11

Software del sistema

Contenido

11.1 Introducción 278
11.2 Clasificación del software de sistema 278
11.3 Sistema operativo 278
11.4 Traductores de lenguaje 282
11.5 Resumen 285
11.6 Ejercicios propuestos 286
11.7 Contenido de la página Web de apoyo 288

Objetivos

- Conocer los conceptos básicos relacionados con el software de base.
- Reconocer la relación entre la instrucción software y la capacidad del hardware para decodificarla y lograr su ejecución
- Reconocer, por medio de un lenguaje de programación, la relación lenguaje simbólico y lenguaje de máquina.

En la página Web de apoyo encontrará un breve comentario de la autora sobre este capítulo.

11.1 Introducción

En este capítulo se introducen los primeros conceptos relacionados con el software de sistema. Con el único fin de que se interprete su mención en esta obra, se comentan las actividades comunes de los sistemas operativos organizadas por nivel de administración. En el caso de los procesos, interesa conocer sus cincos estados básicos y la conmutación de uno a otro, para comprender, por ejemplo, qué ocurre cuando un proceso recibe una interrupción o para justificar el uso de la pila.

Además, la administración de memoria, tanto segmentada como paginada, en modo virtual o no, requiere hardware de sustentación, por ejemplo, la unidad de traducción de una dirección virtual a una dirección física. Estos componentes funcionales diseñados en hardware justifican la breve descripción que haremos a continuación acerca del software de sistema. Se destacan los problemas propuestos que permiten relacionar el código de máquina con el lenguaje *Assembler* y los lenguajes de alto nivel.

11.2 Clasificación del software de sistema

El software del sistema es el nexo entre las necesidades del usuario y las capacidades del hardware. Se considera que está integrado por los siguientes componentes:

- Software de base.
- Software de comunicaciones.
- Software de administración de bases de datos.

El sistema operativo administra los recursos, siempre limitados, del sistema.

El software de base controla y respalda en cierto modo el software de las otras categorías, y todas ellas están íntimamente relacionadas en mayor o menor grado con el diseño del hardware, lo que las hace aptas para una computadora y sus compatibles y no para otros. El núcleo del software de base se denomina **sistema operativo**; sus componentes supervisan y controlan la actividad de los recursos físicos (hardware) y los recursos lógicos (usuarios, procesos, archivos, actividades de entrada/salida).

También forma parte del software de base la interfaz gráfica de usuario (GUI o *Graphical User Interface*), cuyo objetivo principal es crear un entorno organizado para el usuario y los utilitarios o utilidades.

Cuando un programador desarrolla un algoritmo para la resolución de un problema con una computadora, éste se debe expresar en términos de las reglas sintácticas y semánticas de un lenguaje de programación. El programador debe atenerse a las reglas de codificación del lenguaje; luego se requiere un procesamiento explícito o no de un programa traductor, por lo general un compilador.

Cada uno de los pasos elementales del algoritmo se representa por medio de una sentencia. Un conjunto de sentencias constituye un programa fuente, que al traducirlo genera un programa ejecutable para el procesador de esa computadora. El conjunto de estos programas conforma el software de aplicación, que utiliza como soporte el entorno que crea el software de base para ejecutarse.

11.3 Sistema operativo

En el diccionario de la Real Academia Española se encontraron varias opciones para definir el verbo administrar. Todas son compatibles con la función del sistema operativo:

- Suministrar, proporcionar o distribuir.

- Graduar o dosificar el uso de algo, para obtener mayor rendimiento de ello o para que produzca mejor efecto.
- Ordenar, disponer, organizar.
- Gobernar, ejercer la autoridad o el mando.

Un sistema operativo es una colección de programas que administran la operación de una (o varias) computadoras, con el fin de obtener un comportamiento eficiente. Es una plataforma software que asigna recursos y supervisa al resto de los programas que se ejecutan en la computadora. El sistema operativo no realiza por sí solo ninguna función que genere resultados para las aplicaciones del usuario, sino que es un medio para que estos programas los obtengan. La asignación de recursos, de hardware o software, que se requieren durante la ejecución puede diferenciarse a partir de las siguientes funciones del sistema operativo:

- Control de tiempos de ejecución en el procesador y sincronización de los procesos.
- Asignación de espacio de memoria requerido para cada proceso.
- Asignación de espacio de almacenamiento y recuperación de archivos (en memorias auxiliares).
- Interfaz con los manejadores de dispositivos de E/S.
- Contabilidad del tiempo de uso.
- Control y recuperación de errores.
- Protección en el uso de recursos.
- Comunicación con el usuario.
- Prevención de errores en el mal uso del sistema.

Su objetivo principal, al ejecutar aplicativos, es crearles un entorno organizado, abastecer sus requerimientos y solucionar los problemas que surjan durante la ejecución.

Los programas que constituyen el sistema operativo son desarrollados por fabricantes de la industria del software y proporcionados a los compradores. Están diseñados para hacer el mejor uso de los componentes de cada sistema de computación. Pueden existir diferentes sistemas operativos pertenecientes al mismo fabricante. En cuanto al usuario, le ofrece comodidad, ya que lo libera de programar tareas rutinarias, y mejora la eficiencia del sistema, sobre todo en lo que se refiere a tratar de minimizar el tiempo ocioso de la CPU.

Se puede afirmar que hay dos grandes componentes software del sistema operativo:

Residentes: también llamados supervisores. Son componentes que residen de manera permanente en la memoria principal durante todo el procesamiento.

Transitorios: residen sólo cuando se los necesita y están almacenados en memorias secundarias cuando no están en la memoria principal.

Se denomina software de aplicación al conjunto de programas que atienden los problemas específicos del usuario. Es concebido o creado por el programador en una empresa o por un fabricante; por ejemplo, el procesador de texto que se utiliza para escribir este capítulo fue diseñado por una empresa que vende productos de software denominados aplicaciones o aplicativos y que sirven para atender trabajos específicos del usuario.

Se espera que el diseño de un sistema operativo maximice la eficiencia global del procesamiento en la computadora. Debe supervisar todas las actividades, ejecutar programas especiales de sistema cuando sea necesario, asignar nuevos recursos y planificar los trabajos para la actividad continua del sistema.

11.3.1 Niveles de administración del sistema operativo

11.3.1.1 Administración del procesador y los procesos

Un proceso es un programa, o parte de él, en estado de ejecución, o sea que está en alguna de las etapas de su ciclo de vida. Está constituido por el código ejecutable, los datos con los que va a operar, el estado de los registros en la CPU y su propio estado respecto de otros procesos.

El despachador es el módulo de sistema operativo que asigna el proceso al procesador, pertenece al nivel de administración más cercano al proceso. Es el que permite la creación de un nuevo proceso aceptando la solicitud de otro o de un usuario. La creación de un proceso implica una serie de actividades que arranca con el reconocimiento del proceso por parte del sistema operativo, que verifica que haya recursos suficientes para su ejecución; por ejemplo, el proceso requiere memoria principal para sus instrucciones y datos. En esta etapa le asigna una identificación, un estado, una prioridad y demás elementos para realizar su seguimiento. En consecuencia, el despachador es el que administra el contexto asociado a un proceso para que pase por los distintos estados de su ciclo de vida A continuación describiremos cinco etapas o estados en los que evoluciona hasta finalizar y salir del sistema.

- Estado nuevo. Se puede decir que un programa ejecutable "adquiere vida" cuando se le crea un entorno adecuado para comenzar su ciclo de vida. Ciclo que finaliza cuando termina su ejecución de manera normal o anormal.
- Estado listo o preparado. Es cuando un proceso tiene recursos asignados suficientes para que la CPU comience o continúe su ejecución, pero no tiene la asignación del procesador.
- Estado de ejecución de un proceso. Se denomina así al estado en el que se está haciendo uso de la CPU.
- Estado en espera. Es aquel en el que el proceso no puede continuar su ejecución porque le falta algún recurso que se le proveerá luego.
- Estado parado. Es aquel en el que el proceso terminó su ejecución y se requiere un tiempo en el que el sistema operativo libere los recursos que le había asignado y destruya toda la información de contexto del proceso para eliminarlo del sistema.

Cuando el proceso P1 en ejecución queda en espera porque, por ejemplo, necesita una operación de E/S, el sistema operativo posibilita que otro proceso P2, que está en estado listo pase a estado de ejecución. Cuando P1 se pone en lista de espera y P2 se ejecuta, se produce la intervención del *dispatcher*. Al finalizar la operación de E/S, P1 podrá pasar de nuevo a estado listo.

11.3.1.2 Administración de memoria

El sistema operativo asigna la memoria que un proceso requiere para su ejecución. También administra la memoria virtual en los sistemas operativos que cuenten con esta facilidad. El concepto de memoria virtual se sigue utilizando en sistemas operativos actuales. Recuerde que si un programa, debido a su tamaño, no puede acomodarse en forma completa en la memoria principal, el sistema operativo puede fragmentarlo y facilitar el intercambio de fragmentos entre memoria principal y memoria de disco. Esto le permite al sistema ampliar la memoria principal utilizando el disco como una extensión de ella.

El control de programas en estado de ejecución fija límites y restricciones entre procesos como, por ejemplo, a qué partes de la memoria puede acceder un programa en ejecución y cuáles no.

11.3.1.3 Administración de dispositivos de entrada/salida

Un dispositivo de entrada/salida requiere comandos o instrucciones específicas. Por ejemplo, un disco debe comenzar a girar antes de que se acceda a él para su lectura o escritura. El lenguaje de comandos del disco es particular y distinto del de la impresora; además, un dis-

positivo genera información de su propio estado. Todos estos detalles deben quedar aislados de las aplicaciones, de modo que el acceso a cada uno de ellos "sea visto" como una simple operación de lectura o escritura.

Encapsular los detalles propios de cada dispositivo es tarea de los manejadores de dispositivo o *drivers*. De este modo, un fabricante de dispositivo escribe su propio *driver* para que sirva de software para el conjunto de sistemas operativos más usados, por lo que el diseñador no requiere desarrollar una nueva versión de cada sistema operativo en cuestión.

La programación de funciones estandarizadas agrupando dispositivos por clases generales corresponde al diseño de un sistema operativo; así se crea una capa de software que actúa de interfaz entre el proceso y la clase de dispositivo. Cuando un proceso requiere una entrada/salida produce una "llamada al sistema" (*system call*) que convoca a la función estándar. Dentro de los servicios brindados por el sistema operativo, debemos mencionar las llamadas al sistema (*system calls*), en este caso de "entrada/salida". Un proceso en curso suspende su ejecución para solicitar un servicio del sistema operativo por medio de estas instrucciones "especiales", que tienen un nivel de privilegio mayor que las restantes y se conocen como primitivas.

Por ejemplo, cuando un proceso de usuario quiere grabar un registro en un archivo en disco, la primitiva invoca la función estándar creada. Esta rutina requiere que el proceso "le pase" parámetros. Si en el disco se detecta una falla que provoca que la grabación no sea posible, la función recibe como parámetro un indicativo del error. Si el error no se produce después de que el sistema operativo ejecutó la llamada al sistema, el proceso puede continuar su ejecución. Por ejemplo, en *GNU/LINUX* la función *ioctl()* permite que el núcleo del sistema operativo se comunique con un *driver*. Esta función requiere un primer argumento que identifique el *driver*, un segundo argumento que indique cuál es la operación y un tercer argumento que establezca dónde queda el resultado de la operación.

11.3.1.4 Administración de archivos

Supervisa la gestión de archivos para su creación, acceso y eliminación. Define la política que determina de qué forma serán almacenados físicamente. Determina la asignación de un archivo a una aplicación, y si corresponde de acuerdo con los derechos de acceso. Informa de su disponibilidad cuando el sistema permite que varias aplicaciones puedan acceder a archivos de manera alternada.

11.3.2 Tipos de sistemas operativos

11.3.2.1 Multitarea y tiempo compartido

Los sistemas operativos de tiempo compartido tratan de administrar los recursos repartiéndolos de manera equitativa. Los sistemas de multitarea son capaces de administrar procesos concurrentes y permiten que tanto las instrucciones como los datos procedentes de varios procesos residan al mismo tiempo en la memoria principal y eventualmente en el disco. Los *task* o tareas activas compiten de manera simultánea por los recursos del sistema en forma alternada.

11.3.2.2 Multiusuario

Los sistemas operativos multiusuario permiten el acceso de varios usuarios desde distintas terminales administradas por el mismo sistema operativo.

11.3.2.3 Tiempo real

Estos sistemas tienen como objetivo proporcionar tiempos más rápidos de respuesta, pues se utilizan para brindar soporte a sistemas como el control de transportes, el control de pro-

cesos industriales, los cajeros automáticos, por dar algunos ejemplos. La característica más importante de estos sistemas es que sus acciones se deben ejecutar en intervalos de tiempo determinados por la dinámica de los sistemas físicos que supervisan o controlan; además, deben ser sistemas rigurosos en cuanto a la integridad de la información, asegurar un servicio sin interrupciones y que soporte políticas de seguridad eficientes.

11.4 Traductores de lenguaje

Son programas cuya función es convertir los programas escritos por el usuario en lenguaje simbólico a lenguaje de máquina. Un programa escrito en lenguaje simbólico se denomina fuente, mientras que un programa en lenguaje de máquina se denomina ejecutable. Las instrucciones del programa fuente constituyen la entrada al traductor de lenguaje, la salida serán las instrucciones del programa ejecutable. La traducción comprende el análisis del léxico y la sintaxis de cada instrucción o sentencia, así como las referencias lógicas a las que apunta cada instrucción. Si del análisis surgieran errores, el traductor generará un informe donde indicará el lugar donde se produjo y cuál es el tipo de error cometido. Para realizar el análisis sintáctico, el programa traductor controla cada sentencia del programa fuente (nombres de datos, signos de puntuación y palabras propias del lenguaje para que tengan una construcción aceptada por el traductor). Se necesita un traductor para cada lenguaje simbólico que se utilice. Se destacan tres tipos de traductores de lenguaje: ensambladores, intérpretes y compiladores.

11.4.1 Ensambladores

El término ensamblador (*assembler*) se refiere a un tipo de software traductor que se encarga de traducir un archivo fuente escrito en un lenguaje Assembler a un archivo cuyas instrucciones estén en código máquina, que es ejecutable directamente por el procesador para el que se creó.

La razón por la que se creó este tipo de traductor es "humanizar", por medio de un lenguaje simbólico, la escritura de programas a nivel código de máquina, que es el código binario "decodificable" por el procesador pero imposible de escribir en la práctica. Así, el código *8E26* se representa en Assembler *x86* como *MOV BH* en el Assembler, donde *MOV* es mover y *BH* es el nombre de un registro de CPU.

Este tipo de traductor evolucionó como todos los lenguajes de programación y en la actualidad dejó de llamarse ensamblador para identificarse como compilador *Assembler*.

Existe la creencia errónea de que el Assembler es un lenguaje "antiguo"; esto se debe a que la mayoría de los programadores trabaja con lenguajes más evolucionados, como aquellos muy utilizados en el presente que son los visuales. Sin embargo, a la hora de programar a nivel hardware, la única posibilidad la ofrece el set de instrucciones Assembler del procesador que se esté utilizando. Todo procesador, grande o pequeño, desde el de una computadora personal hasta el de una supercomputadora, posee su lenguaje máquina y, por lo tanto, su simbólico de máquina.

Los traductores se dividen en dos grupos en función de la relación entre lenguaje fuente y lenguaje de máquina. El primer grupo traduce una instrucción de un lenguaje fuente y genera una única instrucción de máquina; ese lenguaje fuente es su Assembler. El segundo grupo lo constituyen los lenguajes de alto nivel en los que una sentencia se traduce a varias instrucciones en código de máquina.

Como en todo Assembler hay en mayor medida una correspondencia *1* a *1* entre instrucciones simbólicas e instrucciones de máquina y, por ello, se puede realizar la traducción inversa, denominada *desensamble*.