ESCUELA DE PREPARACIÓN DE OPOSITORES
E. P. O.
C/. La Merced, 8 – Bajo A Telf.: 968 24 85 54
30001 MURCIA

INF15 – SAI15

Sistemas operativos. Componentes. Estructura. Funciones. Tipos.

Esquema.

1 Introducción.

Sin el software, una computadora es en esencia una masa metálica sin utilidad. Con el software, una computadora puede almacenar, procesar y recuperar información. El software para computadoras puede clasificarse en general en dos clases: los programas de sistema, que controlan la operación de la computadora en sí y los programas de aplicación, los cuales resuelven problemas para sus usuarios. El programa fundamental de todos los programas de sistema es el sistema operativo, que controla todos los recursos de la computadora y proporciona la base sobre la cual pueden escribirse los programas de aplicación.

Un sistema de computación moderno consta de uno o más procesadores, cierta memoria principal, relojes, terminales, discos, interfaces de red y otros dispositivos de

entrada/salida. En fin, un sistema complejo. La escritura de programas que mantengan un registro de todos estos componentes y los utilice en forma correcta, ya no digamos en forma óptima, es una labor difícil. Si cada programador tuviera que preocuparse por la forma de funcionamiento de las unidades de disco y con las docenas de cosas que podrían ir mal al leer un bloque de un disco, es poco probable que pudieran escribirse muchos programas.

Hace muchos años, quedó claro que debía determinarse una forma de proteger a los programadores de la complejidad del hardware. La forma en que esto ha evolucionado de manera gradual es colocando un nivel de software por encima del simple hardware, con el fin de controlar todas las partes del sistema y presentar al usuario una interfaz o máquina virtual que facilite la comprensión del programa. Este. nivel de software es el sistema operativo.

En la siguiente figura se muestra la situación. En la parte inferior se encuentra el hardware, el cual consta, en la mayoría de los casos, de varias capas. La capa más baja contiene los dispositivos físicos. La forma de construirlos y su funcionamiento son terreno del ingeniero eléctrico.

A continuación viene un software primitivo que controla de forma directa los dispositivos y proporciona una interfaz más limpia con la siguiente capa. Este software, llamado microprograma, se localiza, por lo general, en la memoria exclusiva para lectura. Es en realidad un intérprete, busca las instrucciones de lenguaje máquina para llevarlas a cabo como una serie de pequeños pasos. El conjunto de instrucciones que interpreta el microprograma define el lenguaje máquina, que no es en sí parte de la máquina pero todos los fabricantes de computadoras lo describen de esa forma, por lo que muchas personas piensan en él como si fuera la "máquina" real. En ciertas máquinas, el microprograma se implanta en el hardware y no es en realidad una capa distinta.

El lenguaje máquina tiene, por lo general, de 50 a 300 instrucciones, la mayoría de las cuales sirve para desplazar datos a través de la máquina, hacer operaciones aritméticas y comparar valores. En esta capa, los dispositivos de entrada/salida se controlan al cargar valores en registros de dispositivo especiales. Por ejemplo, se puede ordenar un disco que lea al cargar los valores de la dirección del disco, la dirección de la memoria principal, el byte de conteo y la instrucción (READ o WRITE) en sus registros. En la práctica, son necesarios muchos más parámetros y el estado que regresa

la unidad después de una operación es muy complejo. Además, el tiempo juega un papel importe en la programación de la mayoría de los dispositivos de entrada/salida.

Una de las principales funciones del sistema operativo es ocultar toda esta complejidad y proporcionar al programador un conjunto más conveniente de instrucciones con el cual trabajar. Por ejemplo, READ BLOCK FROM FILE es conceptualmente más sencillo que tener que preocuparse de los detalles de movimiento de las cabezas del disco, esperar que éstas se detengan, etc.

Por encima del sistema operativo está el resto del software de sistema. Aquí se encuentran el intérprete de comandos (shell), compiladores, editores y programas de aplicación independientes similares a ellos. Es importante observar que estos programas no son parte del sistema operativo, aunque sea usual que el fabricante de la computadora los proporcione junto con él. Este es un punto sutil, pero crucial. El sistema operativo es esa parte del software que se ejecuta en modo central o modo de supervisión. Está protegido contra la alteración por parte del usuario por el hardware (sin considerar, en este momento, algunos de los antiguos microprocesadores, que no tenían protección alguna del hardware). Los compiladores y editores se ejecutan en modo usuario. Si un usuario no desea utilizar un compilador particular, tiene libertad de escribir el suyo propio, si así lo desea; no tiene la libertad de escribir su propio controlador de interrupciones del disco, que es parte del sistema operativo y que, por lo general, queda protegido por el hardware contra los intentos de escritura en él por parte del usuario.

Por último, sobre los programas de sistema se encuentran los programas de aplicación. Estos programas son escritos por los usuarios para resolver sus problemas particulares, como el procesamiento de datos comerciales, cálculos de ingeniería o juegos.

2 Funciones de los sistemas operativos.

La mayoría de los usuarios de computadora tienen cierta experiencia con un sistema operativo, pero es difícil poder precisar la definición de éste. Parte del problema es que los sistemas operativos llevan a cabo, dos funciones que, en esencia, no tienen relación entre sí.

2.1 El sistema operativo como una máquina extendida.

La arquitectura a nivel del lenguaje de máquina (conjunto de instrucciones, organización de la memoria, E/S y estructura del bus) de la mayoría de las computadoras es primitiva y difícil de programar, particularmente en la entrada/salida.

El programador desea una abstracción sencilla y de alto nivel con la cual trabajar. En el caso de los discos, una abstracción típica es que el disco contenga una colección de archivos, cada uno de ellos con un nombre. Cada archivo puede ser abierto para la lectura o escritura, después se lee o se escribe y, por último, se cierra. Los detalles de si el registro debe utilizar la modulación de frecuencia modificada o de cuál es el estado actual del motor no deben aparecer en la abstracción presentada al usuario.

El programa que oculta la verdad acerca del hardware al programador y presenta una agradable y sencilla visión de los archivos con su nombre, los cuales se puedan leer o escribir en ellos es, por supuesto, el sistema operativo. Así como el sistema operativo protege al programador del hardware del disco y presenta una sencilla interfaz orientada a archivos, también oculta varios asuntos poco agradables relacionados con las

interrupciones, cronómetros, control de la memoria y otras características de bajo nivel. En cada caso, la abstracción que se presenta al usuario del sistema operativo es mucho más simple y fácil de utilizar que el hardware subyacente.

En esta perspectiva, la función del sistema operativo es presentar al usuario el equivalente de una máquina extendida o máquina virtual que sea más fácil de programar que el hardware subyacente.

2.2 El sistema operativo como controlador de recursos.

El concepto del sistema operativo como algo que en primer lugar proporciona a sus usuarios una interfaz conveniente entra en una visión de abajo hacia arriba. Un punto de vista alternativo, de arriba hacia abajo, sostiene que el sistema operativo está ahí para controlar todas las piezas de un complejo sistema. Las computadoras modernas constan de procesadores, memorias, cronómetros, discos, terminales, unidades de cinta magnética, interfaces de red, impresoras láser y una amplia gama de otros dispositivos. Desde este punto de vista, la labor del sistema operativo es la de proporcionar una asignación ordenada y controlada de los procesadores, memorias y dispositivos de E/S para los varios programas que compiten por ellos.

Imaginemos lo que ocurriría si tres programas que están en ejecución en cierta computadora intentaran imprimir su salida de forma simultánea en la misma impresora. Los primeros renglones de impresión serían del programa 1, los siguientes del programa 2, luego algunos del programa 3, etc. El resultado sería un caos. El sistema operativo puede poner orden en el caos potencial, al almacenar en el disco toda la salida destinada a la impresora. Al concluir uno de los programas, el sistema operativo podrá entonces copiar su salida desde el disco de archivo donde se encontraba hacia la impresora, mientras el otro programa continuaría generando más salida, sin importar el hecho de que la salida no sea dirigida a la impresora (todavía).

Si una computadora tiene varios usuarios, es todavía más evidente la necesidad del control y protección de la memoria, los dispositivos de E/S y demás recursos. Esta necesidad surge del hecho de que, con frecuencia, los usuarios deben compartir costosos recursos. Sin señalar el aspecto económico, también se requiere a menudo compartir la información entre aquellos usuarios que trabajan juntos. En resumen, este punto de vista del sistema operativo sostiene que su principal tarea es la de llevar un registro de la utilización de los recursos, dar paso a las solicitudes de recursos, llevar la cuenta de su uso y mediar entre las solicitudes en conflicto de los distintos programas y usuarios.

3 Conceptos de los sistemas operativos.

La interfaz entre el sistema operativo y los programas del usuario se define como el conjunto de "instrucciones ampliadas" que proporciona el sistema operativo. Estas instrucciones ampliadas se conocen como llamadas al sistema. Las llamadas al sistema crean, eliminan y utilizan varios objetos del software, controlados por el sistema operativo. Los más importantes son los procesos y archivos.

3.1 Procesos.

Un concepto central en todos los sistemas operativos es el de proceso. Un proceso es básicamente un programa en ejecución. Consta del programa ejecutable, sus datos y pila, contador y otros registros, además de toda la información necesaria para ejecutar el programa.

La forma más sencilla para tener una idea intuitiva de un proceso es pensar en los sistemas de tiempo compartido. Periódicamente, el sistema operativo decide detener la ejecución de un proceso y comenzar la ejecución de otro; por ejemplo, si el primero de ellos ha utilizado en el último segundo una porción de tiempo de la CPU mayor de la permitida.

Cuando un proceso se detiene de forma temporal, como en el ejemplo, éste debe volverse a inicializar en el mismo estado en que se encontraba al detenerse. Esto quiere decir que toda la información relativa al proceso debe almacenarse de forma explícita en alguna parte durante la suspensión. Por ejemplo, si el proceso abrió varios archivos, debe registrarse en algún lugar la posición exacta del proceso en los archivos de modo que una instrucción READ dada después de reiniciado el proceso lea los datos adecuados. En muchos sistemas operativos, toda la información relativa a un proceso, distinta del contenido de su propio espacio de dirección, se almacena en una tabla del sistema operativo llamada tabla de procesos, la cual consta de un array (o lista enlazada) de estructuras, una por cada proceso existente en ese momento.

Así, un proceso (suspendido) consta de su espacio de dirección, llamado imagen central y los datos de su tabla de procesos, que entre otras cosas contiene sus registros.

Las llamadas al sistema de control de procesos fundamentales son las que se ocupan de la creación y fin de los procesos. Un proceso llamado intérprete de comando o shell lee los comandos a partir de un terminal. El usuario acaba de escribir un comando que solicita la compilación de un programa. El shell debe crear entonces un nuevo proceso que ejecutará el compilador. Cuando ese proceso ha concluido la compilación, ejecuta una llamada al sistema para terminarlo.

Si un proceso puede crear uno o más procesos (conocidos como procesos hijo) y estos procesos pueden crear a su vez procesos hijo, llegaremos a una estructura de árbol.

Se dispone de otros tipos de llamadas a proceso para solicitar más memoria (o liberar la memoria no utilizada), esperar a que un proceso hijo termine o bien cambiar el programa por otro distinto.

En ciertas ocasiones, existe la necesidad de comunicar información a un proceso en ejecución de forma que éste no quede en espera de ella. Por ejemplo, un proceso que se comunica con otro en una computadora distinta lo hace mediante el envío de mensajes en una red. Para protegerse de la posibilidad de que un mensaje o su réplica se pierdan, el emisor puede solicitar que su propio sistema operativo lo notifique después de un número determinado de segundos, de forma que pueda retransmitir el mensaje si no ha recibido una confirmación todavía. Después de establecer este cronómetro, el programa puede continuar con otra tarea.

Al transcurrir ese número determinado de segundos, el sistema operativo envía una señal al proceso. La señal hace que el proceso se suspenda de forma temporal sin importar lo que haga, guarda sus registros en la pila y comienza a ejecutar un procedimiento especial de control de la señal; por ejemplo, para retransmitir un mensaje que podría estar perdido. Cuando dicho procedimiento termina su labor, el proceso en ejecución continúa a partir del estado en que se encontraba justo antes de la señal. Las señales son el análogo en software de las interrupciones de hardware y pueden ser generadas por una variedad de causas, además de la expiración de los cronómetros. Muchas de las trampas detectadas por el hardware, como la ejecución de una instrucción ilegal o uso de una dirección inválida, se convierten también en señales hacia el proceso

culpable. Las señales se utilizan también para la comunicación entre procesos, cuando un proceso desea comunicar algo urgente a otro.

En un sistema de multiprogramación, es importante mantener en un registro a qué usuario pertenece cada proceso. En dicho sistema, cada usuario autorizado tiene asignada una *uid* (identificación del usuario), que, por lo general, es un entero de 16 ó 32 bits. Cada proceso tiene asignada la clave de un usuario. Cuando el proceso envía una señal a otro, se debe hacer una verificación para ver si el emisor y el receptor tienen la misma identificación del usuario. De forma similar, las personas se pueden dividir en grupos (equipos de proyecto, departamentos, etc.) cada uno de ellos con su *gid* (identificación de grupo). Ambas identificaciones juegan un papel en la protección de la información contenida en la computadora.

3.2 Archivos.

La otra gran categoría de llamadas al sistema se relaciona con el sistema de archivos. Como ya mencionamos, una de las funciones principales del sistema operativo es ocultar las peculiaridades de los discos y demás dispositivos de E/S, para presentar al programador un modelo agradable y nítido de archivos independientes de los dispositivos. Es evidente la necesidad de las llamadas al sistema en la creación, eliminación, lectura y escritura de archivos. Antes de poder leer un archivo, hay que abrir éste; después de leer un archivo, éste debe cerrarse; las llamadas permiten hacer todo esto.

Para poder proporcionar un espacio donde almacenar los archivos, la mayoría de los sistemas operativos soportan el concepto de directorio como una forma de agrupar los archivos. Por ejemplo, un estudiante puede disponer de un directorio por cada curso que tome (para los programas necesarios en cada curso), otro directorio para el correo electrónico y otro directorio más para los juegos de computadora. En esa situación, las llamadas al sistema son necesarias para crear y eliminar directorios. Las llamadas también permiten colocar un archivo existente en un directorio y eliminar un archivo de un directorio. Los datos de un directorio pueden ser archivos u otros directorios. Este modelo también da lugar a una jerarquía: el sistema de archivos, como se muestra en la siguiente figura.

El proceso y las jerarquías de archivos se organizan como árboles, pero sólo son similares hasta ahí. Las jerarquías de los procesos no son, en general, muy profundas, mientras que las jerarquías de archivos tienen por lo general, muchos niveles. Las jerarquías de los procesos tienen usualmente una vida corta, de unos cuantos minutos, mientras que la jerarquía del directorio puede durar años. La propiedad y la protección también son distintas en los procesos y los archivos. Lo común es que un proceso padre pueda controlar o incluso tener acceso a un proceso hijo, pero casi siempre, los mecanismos permiten que los archivos y directorios sean leídos por un grupo más amplio que los propietarios.

Cada uno de los archivos que se encuentran dentro de la jerarquía del directorio puede determinarse mediante el nombre de la ruta de acceso desde la parte superior de dicha jerarquía, el directorio raíz. Tales nombres absolutos de la ruta de acceso constan de la lista de directorios que hay que recorrer desde el directorio raíz hasta llegar al archivo, con diagonales entre los componentes. En la figura anterior, la ruta de acceso para el archivo *CS101* es */Academia/Prof.Brown/Cursos/CS101*. La primera diagonal indica que la ruta es absoluta; es decir, que comienza con el directorio raíz.

En cada momento, cada proceso tiene activo un directorio de trabajo, en el que buscan aquellos nombres de las rutas de acceso que no comiencen con una diagonal. En la figura anterior, si */Academia/Prof.Brown* fuera el directorio activo, entonces el uso del nombre de la ruta de acceso *Cursos/CS101* daría como resultado el mismo archivo del ejemplo anterior. Los procesos pueden modificar su directorio de trabajo mediante una llamada al sistema que especifique el nuevo directorio de trabajo. Como punto colateral, ciertos sistemas utilizan \ en vez de / como separador en los nombres de las rutas de acceso.

Si varias personas tienen acceso a la misma computadora, es importante tener un medio de protección de la privacía de los archivos de cada persona. Distintos sistemas tienen distintos esquemas. Por ejemplo, en UNIX, los archivos y directorios se protegen al asignarles un código de protección binario de 9 bits. El código de protección consta de campos de 3 bits, uno para el propietario, otro para los demás miembros del grupo del propietario (el administrador del sistema divide a los usuarios en grupos) y otro para las demás personas. Cada campo tiene un bit para acceso a la lectura del archivo, otro para la escritura en el archivo y otro para la ejecución del archivo. Estos tres bits se conocen como los bits *rwx* (*Read*, *Write* y *eXecute*). Por ejemplo, el código de protección *rwxr-x--x* indica que el propietario puede leer, escribir en o ejecutar el archivo; los otros miembros del grupo pueden leer o ejecutar el archivo (pero no escribir en él) y las demás personas pueden ejecutar el archivo (pero no leer o escribir). En un directorio, *x* indica el permiso para la búsqueda. Un guión indica que no se dispone del permiso correspondiente.

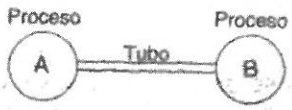
Antes de que un archivo pueda leerse o escribir en él, debe abrirse, momento en el cual se verifican los permisos. Si se permite el acceso, el sistema devuelve un entero pequeño, llamado descriptor del archivo, para usarlo en las operaciones subsecuentes. Si se prohíbe el acceso, se devuelve un código de error.

Muchos sistemas operativos, entre los cuales están MS-DOS y UNIX, proporcionan una abstracción para que los usuarios lleven a cabo la entrada/salida sin tener que preocuparse por todos los detalles del hardware. Esta abstracción representa a

cada dispositivo de E/S como un archivo especial. Los archivos especiales aparecen como una forma de hacer que los dispositivos de E/S se vean como archivos. De esta forma, se puede escribir y leer en ellos mediante las mismas llamadas al sistema utilizadas en la lectura y escritura de archivos. Existen dos tipos de archivos especiales: archivos especiales de bloque y archivos especiales de carácter. Los archivos especiales de bloque se utilizan para modelar los dispositivos que constan de un conjunto de bloques de direccionamiento aleatorio, como los discos. Al leer un archivo especial de bloque y leer, digamos, el bloque 4, el programa puede tener un acceso directo al cuarto bloque del dispositivo, sin importar la estructura del sistema de archivos contenido en él. Los programas que realizan labores de mantenimiento necesitan a menudo esta característica. El acceso a los archivos especiales queda controlado mediante el mismo mecanismo de protección que se utiliza para proteger los demás archivos, de forma que, por ejemplo, el poder de acceso directo a los dispositivos de E/S puede quedar restringido al administrador del sistema. Por razones de seguridad, con frecuencia es necesario un acceso restringido a dispositivos de E/S tales como el disco crudo y la red.

Los archivos especiales de red se utilizan para modelar los dispositivos que constan de flujos de caracteres, en vez de bloques con direccionamiento aleatorio de tamaño fijo. Los terminales, impresoras en línea e interfaces de red son ejemplos de dispositivos especiales de carácter. La forma normal de lectura y escritura de un programa en la terminal del usuario es la lectura o escritura del archivo especial de caracteres correspondiente. En UNIX y MS-DOS, al iniciar un proceso, se establece que el descriptor de archivo 0, llamado entrada estándar, indica el terminal para fines de lectura. El descriptor 1, llamado salida estándar, indica la terminal para fines de escritura. El descriptor de archivo 2, llamado error estándar, también indica la terminal para la salida, pero, por lo general, sólo se utiliza para escribir mensajes de error.

La última característica que analizaremos en esta visión general es aquella que se refiere tanto a los procesos como a los archivos: las tuberías. Una tubería es una especie de pseudoarchivo que se puede utilizar para conectar dos procesos, como se muestra en la siguiente figura.



Cuando el proceso A desea enviar datos al proceso B, escribe en la tubería como si fuera un archivo de salida. El proceso B puede leer los datos, al leer de la tubería como si fuera un archivo de entrada. Así, la comunicación entre procesos se ve como la lectura y escritura ordinaria en archivos. Más aún, la única forma en que un proceso, puede descubrir que el archivo de salida en el que escribe no es en realidad un archivo, sino una tubería, es mediante una llamada especial al sistema. Tanto MS-DOS como UNIX soportan las tuberías.

3.3 Llamadas al sistema.

Los programas del usuario se comunican con el sistema operativo y le solicitan servicio mediante las llamadas al sistema. A cada una de estas llamadas le corresponde un procedimiento de la biblioteca que pueden llamar los programas del usuario. Este procedimiento pone los parámetros de la llamada al sistema en un lugar específico, como pueden ser los registros de la máquina, para después ejecutar una instrucción TRAP (un tipo de llamada a procedimiento protegido) para iniciar el sistema operativo.

La finalidad del procedimiento de biblioteca es ocultar los detalles de la instrucción TRAP y hacer que las llamadas al sistema parezcan llamadas comunes a un procedimiento.

Cuando el sistema operativo recupera el control después de TRAP, examina los parámetros para ver si son válidos; en ese caso, desarrolla el trabajo solicitado. Al terminar, el sistema operativo coloca un código de estado en un registro, para indicar si tuvo éxito o fracaso; además, ejecuta una instrucción RETURN FROM TRAP, para regresar el control al procedimiento de biblioteca. Este procedimiento regresa entonces a quien hizo la llamada y regresa el código de estado como un valor de función. A veces, dentro de los parámetros regresan valores adicionales.

Para poner en claro el mecanismo de la llamada a función, revisemos un ejemplo sencillo, la llamada al sistema READ que se utiliza en UNIX y MS-DOS. Tiene tres parámetros: el primero es el archivo por leer, el segundo determina un almacén (buffer) en el cual colocar los datos del archivo y el tercero indica el número de bytes por leer. Una llamada a READ desde un programa en C se vería como: *count = read(file, buffer, nbytes);*. Existe una sutil diferencia entre el procedimiento de biblioteca que puede llamar el usuario, *read* y la verdadera llamada al sistema, READ, llamada por *read.*

El efecto de llamar al procedimiento *read* (la llamada al sistema) consiste en que los datos del archivo dado pasen al almacén (buffer) donde el programa pueda tener acceso a ellos. El procedimiento regresa el número de bytes leídos de *count*. Este valor es, por lo general, igual a *nbytes*, pero también puede ser menor si, por ejemplo, se encuentra el final de archivo durante la lectura.

Si la llamada al sistema no puede llevarse a cabo, ya sea debido a un parámetro inválido o a un error del disco, *count* se hace igual a -1 y el número de error se pone en una variable global, donde el programa pueda inspeccionarla. Los programas deben siempre verificar los resultados de las llamadas al sistema para ver si no ocurren interrupciones.

El número y tipo de llamadas al sistema varía de un sistema operativo a otro. Por lo general, hay llamadas al sistema para crear procesos, controlar la memoria, leer y escribir y hacer labores de entrada/salida, como por ejemplo, lectura de una terminal o impresión mediante una a impresora.

3.4 El Shell.

El sistema operativo es el código que lleva a cabo las llamadas al sistema. Los editores, compiladores, ensambladores, enlazadores e intérpretes de comandos no son parte del sistema operativo, aunque sean importantes y útiles. Con el riesgo de confundir un poco el asunto, analizaremos en esta sección al intérprete de comandos de UNIX, llamado Shell, el cual, sin ser parte del sistema operativo, hace un uso extenso de muchas de las características del sistema, por lo que sirve como un buen ejemplo del uso que puede darse a las llamadas al sistema. También es la interfaz principal entre un usuario sentado frente a un terminal y el sistema operativo.

Cuando un usuario inicia una sesión, el shell se inicializa. Shell reconoce al terminal como su entrada y salida estándar. Comienza mostrando el indicador, un carácter tal como un signo dólar, que indica al usuario que Shell aguarda para aceptar un comando. Si el usuario escribe entonces *date* por ejemplo, Shell crea un proceso hijo y ejecuta el programa *date* como hijo. Al ejecutarse el proceso hijo, Shell espera hasta que

éste termine. Al finalizar el hijo, Shell exhibe de nuevo el símbolo de comando e intenta leer la siguiente línea de entrada.

El usuario puede determinar que la salida estándar cambie de dirección a un archivo al escribir, por ejemplo *date > file*. De forma análoga, la entrada estándar también puede cambiar de dirección, como en *sort < file1 > file2*, lo que llama al programa *sort* con una entrada tomada de *file1* y salida enviada a *file2*.

La salida de un programa se puede utilizar como la entrada de otro, si éstos se conectan mediante una tubería. Así, *cat file1 file2 file3 | sort > /dev/lp &* llama al programa *cat* para que concatene tres archivos y envíe la salida a *sort* para ordenar todas las líneas de forma alfabética. La salida de *sort* se dirige hacia el archivo */dev/lp*, que es un nombre común para el archivo especial de carácter asociado a la impresora de línea. (Es una convención que todos los archivos especiales se mantengan en el directorio */dev*, al menos en los sistemas UNIX).

Si un usuario utiliza un signo & después de un comando, Shell no espera a que este concluya, sino que proporciona de forma inmediata un indicador. Así, *cat file1 file2 file3 | sort > /dev/lp &* inicia el ordenamiento como una tarea asíncrona, lo que permite al usuario continuar su trabajo de forma normal mientras dicho ordenamiento está en proceso.

4 Componentes.

Por lo general, los sistemas operativos tienen cuatro componentes principales: control de procesos, administración de la memoria, control de archivos y control de los dispositivos de E/S.

4.1 Control de procesos.

El concepto central de cualquier sistema operativo es el proceso, una abstracción de un programa en ejecución. Todas las computadoras modernas hacen varias cosas al mismo tiempo. A la vez que ejecuta un programa del usuario una computadora puede leer de un disco e imprimir en una terminal o impresora. En un sistema multiprogramación, la CPU también alterna de programa en programa, ejecutando cada uno de ellos por decenas o cientos de milisegundos. Aunque, en sentido estricto, la CPU ejecuta en cierto instante un solo programa, durante un segundo puede trabajar con varios de ellos, lo que da una apariencia de paralelismo.

La interacción entre procesos puede derivar en condiciones de competencia, situaciones en las que la perfecta sincronización determina el resultado. Para evitar las condiciones de competencia, se introdujo el concepto de sección crítica, que es una sección de código en la que un proceso realiza algo con el estado compartido y no desea que otros procesos trabajen ahí también. Las secciones críticas proporcionan la exclusión mutua. Existen varias primitivas de la comunicación entre procesos. Entre éstas están los semáforos y la transferencia de mensajes.

Para implementar la alternancia entre los procesos es necesario que un planificador seleccione el proceso que debe ejecutar la CPU. Se conocen muchos algoritmos de planificación. La labor de un algoritmo de planificación es determinar el proceso que debe ejecutarse a continuación, tomando en cuenta factores tales como el tiempo de respuesta, la eficacia y la equidad. Entre los algoritmos de planificación más conocidos están el round robin, planificación por prioridades, colas de varios niveles, primero el trabajo más corto y la planificación garantizada. En algunos sistemas, el

mecanismo de planificación y la política de planificación se dividen, lo que permite una mayor flexibilidad.

4.2 Administración de la memoria.

La parte del sistema operativo que administra la memoria se llama administrador de memoria. Su labor consiste en llevar un registro de las partes de memoria que se estén utilizando y aquéllas que no, con el fin de asignar espacio en memoria a los procesos cuando éstos la necesiten y liberarlo cuando terminen, así como administrar el intercambio entre la memoria principal y el disco, en los casos en que la memoria principal no pueda albergar a todos los procesos.

Los sistemas más sencillos no realizan intercambio alguno. Una vez que se carga un programa en memoria, permanece ahí hasta terminar. Algunos sistemas operativos sólo permiten que, en cada momento, exista un único proceso en la memoria, a la vez que otros permiten la multiprogramación.

El siguiente paso es el intercambio. Cuando se utiliza, el sistema puede controlar más procesos de los que caben en la memoria. Los procesos para los que no existe espacio se intercambian con el disco. Se puede llevar un registro del espacio libre en memoria y en disco mediante un mapa de bits, una lista de huecos o un sistema asociado.

Con frecuencia, las computadoras más avanzadas tienen cierta forma de memoria virtual. En su forma más simple, el espacio de direcciones de cada proceso se divide en bloques de tamaño uniforme llamados páginas, las cuales se pueden colocar dentro de cualquier marco para página disponible en memoria. Cuando las tablas de páginas son muy grandes, se puede utilizar un esquema de paginación con varios niveles para que las páginas se paginen a sí mismas.

Para mejorar el rendimiento, casi todas las computadoras que soportan la paginación tienen una memoria asociativa para una relación rápida del número de página virtual con el número del marco físico. La tabla de páginas se consulta sólo en caso de algún faltante.

Se conocen muchos algoritmos para el reemplazo de páginas. Dos de los mejores y que son realizables son el del reloj y el de maduración. Los algoritmos de paginación se pueden modelar en forma teórica mediante el concepto de distancias de la cadena, el cual permite hacer ciertas predicciones.

Una alternativa a la paginación pura es la segmentación, con o sin segmentos paginados.

4.3 Control de archivos.

Todas las aplicaciones necesitan almacenar y recuperar la información. Tenemos tres condiciones esenciales para el almacenamiento de la información a largo plazo: debe ser posible almacenar una cantidad muy grande de información, la información debe sobrevivir a la conclusión del proceso que la utiliza, y debe ser posible que varios procesos tengan acceso concurrente a la información. La solución a estos problemas es el almacenamiento de la información en discos y otros medios externos, en unidades llamadas archivos. La información almacenada en los archivos debe ser persistente; es decir, no debe verse afectada por la creación y término de un proceso. Un archivo debe desaparecer sólo en caso de su eliminación explícita por parte del propietario.

Los archivos son administrados por el sistema operativo. Su estructura, nombre, forma de acceso, uso, protección e implantación son temas fundamentales en el diseño de un sistema operativo. Aquella parte del sistema operativo que trabaja con los archivos se conoce como el sistema de archivos.

Los implementadores del sistema de archivos deben preocuparse por llevar un registro de los bloques en disco correspondientes a cada archivo, la forma de compartir los archivos y la forma de administrar el espacio libre en disco. Los directorios se pueden ordenar de varias formas, en un rango que va desde colocar el nombre, atributos y direcciones en disco hasta sólo colocar en ellos el nombre y un número de nodo-i. Los archivos compartidos, el manejo de los bloque defectuosos, los respaldos, la consistencia y el caché también son temas importantes.

4.4 Control de los dispositivos de E/S.

Una de las funciones principales de un sistema operativo es el control de todos los dispositivos de E/S de la computadora. Debe enviar comandos a los dispositivos, detectar las interrupciones y controlar los errores. También debe proporcionar una interfaz entre los dispositivos y el resto del sistema, sencilla y fácil de usar. En la medida de lo posible, la interfaz debe ser la misma para todos los dispositivos (independencia del dispositivo).

El software de E/S podemos dividirlo en cuatro niveles: las rutinas de interrupción, los manejadores de dispositivos, el software independiente de los dispositivos y las bibliotecas y spoolers de E/S que se ejecutan en el espacio del usuario. Las rutinas de interrupción guardan el estado de la máquina, después dan servicio al dispositivo y verifican los errores. En general, es recomendable utilizarlas lo menos posible, de manera consistente con el rendimiento requerido. Los manejadores de dispositivos controlan todos los detalles específicos de uno o más dispositivos. Su labor es ocultar los detalles engorrosos a los niveles superiores. El software independiente del dispositivo realiza cosas como el almacenamiento en buffers y la asignación. Este software es el mismo para varios dispositivos.

Los tres tipos principales de dispositivos son: discos, relojes y terminales. Para los discos, el algoritmo principal es la forma de planificar el movimiento del brazo. Para los relojes, lo importante es mantener un registro de varias solicitudes con un solo reloj. Para el caso de los terminales hay que distinguir entre la entrada y la salida. Los aspectos de la entrada son el modo crudo o el modo cocinado, el almacenamiento en buffers, el eco, los caracteres de relleno, la edición de líneas y el manejo de caracteres especiales. El software de salida se encarga del movimiento del cursor y otras secuencias de escape, entre otros aspectos.

5 Estructura de los sistemas operativos.

En las secciones siguientes, examinaremos cinco estructuras distintas, con el fin de tener una idea del espectro de posibilidades.

5.1 Sistemas monolíticos.

Este tipo de organización es con mucho la más común. La estructura consiste en que no existe estructura alguna. El sistema operativo se escribe como una colección de procedimientos, cada uno de los cuales puede llamar a los demás cada vez que así lo requiera. Cuando se usa esta técnica, cada procedimiento del sistema tiene una interfaz

bien definida en términos de parámetros y resultados y cada uno de ellos es libre de llamar a cualquier otro, si éste último proporciona cierto cálculo útil para el primero.

Para construir el programa objeto real del sistema operativo mediante este punto de vista, uno compila de forma individual los procedimientos o los archivos que contienen los procedimientos y después los enlaza en un solo archivo objeto. En términos del ocultamiento de la información, ésta es prácticamente nula: cada procedimiento es visible a los demás.

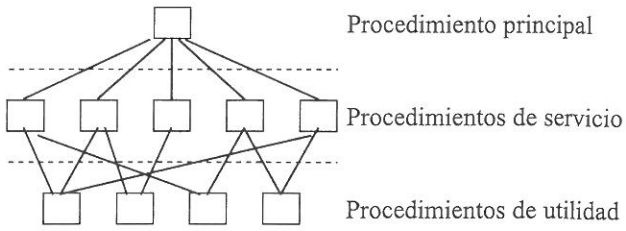
Sin embargo, incluso en los sistemas monolíticos es posible tener al menos algo de estructura. Los servicios (llamadas al sistema) que proporciona el sistema operativo se solicitan colocando los parámetros en lugares bien definidos, como en los registros o en la pila, para después ejecutar una instrucción especial de trampa de nombre llamada al núcleo o llamada al supervisor.

Esta instrucción cambia la máquina del modo usuario al modo núcleo (también conocido como modo supervisor) y transfiere el control al sistema operativo. El sistema operativo examina entonces los parámetros de la llamada, para determinar cuál de ellas se desea realizar. A continuación, el sistema operativo analiza una tabla que contiene en la entrada *k* un puntero al procedimiento que realiza la k-ésima llamada al sistema. Esta operación identifica el procedimiento de servicio, al cual se llama. Por último, la llamada al sistema termina y el control regresa al programa del usuario.

Esta organización sugiere una estructura básica del sistema operativo:

1. Un programa principal que llama al procedimiento del servicio solicitado.
2. Un conjunto de procedimientos de servicio que llevan a cabo las llamadas al sistema.
3. Un conjunto de procedimientos de utilidad que ayudan al procedimiento de servicio.

En este modelo, para cada llamada al sistema existe un procedimiento de servicio que se encarga de él. Los procedimientos de utilidad hacen cosas necesarias para varios procedimientos de servicio, como por ejemplo buscar los datos de los programas del usuario. Esta división de los procedimientos en tres capas se muestra en la figura.



5.2 Sistemas con capas.

Una generalización del punto de vista de la figura anterior consiste en organizar el sistema operativo como una jerarquía de capas, cada una construida sobre la inmediata inferior. El primer sistema construido de esta manera fue el sistema THE (*Technische Hogeschool Eindhoven*), que era un sistema sencillo de procesamiento por lotes. El sistema tenía 6 capas. La capa 0 trabaja con la asignación del procesador y alterna entre los procesos cuando ocurren las interrupciones o expiran los cronómetros. Sobre la capa 0, el sistema consta de procesos secuenciales, cada uno de los cuales se podía programar sin tener que preocuparse por el hecho de que varios procesos

estuvieran en ejecución en el mismo procesador. En otras palabras, la capa 0 proporcionaba la multiprogramación básica de la CPU.

La capa 1 realizaba la administración de la memoria. Asignaba el espacio de memoria principal para los procesos y un recipiente de palabras de 512K se utilizaba para almacenar partes de los procesos (páginas) para las que no existía lugar en la memoria principal. Por encima de la capa 1, los procesos no debían preocuparse si estaban en la memoria o en el recipiente; el software de la capa 1 se encargaba de garantizar que las páginas llegaran a la memoria cuando fueran necesarias.

La capa 2 se encargaba de la comunicación entre cada proceso y la consola del operador. Por encima de esta capa, cada proceso tiene su propia consola de operador. La capa 3 controla los dispositivos de E/S y guarda en almacenes (buffers) los flujos de información entre ellos. Por encima de la capa 3, cada proceso puede trabajar con dispositivos abstractos de E/S con propiedades adecuadas, en vez de dispositivos reales con muchas peculiaridades. La capa 4 es donde estaban los programas del usuario. Estos no tenían que preocuparse por el proceso, memoria, consola o control de E/S. El proceso operador del sistema se localizaba en la capa 5.

Una generalización más avanzada del concepto de capas se presentó en el sistema MULTICS. En lugar de capas, MULTICS estaba organizado como una serie de anillos concéntricos, siendo los anillos interiores los privilegiados. Cuando un procedimiento de un anillo exterior deseaba llamar a un procedimiento de un anillo interior, debía hacer el equivalente a una llamada al sistema; es decir, una instrucción TRAP cuyos parámetros eran validados con cuidado antes de permitir que procediese la llamada. Aunque todo el sistema operativo era parte del espacio de direcciones de cada proceso del usuario en MULTICS, el hardware posibilitó el diseño de procedimientos individuales (segmentos de memoria, en realidad) de forma protegida contra la lectura, escritura o ejecución.

Mientras que el esquema de capas de THE era en realidad un apoyo al diseño, debido a que todas las partes del sistema estaban, en última instancia, ligadas entre sí en un solo programa objeto, en MULTICS, el mecanismo de anillos estaba más presente durante el tiempo de ejecución y era reforzado por el hardware. La ventaja del mecanismo de anillos es su facilidad de extensión para estructurar subsistemas del usuario.

5.3 Máquinas virtuales.

Las versiones iniciales de OS/360 eran, en sentido estricto, sistemas de procesamiento por lotes. Sin embargo, muchos de los usuarios de 360 deseaban tener tiempo compartido, de forma que varios grupos, tanto dentro como fuera de IBM, decidieron escribir sistemas de tiempo compartido para tal sistema. El sistema de tiempo compartido oficial de IBM, TSS/360, tuvo una entrega retrasada y cuando finalmente apareció, era tan grande y lento que pocos lugares lo adoptaron. Un grupo del Centro Científico de IBM en Cambridge, Mass., produjo un sistema con diferencias radicales, el cual fue aceptado por IBM como producto y que ahora tiene un uso extenso.

Este sistema, cuyo nombre original era CP/CMS y que ahora se llama VM/370 se basó en una observación astuta: un sistema de tiempo compartido proporciona multiprogramación y una máquina extendida con una interfaz más conveniente que el mero hardware. La esencia de VM/370 es la total separación entre estas dos funciones.

El corazón del sistema, llamado monitor de la máquina virtual, se ejecuta en el hardware simple y realiza la multiprogramación, proporcionando no una, sino varias máquinas virtuales a la siguiente capa superior. Sin embargo, a diferencia de los demás sistemas operativos, estas máquinas virtuales no son máquinas extendidas, con archivos u otras características adecuadas. En lugar de esto, son copias exactas del hardware simple, con su modo núcleo/usuario, E/S, interrupciones y todo lo demás que posee la máquina real.

Puesto que cada máquina virtual es idéntica al hardware real, cada una puede ejecutar cualquier sistema operativo que se ejecute de forma directa sobre el hardware.

Las distintas máquinas virtuales pueden, y por lo general lo hacen, ejecutar distintos sistemas operativos. Algunas ejecutan uno de los descendientes de OS/360 para el procesamiento por lotes, mientras que otras ejecutan un sistema interactivo para un solo usuario, llamado CMS (*Conversational Monitor System*), para los usuarios del tiempo compartido.

Cuando un programa CMS ejecuta una llamada al sistema, la llamada es atrapada por el sistema operativo en su propia máquina virtual, no a VM/370, de la misma forma en que lo haría si se ejecutara en una máquina real en vez de una virtual. CMS proporciona entonces las instrucciones normales de E/S en hardware para la lectura del disco virtual o lo necesario para llevar adelante la llamada. Estas instrucciones de E/S son atrapadas por VM/370, que entonces las lleva a cabo como parte de la simulación del hardware verdadero. Al hacer una separación total de las funciones de multiprogramación y proporcionar una máquina extendida, cada una de las partes puede ser más sencilla, flexible y tener un fácil mantenimiento.

5.4 Modelo cliente/servidor o microkernel.

VM/370 ganó mucho en sencillez al trasladar gran parte del código tradicional del sistema operativo (al implantar la máquina extendida) en una capa superior, CMS. Sin embargo, VM/370 sigue siendo un programa complejo, puesto que la simulación de varios 370 virtuales no es tan simple (en especial si se desea que todo sea eficaz).

Una tendencia de los sistemas operativos modernos es la de explotar más esta idea de mover el código a capas superiores y eliminar la mayor parte posible del sistema operativo para mantener un núcleo mínimo (microkernel). El punto de vista usual es el de implementar la mayoría de las funciones del sistema operativo en los procesos del usuario. Para solicitar un servicio, como la lectura de un bloque de cierto archivo, un proceso del usuario (denominado en este caso proceso cliente) envía la solicitud a un proceso servidor, que realiza entonces el trabajo y devuelve la respuesta.

Se basa en que sólo las funciones absolutamente esenciales del núcleo del sistema operativo deben permanecer en el núcleo. Concretamente, un micronúcleo debe incluir aquellas funciones que dependen directamente del hardware y cuya funcionalidad es necesaria para dar soporte a las aplicaciones y servidores que ejecutan en modo de núcleo. Estas funciones se engloban en las siguientes categorías generales: gestión de memoria a bajo nivel, comunicación entre procesos (IPC) y gestión de interrupciones y E/S. Las aplicaciones y los servicios menos esenciales se construyen sobre el micronúcleo y se ejecutan en modo usuario. Por tanto, la función principal del núcleo es controlar la comunicación entre los clientes y los servidores. Al separar el sistema operativo en partes, cada una de ellas controla una faceta del sistema, como el servicio a archivos, servicio a procesos, servicio a terminales o servicio a la memoria, cada parte es pequeña y controlable. Además, puesto que todos los servidores se

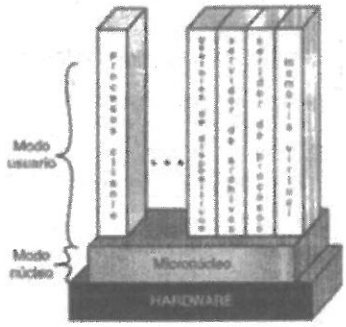
ejecutan como procesos en modo usuario y no en modo núcleo, no tienen acceso directo al hardware. En consecuencia, si hay un error en el servidor de archivos, éste puede fallar, pero esto no afectará en general a toda la máquina.

Otra de las ventajas del modelo cliente/servidor es su capacidad de adaptación para su uso en los sistemas distribuidos. Si un cliente se comunica, con un servidor mediante mensajes, el cliente no necesita saber si el mensaje se maneja de forma local, en su máquina, o si se envía por medio de una red a un servidor en una máquina remota. En lo que respecta al cliente, lo mismo ocurre en ambos casos: se envió una solicitud y se recibió una respuesta.

La idea anterior de un núcleo que sólo controla el transporte de mensajes de clientes a servidores y viceversa no es totalmente real. Algunas funciones del sistema operativo (como el cargado de comandos en los registros físicos del dispositivo de E/S) son difíciles, si no es que imposibles de realizar a partir de programas del usuario. Existen dos formas de enfrentar este problema. Una es la de hacer que algunos procesos de servidores críticos (por ejemplo, las directivas de los dispositivos de E/S) se ejecuten en realidad en modo núcleo, con acceso total al hardware, pero de forma que se comuniquen con los demás procesos mediante el mecanismo normal de mensajes.

La otra forma es la de construir una cantidad mínima de mecanismos dentro del núcleo, pero manteniendo las decisiones de política relativas a los usuarios dentro del espacio del usuario. Por ejemplo, el núcleo podría reconocer que cierto mensaje enviado a una dirección especial indica que se tome el contenido de ese mensaje y se cargue en los registros del dispositivo de E/S de algún disco, para iniciar la lectura del disco. En este ejemplo, el núcleo ni siquiera inspeccionaría los bytes del mensaje para ver si son válidos o tienen algún sentido; sólo los copiaría a ciegas en los registros del dispositivo del disco. (Es evidente que debe utilizarse cierto esquema para limitar tales mensajes sólo a los procesos autorizados.) La separación entre mecanismo y política es un concepto importante; aparece una y otra vez en diversos contextos de los sistemas operativos.

Los componentes del sistema operativo externos al micronúcleo se implementan como procesos servidores; estos interactúan uno con otro mediante mensajes distribuidos a través del micronúcleo. De este modo, el micronúcleo funciona como un distribuidor de mensajes: valida los mensajes, los pasa entre las componentes y otorga el acceso al hardware. El micronúcleo también actúa como función de protección, evitando el intercambio de mensajes cuando no está permitido.



Por ejemplo, si una aplicación quiere abrir un archivo, envía un mensaje al servidor del sistema de archivos. Si desea crear un proceso o hilo, envía un mensaje al servidor de procesos. Cada uno de los servidores puede enviar mensajes al resto y puede

invocar a las funciones primitivas del micronúcleo. Esta es una arquitectura cliente/servidor en un solo computador.

Este concepto ha recibido mucha atención últimamente. Este enfoque fue popularizado por el sistema operativo Mach. Otra aplicación muy conocida del enfoque de micronúcleos es Windows 2000. Actualmente, otros productos presumen de tener implementaciones de micronúcleo y este enfoque general de diseño parece que va asentarse en casi todos los computadores personales, estaciones de trabajo y sistemas operativos servidores que se desarrollen en el futuro.

6 Tipos de sistemas operativos.

La clasificación más usada y conocida desde el punto de vista del usuario final es la basada en el servicio del sistema operativo:

- Monousuario. Los sistemas operativos monousuario son aquéllos que soportan a un usuario a la vez, sin importar el número de procesadores que tenga la computadora o el número de procesos o tareas que el usuario pueda ejecutar en un mismo instante de tiempo. Las computadoras personales típicamente se han clasificado en este renglón.
- Multiusuario. Los sistemas operativos multiusuario son capaces de dar servicio a más de un usuario a la vez, ya sea por medio de varios terminales conectados a la computadora o por medio de sesiones remotas en una red de comunicaciones. No importa el número de procesadores en la máquina ni el número de procesos que cada usuario puede ejecutar simultáneamente.
- Monotarea. Los sistemas monotarea son aquellos que sólo permiten una tarea a la vez por usuario. Puede darse el caso de un sistema multiusuario y monotarea, en el cual se admiten varios usuarios al mismo tiempo pero cada uno de ellos puede estar haciendo sólo una tarea a la vez.
- Multitarea. Un sistema operativo multitarea es aquél que le permite al usuario estar realizando varias labores al mismo tiempo. Por ejemplo, puede estar editando el código fuente de un programa durante su depuración mientras compila otro programa, a la vez que está recibiendo correo electrónico en un proceso en background. Es común encontrar en ellos interfaces gráficas orientadas al uso de menús y el ratón, lo cual permite un rápido intercambio entre las tareas para el usuario, mejorando su productividad.
- Uniproceso. Un sistema operativo uniproceso es aquél que es capaz de manejar solamente un procesador de la computadora, de manera que si la computadora tuviese más de uno le sería inútil.
- Multiproceso. Un sistema operativo multiproceso se refiere al número de procesadores del sistema, que es más de uno y éste es capaz de usarlos todos para distribuir su carga de trabajo. Generalmente estos sistemas trabajan de dos formas: simétrica o asimétricamente. Cuando se trabaja de manera asimétrica, el sistema operativo selecciona a uno de los procesadores el cual jugará el papel de procesador maestro y servirá como pivote para distribuir la carga a los demás procesadores, que reciben el nombre de esclavos. Cuando se trabaja de manera simétrica, los procesos o partes de ellos (*threads*) son enviados indistintamente a cualesquiera de los procesadores disponibles,

teniendo, teóricamente, una mejor distribución y equilibrio en la carga de trabajo bajo este esquema.

Un thread es la parte activa en memoria y en ejecución de un proceso, lo cual puede consistir de un área de memoria, un conjunto de registros con valores específicos, la pila y otros valores de contexto.

Otra posible clasificación de los sistemas operativos es por la forma de ofrecer sus servicios:

- Sistemas operativos de red. Los sistemas operativos de red se definen como aquellos que tiene la capacidad de interactuar con sistemas operativos en otras computadoras por medio de un medio de transmisión con el objeto de intercambiar información, transferir archivos, ejecutar comandos remotos y un sin fin de otras actividades. El punto crucial de estos sistemas es que el usuario debe saber la sintaxis de un conjunto de comandos o llamadas al sistema para ejecutar estas operaciones, además de la ubicación de los recursos a los que desee acceder.
- Sistemas operativos distribuidos. Los sistemas operativos distribuidos abarcan los servicios de los de red, logrando integrar recursos (impresoras, unidades de respaldo, memoria, procesos, unidades centrales de proceso, etc.) en una sola máquina virtual a la que el usuario accede de forma transparente. Es decir, ahora el usuario ya no necesita saber la ubicación de los recursos, sino que los conoce por nombre y simplemente los usa como si todos ellos fuesen locales a su lugar de trabajo habitual.

7 Conclusiones.

En este tema partimos del análisis de los sistemas operativos desde dos puntos de vista: controladores de recursos y máquinas extendidas. Desde el punto de vista del controlador de recursos, la labor del sistema operativo es la de controlar con eficacia las distintas partes del sistema. Desde el punto de vista de máquina extendida, su labor es la de proporcionar a los usuarios una máquina virtual que tenga un uso más conveniente que la máquina real.

Después presentamos dos de los conceptos fundamentales de los sistemas operativos, el proceso y el archivo. También describimos en forma breve las llamadas al sistema.

En el siguiente punto, presentamos los cuatro componentes principales de los sistemas operativos: control de procesos, administración de la memoria, control de archivos y control de los dispositivos de E/S.

A continuación, examinamos distintas formas de estructurar un sistema operativo: como sistema monolítico, como una jerarquía de capas, como un sistema con máquina virtual y como un modelo cliente-servidor.

Por último, presentamos los diferentes tipos de sistemas operativos, clasificándolos en función del servicio ofrecido al usuario, y en función de la forma de ofrecer dichos servicios.