puede emplearse la información de seguridad para restringir el acceso al objeto. Por ejemplo, un proceso puede crear un semáforo con nombre con la intención de que sólo los procesos que conozcan su nombre sean capaces de abrir y usar el semáforo. La señal de acceso asociada con el objeto semáforo enumerará aquellos procesos (en realidad, los identificadores de los usuarios que son propietarios de los procesos) que pueden acce- der al semáforo.

Windows NT no es un sistema operativo completamente orientado a objetos. No está implementado en un lenguaje orientado a objetos. Las estructuras de datos que residen por completo dentro de un mismo componente del ejecutor no se representan como objetos. Además, NT no ofrece soporte para algunas capacidades habituales de los sistemas orienta- dos a objetos, tales como la herencia y el polimorfismo. No obstante, NT ilustra la potencia de la tecnología orientada a objetos y representa la tendencia creciente a usar esta tecnología en el diseño de sistemas operativos.

**SISTEMA UNIX, VERSIÓN V**

**Historia**

La historia de UNIX es una historia muchas veces contada que no se va a repetir aquí con mucho detalle. En su lugar, se ofrecerá un breve resumen.

UNIX fue desarrollado inicialmente en los Laboratorios Bell y llegó a ser operativo en una PDP-7 en 1970. Algunas personas involucradas de los Laboratorios Bell también habían participado en el trabajo sobretiempo compartido que se llevaba a cabo en el proyecto MAC del MIT. Aquel proyecto llevó primero al desarrollo del CTSS y luego al de Multics. Aun- que es habitual decir que UNIX es una versión a pequeña escala de Multics, los desarrolla- dores de UNIX dicen estar más influenciados en realidad por CTSS .

Merece la pena comentar algo sobre Multics. Multics estuvo no sólo años sino décadas adelantado a su tiempo. Incluso a mediados de los 80, casi 20 años después de que llegara a ser operativo, Multics disponía de características de seguridad superiores y una mayor sofisticación en la interfáz de usuario y en otros campos que los sistemas operativos contemporáneos de los grandes computadores que puedan compararse. Aunque los desarrolladores de UNIX abandonaron el proyecto de Multics porque, según su opinión, éste fue un fracaso, Multics fue cedido después a Honeyweil y llegó a disfrutar de un modesto éxito comercial. Lo que tuvo Honeyweil no lo tuvieron los otros dos sistemas operativos de grandes computadores, uno de los cuales fue comercializado de forma muy agresiva. Multics pudo haber tenido un gran éxito.

Mientras tanto, el trabajo sobre UNIX en los Laboratorios Bell y, después, en otros luga- res, produjo una serie de versiones de UNIX. El primer hito notable fue llevar el sistema UNIX de la PDP-7 a una PDP-11. Esta fue la primera señal de que UNIX sería un sistema operativo para todos los computadores. El siguiente hito importante fue la reescritura de UNIX en el lenguaje de programación C. La implementación en C demostró las ventajas de usar un lenguaje de alto nivel para la mayor parte del código del sistema, si no todo. Hoy en día, casi todas las implementaciones de UNIX están escritas en C.

Estas primeras versiones de UNIX fueron muy populares en los Laboratorios Bell. En 1974, el sistema UNIX fue descrito por primera vez en una revista técnica [RITC74]. Esto despertó gran interés en el sistema. Se otorgaron licencias de UNIX a instituciones comer- ciales y universidades. La primera versión ampliamente disponible fuera de los Laboratorios Bell fue la Versión 6, en 1976. La siguiente, la versión 7, lanzada en 1978, es el antepasado de la mayoría de los sistemas UNIX modernos. El más importante de los sistemas no desa- rrollados por AT&T8 fue el realizado en la Universidad de California en Berkeley. Se le llamó UNIX BSD y ejecutaba primero en una PDP y, más tarde, en máquinas VAX. AT&T continuó desarrollando y refinando el sistema. Hacia 1982, los Laboratorios Bell habían combinado varias variantes del UNIX de AT&T en un único sistema, que fue comerciali- zado como Sistema ÜNÍX, versión *III.* Posteriormente se le añadió un cierto número de elementos hasta llegar al Sistema UNIX, versión V.

**Descripción**

El hardware bá- sico está rodeado por el software del sistema operativo. El sistema operativo se llama a me- nudo *núcleo del sistema* o, simplemente, núcleo *(kernel),* para realzar su aislamiento de las aplicaciones y de los usuarios. Sin embargo, UNIX viene equipado con una serie de servicios de usuario e interfaces que se consideran parte del sistema. Estos pueden agruparse en un *shell,* algún otro software de interfaz y las componentes del compilador de C. La capa exterior está formada por las aplicaciones de los usuarios y una interfaz de usuario con el compilador C.

Los programas de usuario pueden invocar a los servicios del sistema operativo directamente o a través de programas de biblioteca. La interfaz de llamadas al sistema es la frontera con el usuario y le permite al software de alto nivel el acceso a las funciones específicas del núcleo. En el otro extremo, el sistema operativo contiene rutinas primitivas que interactúan directamente con el hardware. Entre estas dos interfaces, el sistema está dividido en dos partes fundamentales, una ocupada del control de los procesos y la otra relacionada con la gestión de archivos y la E/S. El sub- sistema de control de procesos es el responsable de la gestión de memoria, la planificación y expedición de los procesos y la sincronización y comunicación entre procesos. El sistema de archivos intercambia los datos entre la memoria y los dispositivos externos, tanto en flujos de caracteres como en bloques. Para lograr esto, se utilizan varios manejadores de dispositivo. Para las transferencias de bloques se utiliza un método de cache de disco: Se coloca un buffer del sistema en la memoria principal entre el espacio de direcciones del usuario y el dispositivo externo.

**MVS**

**Historia**

Hacia 1964, IBM estaba firmemente establecida en el mercado de computadores con sus máquinas de la serie 7000. En aquel año, IBM anunció el Sistema/360, una nueva familia de productos de computadores. Aunque el anuncio en sí no fue una sorpresa, venía con algunas noticias poco agradables para los clientes de IBM: la línea de productos 360 era incompatible con las viejas máquinas de IBM. Por tanto, la transición al 360 sería difícil para la base de usuarios. Esto fue un paso audaz de IBM pero que se creía necesario para romper con algunas de las limitaciones de la arquitectura 7000 y fabricar un sistema capaz de evolucionar con la nueva tecnología de circuitos integrados. La estrategia salió bien tanto financiera como técnicamente. El 360 fue el éxito de la década y asentó IBM como el irresistible vendedor de computadores con una cuota de mercado superior al 70%. Con algunas modificaciones y ampliaciones, La arquitectura del 360 se ha mantenido hasta el día de hoy, más de un cuarto de siglo después, como la arquitectura de Los grandes computadores IBM.

Aunque el software del sistema operativo de los computadores centrales de IBM ha cambiado más allá de lo que se reconoce como la oferta inicial del 360, sus orígenes se remontan a aquellos primeros tiempos. El Sistema/360 fue la primera familia de computadores planificada por la industria. La familia cubría un amplio rango de rendimiento y coste. Los diferentes modelos eran compatibles en el sentido de que un programa escrito para un modelo era capaz de ejecutarse en otro modelo de la misma familia con diferencias sólo en el tiempo necesario para ejecutar. Por esto, el primer sistema operativo que fue anunciado para las nuevas máquinas, el 05/360, intentaba ser un sistema operativo ünico que pudiera operar con todas las máquinas de la familia. Sin embargo, el amplio abanico de la familia 360 y la diversidad de las necesidades del usuario obligaron a IBM a introducir varios sistemas operativos. Se centrará la atención en la línea de desarrollo que originó el MVS.

El OS/360 original fue un sistema por lotes con multiprogramación y permaneció así por un cierto tiempo. Su versión más completa, el MVT (Multiprogramación con un número Variable de Tareas), fue lanzado en 1969 y fue la más flexible de las variantes del OS/360. La asigna- ción de memoria para un trabajo era variable y no tenla que decidirse hasta la ejecución. Llegó a ser el sistema operativo más popular de las grandes IBM/360 y en Las primeras IBM/370. MVT omitió algunas de las características presentes en las ofertas de sus competidores más avanzados, como la capacidad para dar soporte a multiprocesadores, el almacenamiento virtual y depuración a nivel de fuente, pero brindó un conjunto de servicios y utilidades de apoyo más completo que cualquiera de los sistemas operativos contemporáneos. En campos tales como la planificación de trabajos, el soporte de periféricos, la cantidad de sistemas diferentes y la conversión desde los sistemas anteriores, el 0S/360 fue inigualable.

MVT permitía que sólo ejecutaran 15 trabajos concurrentemente. OS/SVS *(Single Virtual Storage,* Almacenamiento Virtual Simple) fue introducido en el año 1972 como un sistema operativo provisional para sacar partido de la arquitectura IBM/370. El añadido más notable fue dar soporte a la memoria virtual. En el SVS se establecía un espacio de direcciones vir- tual de 16MB. Sin embargo, este espacio de direcciones tenía que ser compartido entre el sistema operativo y todos los trabajos activos. Muy pronto, incluso esta cantidad de memo- ria se haría inadecuada.

Como respuesta al crecimiento de las necesidades de memoria de los programas de apli- cación, IBM introdujo el MVS. Al igual que en el SVS y como dictaba la arquitectura 370, las direcciones virtuales estaban limitadas a 24 bits (de ahí los 16 MB). Sin embargo, con el MVS, el límite es de 16 MB *por trabajo.* Es decir, cada trabajo tiene su propia memoria vir- tual dedicada de 16 MB. El sistema operativo traduce los bloques de memoria virtual a la memoria real y es capaz de seguir la pista de las memorias virtuales separadas para cada trabajo. En realidad, cada trabajo normalmente dispone de algo menos de la mitad de la memoria virtual asignada; el resto está disponible para el sistema operativo.

El espacio de direcciones de 24 bits, con memoria virtual separada para cada trabajo, en breve llegó a ser poco apropiado para algunos usuarios. IBM amplió su procesador básico para que manejase direcciones de 31 bits, una característica conocida como *direcciones ampliadas* (XA, *extended addressing).* Para sacar partido de este nuevo hardware, en 1983 se introdujo una nueva versión de MVS, conocida como MVS/XA. Con el MVS/XA, el espacio de direcciones por tarea creció a un máximo de 2GB (gigabytes). Esto, se crea o no, aún se considera inapropiado para algunos entornos y aplicaciones. Por consiguiente, en lo que puede representar la última mayor ampliación de la arquitectura 370 (de la 360, en realidad), IBM desarrolló la Arquitectura de Sistemas Empresariales (ESA, *Enterprise System Architecture)* y un sistema operativo mejorado, el MVS/ESA. El mismo espacio de direcciones de 2GB por trabajo que estaba disponible en el MVS/XA está también disponible para programas y datos. Lo nuevo es que hay hasta 15 espacios de direcciones adicionales de 2GB de datos disponibles sólo para un trabajo específico. Por tanto, el espacio máximo direccionable de memoria virtual por trabajo es de 32GB.

**Descripción**

MVS es, probablemente, el sistema operativo más grande y complejo desarrollado jamás. Requiere un mínimo de 2MB de almacenamiento residente. Una configuración más realista requiere 6MB para el sistema operativo. Los cuatro factores más importantes que han deter- minado el diseño de MVS son los siguientes:

• Soporte para trabajos interactivos y por lotes

• Almacenamiento virtual de hasta 32GB por trabajo o usuario

• Multiproceso fuertemente acoplado; esta arquitectura, que se examinará en el capítulo 9, consiste en una serie de procesadores que comparten la misma memoria principal.

• Asignación sofisticada de recursos y servicios de supervisión para lograr un uso eficiente de la gran memoria del sistema, múltiples procesadores y estructura compleja de canales de E/S.

La necesidad de tratar con varios procesadores es un requisito que no se plantearon OS/2 ni el Sistema UNIX, versión V. Cuando hay varios procesadores, donde cada uno de los cuales puede ejecutar cualquiera de los procesos y cuando se da soporte a la comunicación entre procesos que ejecutan en diferentes procesadores, la complejidad del sistema operativo puede ser significativamente mayor que en una máquina con un monoprocesador.

Un *shell* extemo contiene los servicios y las interfaces visibles para los usuarios y los operado- res del sistema responsables de su mantenimiento y ajuste. Además de una colección de pro- gramas necesarios para la creación y compilación de programas, hay un subsistema de ges- tión de trabajos con las siguientes funciones:

• Interpreta las órdenes del operador (desde la consola del operador) y encamina los mensa- jes adecuados.

• Lee los datos de entrada del trabajo y escribe los datos de salida del trabajo en los disposi- tivos periféricos.

• Asigna los dispositivos de E/S a un trabajo y notifica al operador de cualquier unidad física de datos (rollo de cinta, paquete de discos) que debe montarse antes de ejecutar el trabajo.

• Convierte cada trabajo en tareas que pueden ser procesadas por el administrador de tareas.

Por último, el *shell* externo incluye un elaborado subsistema de gestión para la recupera- ción de errores, que asegura que los fallos del trabajo quedan aislados de modo que no difi- culten el resto del funcionamiento y que puedan diagnosticarse. Sí es posible, se permitirá a los trabajos afectados por el error que continúen su ejecución.

El núcleo del sistema operativo consta de un conjunto de módulos principales o subsiste- mas que interactúan uno con otro, el hardware y el *shell* externo. Estos incluyen lo siguiente:

*• Distribuidor:* El distribuidor puede verse como el administrador de los procesadores. Su función es recorrer la cola de tareas listas y planificar la ejecución de una.

*• Tratamiento de interrupciones:* La arquitectura del Sistema/370 da soporte a una amplia variedad de interrupciones. Cualquier interrupción hace que se suspenda el proceso en curso y se pase el control a la rutina de tratamiento apropiada.

*• Gestión de tareas:* Una tarea es, básicamente, un proceso, en el sentido en que se ha usado este término. La gestión de tareas es la responsable de la creación y eliminación de las ta- reas, el control de sus prioridades, la gestión de las colas de tareas de los recursos reutili- zables en serie (por ejemplo, los dispositivos de E/S) y la sincronización de los sucesos,

*• Gestión de pro gramas:* Este módulo es responsable de enlazar los pasos necesarios involu-crados en la ejecución de un programa. Este módulo puede ser controlado por órdenes del JCL o en respuesta a las solicitudes de los usuarios para compilar y ejecutar un programa.

*• Gestión del almacenamiento:* Este módulo es el responsable de administrar la memoria real y virtual.

*• Gestión de recursos del sistema:* Este modulo es el responsable de la asignación de los re-cursos a los espacios de direcciones (procesos).

*• Métodos de acceso:* Un programa de aplicación suele emplear un método de acceso para llevar a cabo una operación de E/S. Un método de acceso es una interfaz entre el programa de aplicación y el supervisor de E/S. Los métodos de acceso liberan al programa de aplicación de la carga de escribir programas para los canales, construyendo los bloques de control requeridos por el supervisor de B/S y manejando las condiciones de terminación.

*• Supervisor de E/S*: El supervisor de E/S lleva a cabo la inicialización y terminación de las operaciones de E/S a nivel del hardware. Genera la instrucción de inicio de E/S, que provoca que el procesador de un canal de E/S ejecute un programa de E/S en la memoria principal y también trata la interrupción que se origina al completarse la operación de E/S.

Merece la pena comentar unas palabras sobre el gestor de recursos del sistema (*System Resource Manager).* El SRM dota al MVS de un grado de sofisticación único entre los sistemas operativos. Ningún otro sistema operativo de computadores centrales e, incluso, ningún otro sistema operativo, puede igualar las funciones que lleva a cabo el SRM.

El concepto de recurso incluye al procesador, la memoria real y los canales de E/S. Para llevar a cabo la tarea de asignación de recursos, el SRM acumula estadísticas relativas al uso del procesador, los canales y varias estructuras de datos clave. Su propósito es el de ofrecer un rendimiento óptimo, basándose en el análisis y supervisión del rendimiento. Durante la instalación se establecen varios objetivos de rendimiento y estos sirven de gula al SRM, que modifica dinámicamente las características de la instalación y del rendimiento de los trabajos en función de la utilización del sistema. Sucesivamente, el SRM ofrece los informes que capacitan al operador formado para refinar la configuración y los valores de los parámetros y así mejorar el servicio al usuario.

Un ejemplo puede mostrar el sabor de las actividades del SRM. La memoria real se divide en bloques de igual tamaño, denominados *marcos* (encuadres), de los que puede haber varios millares. Cada marco puede albergar un bloque de memoria virtual, que se conoce como *página.* El SRM recibe el control aproximadamente 20 veces por segundo e inspecciona cada uno de los marcos de página. Si la página no ha sido referenciada o cambiada, se incrementa en 1 un contador. Después de un cierto periodo, el SRM promedia estos números para determinar el número medio de segundos que una página permanece sin tocar. El operador del sistema puede revisar esta cantidad para determinar el nivel de "tensión" del sistema. Reduciendo el número de trabajos activos permitidos en el sistema, este promedio puede mantenerse alto. Una guía útil es que la media debe mantenerse por encima de los 2 minutos para evitar problemas serios de rendimiento [JOHN89], Esto puede parecer demasiado, pero no lo es.

**CAPITULO 3**

**Descripción y control de procesos.**

El diseño de un sistema operativo debe reflejar con seguridad los requisitos que se pretende que éste cumpla. Todos los sistemas operativos de multiprogramación, están construidos en torno al concepto de proceso, los requisitos principales que debe satisfacer un sistema operativo están expresados haciendo referencia a los procesos:

• El sistema operativo debe intercalar la ejecución de un conjunto de procesos para maxi-mizar la utilización del procesador ofreciendo a la vez un tiempo de respuesta razonable.

• El sistema operativo debe asignar los recursos a los procesos en conformidad con una política especifica, evitando, al mismo tiempo, el interbloqueo.

• El sistema operativo podría tener que dar soporte a la comunicación entre procesos y la creación de procesos por parte del usuario, labores que pueden ser de ayuda en la estructuración de las aplicaciones.

**ESTADOS DE UN PROCESO**

La misión principal del procesador es ejecutar las instrucciones de la máquina que residen en la memoria principal desde el punto de vista del procesador, éste ejecutará instrucciones de entre un repertorio en una secuencia dictada por los valores cambiante de un registro conocido como el *contador de programa* **(PC,** *Program Counter)* o puntero a las instrucciones. La ejecución de un programa individual se conoce como *proceso* o *tarea.*

El comportamiento de un proceso individual puede caracterizarse por el listado de La secuencia de instrucciones que se ejecutan para dicho proceso. Dicho listado se llama *traza* del proceso.

Se supone que el sistema operativo permite a un proceso continuar su ejecución solo por un máximo de seis ciclos de instrucción, después de los cuales es interrumpido; esto impide que un solo proceso monopolice el tiempo del procesador.

**Un modelo de procesos con dos estados**

La responsabilidad principal del sistema operativo es el control de la ejecución de los pro-cesos; esto incluye la determinación de las pautas de intercalado que se van a seguir y la asignación de recursos a los procesos. El primer paso ma que controle los procesos es describir el comportamiento que se los procesos.

El modelo más sencillo que puede construirse tiene en cuenta que, en un momento dado, un proceso puede estar ejecutándose en el procesador o no. Así pues, un proceso puede estar en uno de dos estados: Ejecución o No Ejecución. Incluso en este modelo tan simple ya se comienzan a apreciar algunos de los elementos de diseño del sistema operativo. Cada proceso debe representarse de forma que el sistema operativo pueda seguirle la pista. Esto es, debe haber información relativa a cada proceso, incluyendo su estado actual y su posición en memoria. Aquellos procesos que no están ejecutándose tienen que guardarse en algún tipo de cola, para que esperen su tumo de ejecución.

Tradicionalmente, todos los procesos eran creados por el sistema operativo de una forma transparente para el usuario o el programa de aplicación y es así como todavía se mantiene en la mayoría de los sistemas operativos actuales. Sin embargo, puede ser útil permitir que un proceso pueda originar la creación de otro proceso. Cuando un proceso es creado por el sistema operativo tras la solicitud explicita de otro proceso, la acción se conoce como *generación de procesos (process spawning).*

Cuando un proceso genera otro, el proceso generador se conoce como *proceso padre* y el proceso generado es el *proceso hijo.*

***Terminación de procesos***.

Un trabaja por lotes debe incluir una instrucción de detención (*Halt*) o una llamada explicita a un servicio del sistema operativo para la terminación. en un sistema de tiempo compartido, el proceso de un usuario particular terminará cuando éste se desconecte del sistema o apague el terminal. En un computador personal o una estación de trabajo, el usuario puede abandonar una aplicación.

**Creación y terminación de procesos**

la vida de un proceso está limitada por su creación y su terminación .

Creación de procesos

Cuando se añade un proceso a los que ya está administrando el sistema operativo, hay que construir las estructuras de datos que se utilizan para administrar el proceso y asignar el espacio de direcciones que va a utilizar el proceso. Estas acciones constituyen la creación de un nuevo proceso.

Cuatro sucesos comunes llevan a la creación de un proceso, primero, en un entorno de trabajo por lotes, un proceso se crea como respuesta a la remisión de un trabajo. En un entorno interactivo, se crea un proceso cuando un nuevo usuario intenta conectarse. En ambos casos, el sistema operativo es el responsable de la creación del nuevo proceso. Un tercer caso en el que el sistema operativo crea un proceso es de parte de una aplicación.

**Un modelo de cinco estados**

Si todos los procesos estuvieran siempre listos para ejecutar, entonces la disciplina de cola de 2 estados seria eficaz. La cola es una lista “primero en entrar, primero en salir” y el procesador opera según un turno rotatorio *(round-robin)* con todos los procesos disponibles.

Algunos procesos en el estado de No Ejecución están listos *para* ejecutar, mientras que otros están bloqueados. utilizando una cola sencilla, Más bien, el distribuidor tendría que recorrer la lista buscando el proceso que no este no bloqueado" y que lleve mas tiempo en la cola.

Una forma más es dividir el estado de No Ejecución en dos estados: *Listo y Bloqueado.* Los cinco estados de este nuevo diagrama son los siguientes:

• *Ejecución:* El proceso que está actualmente en ejecución.

• *Listo:* Proceso que está preparado para ejecutar, en cuanto se le dé la oportunidad.

• *Bloqueados:* Proceso que no puede ejecutar hasta que se produzca cierto suceso, como la terminación de una operación de E/S.

• *Nuevo:* Proceso que se acaba de crear, pero que aún no ha sido admitido por el sistema operativo en el grupo de procesos ejecutables.

• *Terminado:* Un proceso que ha sido excluido por el sistema operativo del grupo de procesos ejecutables, bien porque se detuvo o porque fue abandonado por alguna razón.

Los estados Nuevo y Terminado son construcciones útiles para la gestión de procesos. El estado Nuevo corresponde a los procesos que acaban de ser definidos.

un proceso sale de un sistema en dos pasos. Primero, el proceso termina cuando liega al punto normal de terminación, cuando se abandona debido a un error irrecuperable o cuando otro proceso con la debida autoridad hace que el proceso abandone. las tablas y otra información asociada con el trabajo son conservadas temporalmente por el sistema operativo. Se verán cada una de las posibilidades por turnos:

*• Nulo → Nuevo:* Se crea un nuevo proceso para ejecutar un programa.

*• Nuevo → Listo:* El sistema operativo pasará un proceso del estado Nuevo al estado Listo cuando esté preparado para aceptar un proceso más.

*• Listo →* - *Ejecución:* Cuando es hora de seleccionar un nuevo proceso para ejecutar, el sistema operativo elige a uno de los procesos del estado Listo.

*• Ejecución → Terminado:* El proceso que se está ejecutando es finalizado por el sistema operativo si indica que terminó 0 Si se abandona.

*• Ejecución → Listo:* La razón más común de esta transición es que el proceso que está en ejecución ha alcanzado el tiempo máximo permitido de ejecución interrumpida; casi todos los sistemas operativos con multiprogramación imponen este tipo de norma de tiempo

*• Ejecución → Bloqueado:* Un proceso se pone en el estado Bloqueado si solicita algo por lo que debe esperar.

*• Bloqueado* → *Listo:* Un proceso que está en el estado Bloqueado pasará al estado Listo cuando se produzca el suceso que estaba esperando.

*• Listo* → *Terminado:* Por razones de claridad, esta transición no se muestra en el diagrama de estados.

*• Bloqueado* → *Terminado:* Se aplica el mismo comentario que en el caso anterior.

Ahora hay dos colas: Una cola de Listos y una cola de Bloqueados. A medida que se admiten procesos en el sistema, se sitúan en la cola de Listo. Cuando llega la hora de que el sistema operativo escoja otro proceso para ejecutar, selecciona uno de la cola de Listos. En ausencia de un esquema de prioridades, Cuando un proceso que se está ejecutando es apartado de la ejecución, cuando se produce un suceso, et sistema operativo debe recorrer toda la cola de Bloqueados, buscando aquellos procesos que esperan al suceso. En un sistema operativo grande, puede haber cientos o incluso miles de procesos en dicha cola. Por tanto, serla más eficiente tener una serie de colas, una para cada suceso.

Un retoque final: es conveniente tener un cierto número de colas de Listos, una para cada nivel de prioridad. El sistema operativo podrá determinar fácilmente cuál es el proceso de prioridad más alta que lleva más tiempo esperando.

Procesos suspendidos

***Necesidad del intercambio***

Los tres estados principales que se han descrito (Listo, Ejecución, Bloqueado) Se han construido muchos sistemas operativos empleando solamente estos tres estados.

razón de todo este complicado mecanismo es que las actividades de E/S son mucho más lentas que las de cálculo y, por tanto, el procesador en un sistema de mono-programación está parado la mayor parte del tiempo.. Pero el procesador es tan rápido comparado con la E/S que suele ser habitual que todos los procesos de memoria estén esperando por E/S. Así pues, incluso con multiprogramación, el procesador podría estar desocupado la mayor parte del tiempo.

Primero, se tiene un coste asociado con la memoria principal, que, aunque es pequeño cuando se trata de bits, comienza a acumularse cuando se trata de megabytes y gigabytes de almacenamiento. Segundo, el apetito de los programas por la memoria ha crecido tan rápidamente como el coste de la memoria ha disminuido. Por tanto, una memoria mayor significa procesos mayores, pero no más procesos.

Otra solución es el intercambio, lo que significa mover una parte del proceso o todo el proceso de la memoria principal a disco. El intercambio, es una operación de E/S y, por tanto, hay una posibilidad de que el problema empeore en vez de mejorar. Pero como la E/S con el disco es, en general, la E/S más rápida de un sistema, el intercambio suele mejorar el rendimiento.

Cuando el sistema operativo haya realizado una operación de intercambio de un proceso a disco, tendrá dos opciones para seleccionar el proceso que se va a traer a memoria: Puede admitir un proceso recién creado o puede traer un proceso suspendido previamente. Puede parecer que la preferencia debe ser traer un proceso suspendido previamente para darle servicio, en lugar de hacer crecer la carga total de procesos en el sistema.

no se necesita volver a pensar sobre este aspecto del diseño. Aquí se tienen dos conceptos independientes: si un proceso está esperando un suceso (bloqueado o no), y si un proceso ha sido expulsado de la memoria principal (suspendido o no). Para ordenar estas combinaciones, hacen falta los cuatro estados siguientes:

• *Listo:* El proceso está en memoria principal y listo para la ejecución.

• *Bloqueado:* El proceso está en memoria principal esperando un suceso.

• *Bloqueado y suspendido:* El proceso está en memoria secundaria esperando un suceso.

• *Listo y suspendido:* El proceso está en memoria secundaria pero está disponible para su ejecución tan pronto como se cargue en la memoria principal.

En la discusión se ha supuesto hasta ahora que no se utiliza memoria virtual y que un proceso estará bien en memoria principal o bien fuera de ella por completo. Con un esquema de memoria virtual, es posible ejecutar un proceso que esté solo parcialmente en memoria principal. El uso de la memoria virtual parece eliminar la necesidad del intercambio explicito, ya que cualquier dirección deseada de cualquier proceso puede ser trasladada dentro o fuera de la memoria principal por el hardware de gestión de memoria del procesador. incluso en un sistema de memoria virtual, el sistema operativo siempre tendrá que expulsar de cuando en cuando algunos procesos, de forma completa y explicita, en aras del rendimiento.

Las nuevas e importantes transiciones son las siguientes:

*• Bloqueado* → *Bloqueado y suspendido:* Si no hay procesos Listos, entonces al menos un proceso Bloqueado se expulsa para dar cabida a otro proceso que no esté bloqueado.

*• Bloqueado y suspendido* → *Listo y suspendido:* Un proceso en estado Bloqueado y sus-pendido se pasa al estado Listo y suspendido cuando ocurre el suceso que estaba esperando.

*• Listo y suspendido* → *Listo:* Cuando no hay procesos Listos en la memoria principal, el sistema operativo tendrá que traer uno para continuar la ejecución.

*• Listo* → *Listo y suspendido:* Generalmente, el sistema operativo prefiere suspender a un proceso Bloqueado en vez de a uno Listo, ya que el proceso Listo podría ejecutarse de inmediato, mientras que el proceso Bloqueado estará ocupando espacio en la memoria principal sin poder ejecutarse.

Otras transiciones son también dignas de consideración:

***• Nuevo* → *Listo, Suspendido y Nuevo* → *Listo:*** Cuando se crea un nuevo proceso, se le puede añadir a la cola de listos o a la de listos y suspendidos.

• *Bloqueado y suspendido → Bloqueado:* La inclusión de esta transición puede parecer resultado de un mal diseño..

• *Ejecución → Listo y suspendido:* Generalmente, un proceso en Ejecución pasa al estado Listo cuando expira su fracción de tiempo asignado.

• *Varios → Terminado:* Normalmente, los procesos terminan mientras están ejecutándose, bien porque se completaron o bien por causa de alguna condición drástica de error.

**Otros usos de la suspensión**

el concepto de proceso Suspendido con el hecho de que el proceso no está en memoria principal. Un proceso que no esté en memoria no estará disponible de inmediato para su ejecución, esté o no esperando un suceso.

Se define proceso Suspendido como aquel que tiene las características siguientes:

1. Un proceso que está suspendido no está disponible de inmediato para ejecución.

2. El proceso puede estar esperando o no un suceso. Si lo está, la condición de Bloqueado es independiente de La condición de Suspendido y el acontecimiento del suceso bloqueante no lo habilita para la ejecución.

3. El proceso fue situado en el estado suspendido por un agente (por sI mismo, por el pro-ceso padre o por el sistema operativo) con el fin de impedir su ejecución.

4. El proceso no puede apartarse de este estado hasta que el agente lo ordene explícitamente.

Una razón que ya se ha discutido es la necesidad de expulsar un proceso a disco para dar cabida a un proceso Listo o, simplemente, para aliviar la presión sobre el sistema de memoria virtual de forma que los procesos restantes tengan disponible más memoria principal. El sistema operativo puede tener otros motivos para suspender un proceso. Por ejemplo, puede emplearse un proceso de auditoria o de seguimiento para supervisar la actividad del sistema otro ejemplo:

Se detecta un problema en una línea de comunicaciones y el operador tiene que hacer que el sistema operativo suspenda al proceso que esté usando la línea mientras que se ejecutan algunas pruebas.

Por ejemplo, si un proceso se va a activar periódicamente, pero está libre la mayor parte del tiempo, entonces deberla ser expulsado entre cada uso. Un ejemplo es el de un programa que supervise La utilización o la actividad de los usuarios.

**DESCRIPCIÓN DE PROCESOS**

El sistema operativo es el controlador de los sucesos que se producen en un sistema informático. Es el sistema operativo el que planifica y expide a los procesos para su ejecución en el procesador, el que asigna los recursos a los procesos y el que responde a las solicitudes de servicios básicos realizadas por los programas de usuario. Esencialmente, se puede imaginar al sistema operativo como una entidad que administra el uso que hacen los procesos de los recursos del sistema.

Durante el curso de su ejecución, cada proceso necesita tener acceso a ciertos recursos del sistema, entre los que se incluyen el procesador, los dispositivos de E/S y la memoria principal.

**Estructuras de control del sistema operativo**

Si el sistema operativo va a administrar los procesos y los recursos, entonces tiene que disponer de información sobre el estado actual de cada proceso y de cada recurso. El método universal para obtener esta información es sencillo: El sistema operativo construye y mantiene tablas de información sobre cada entidad que esté administrando.

Las Tablas de memoria se utilizan para seguir la pista de la memoria principal (real) y secundaria (virtual). Parte de la memoria principal está reservada para el uso del sistema operativo; el resto está disponible para el uso de los procesos. Los procesos se mantienen en memoria secundaria mediante alguna *forma* de memoria virtual o por un simple mecanismo de intercambio. Las tablas de memoria deben incluir la información siguiente:

• La asignación de memoria principal a los procesos

• La asignación de memoria secundaria a los procesos

• Cualesquiera atributos de protección de segmentos de memoria principal o virtual, tales como qué procesos pueden acceder a ciertas regiones compartidas de memoria

• Cualquier información necesaria para gestionar la memoria virtual

Las **Tablas de E/S** son utilizadas por el sistema operativo para administrar los dispositivos y los canales de E/S del sistema informático.

El sistema operativo también puede mantener **tablas de archivos**, las cuales ofrecen información sobre la existencia de los archivos, su posición en la memoria secundaria, su estado actual y otros atributos.

el sistema operativo debe mantener tablas de procesos para administrarlos. La memoria, la E/S y los archivos son administrados en nombre de los procesos, por lo que debe haber alguna referencia directa o indirecta a estos recursos en las tablas de procesos. Los archivos que son referidos en las tablas de archivos son accesibles a través de un dispositivo de *E/S* y, algunas veces, estarán en memoria principal o en memoria virtual.

el sistema operativo debe tener algún conocimiento sobre el entorno básico, tal y como cuánta memoria principal hay, cuáles son los dispositivos de E/S y sus identificadores, etc. Este es un asunto de configuración, es decir, cuando se inicializa el sistema operativo, este debe tener acceso a algunos datos de configuración que definan el entorno básico y estos datos deben crearse fuera del sistema operativo, con la asistencia humana.

**Estructuras de control de procesos**

Considérese lo que debe conocer un sistema operativo si tiene que administrar y controlar a los procesos. En primer lugar, debe saber dónde está ubicado el proceso y, en segundo lugar, debe conocer los atributos del proceso que son necesarios para su administración.

*Ubicación de los procesos*

un proceso constará, al menos, de la memoria suficiente para albergar los programas y los datos del proceso. asociado a cada proceso hay una serie de atributos utilizados por el sistema operativo para el control del proceso. Estos atributos se conocen como el bloque de control del proceso.

La ubicación de la imagen de un proceso depende del esquema de gestión de memoria utilizado. la imagen del proceso se guarda como un bloque contiguo de memoria. Este bloque se mantiene en memoria secundaria, normalmente en el disco. Para ejecutar el proceso, la imagen completa debe cargarse en la memoria principal. el sistema operativo necesita conocer la ubicación de cada proceso en el disco y también la ubicación de los procesos que estén en memoria principal. De esta forma, el sistema operativo puede seguir la pista de las partes de la imagen de cada proceso que se quedan en memoria principal.

La mayoría de los sistemas operativos modernos utilizan algún tipo de esquema de gestión de memoria en el que la imagen de un proceso consiste en un conjunto de bloques que no tienen por qué estar almacenados consecutivamente. tales esquemas permiten al sistema operativo tener que traer solo una parte de un proceso en particular. las tablas de procesos deben mostrar la ubicación de cada segmento y/o página de cada imagen de proceso.

**Atributos del proceso**

En un sistema de multiprogramación sofisticado, se requiere una gran cantidad de información de cada proceso para su administración, puede considerarse que esta información reside en un bloque de control del proceso.

Se puede agrupar la información de los bloques de control del proceso en las tres categorías generales siguientes:

• Identificación del proceso

• Información del estado del procesador

• Información de control del proceso

Con respecto a la identificación del proceso, en casi todos los sistemas operativos se le asigna a cada proceso un identificador numérico único. El identificador puede ser tan simple como un Índice en la tabla principal del proceso (véase la figura 3.9). Si no hay identificadores numéricos, entonces debe haber una correspondencia que permita al sistema operativo ubicar las tablas apropiadas a partir del identificador de proceso. Este identificador es útil en varios sentidos.

Además de estos identificadores de proceso, un proceso también puede tener asignado un identificador de usuario que indica quién es el usuario responsable del trabajo.

mientras un proceso está ejecutándose, la información está en los registros. Cuando se interrumpe el proceso, toda la información de los registros debe salvarse de forma que pueda restaurarse cuando el proceso reanude su ejecución.

Los *registros visibles para el usuario* son aquellos accesibles para los programas de usuario y que se usan para almacenar datos temporalmente.

Se emplean varios registros de *control* y de *estado* para controlar la operación del procesador: La mayoría de éstos, en la mayoría de las máquinas, no son visibles para los usuarios. Un registro de control que se encuentra en todos los procesadores es el contador de programa o registro de instrucción, que contiene la dirección de la próxima instrucción que debe leerse.

**Información de control del proceso.** Esta es la información adicional necesaria para que el sistema operativo controle y coordine los diferentes procesos activos.

Cada imagen de proceso consta de un bloque de control de proceso, una pila de usuario, el espacio de direcciones privadas del proceso y cualquier otro espacio de direcciones que comparta con otros procesos.

**El papel del bloque de control del proceso**

El bloque de control de proceso es la estructura de datos central y más importante de un sistema operativo. Los bloques son leídos y/o modificados por casi todos los módulos de un sistema operativo, incluyendo aquellos que tienen que ver con la planificación, la asignación de recursos, el tratamiento de interrupciones y el análisis y supervisión del rendimiento. Puede decirse que el conjunto de los bloques de control de procesos definen el estado del sistema operativo.

La dificultad no está en el acceso, sino más bien en la protección. Existen dos problemas:

.