Introducción a los Sistemas Operativos

1. Sistema Operativo

Un Sistema Operativo (SO) es un software, formado por un conjunto integrado de programas que controla los recursos del ordenador (CPU, memoria, dispositivos de E/S, etc.) y proporciona servicios comunes para la ejecución eficiente de diversos programas de aplicación con una interfaz o máquina virtual que resulta más cómoda de utilizar. En resumen, podemos decir que *el sistema operativo es una interfaz entre el hardware y el usuario*. Para funciones de hardware como la entrada y salida y la asignación de memoria, el sistema operativo actúa como intermediario entre los programas de aplicación y el hardware del ordenador, aunque el código de la aplicación suele ser ejecutado directamente por el hardware, pero con frecuencia llamará al SO o será interrumpido por él. Los sistemas operativos se encuentran en casi cualquier dispositivo que contenga un ordenador, desde teléfonos móviles y consolas de videojuegos hasta superordenadores y servidores web. Ejemplos de sistemas operativos modernos populares para ordenadores personales son Microsoft Windows, Mac OS y Linux. Los dos objetivos principales de un sistema operativo son los siguientes:

- **Facilitar el uso de un sistema informático** Un sistema informático consta de uno o varios procesadores, memoria principal y muchos tipos de dispositivos de E/S, como discos, cintas, terminales, interfaces de red, etc. Escribir programas para utilizar estos recursos de hardware de forma correcta y eficiente es un trabajo extremadamente difícil, que requiere un conocimiento profundo del funcionamiento de los recursos. Por ello, para que los sistemas informáticos puedan ser utilizados por un gran número de usuarios, hace ya varios años que se hizo evidente la necesidad de proteger a los programadores de la complejidad de los recursos de hardware. La solución, que ha ido evolucionando gradualmente, consiste en colocar una capa de software sobre el hardware para gestionar todas las partes del sistema y presentar al usuario una interfaz o máquina virtual más fácil de programar y utilizar. Esta capa de software se denomina sistema operativo.

  La arquitectura lógica de un sistema informático se muestra en la Fig. 01. Como se muestra en la figura, los recursos de hardware están rodeados por la capa del sistema operativo, que a su vez está rodeada por una capa de otro software del sistema (como compiladores, editores, intérprete de comandos, utilidades, etc.) y un conjunto de programas de aplicación (como aplicaciones comerciales de procesamiento de datos, aplicaciones científicas y de ingeniería, aplicaciones de entretenimiento y educativas, etc.). Por último, los usuarios finales ven el sistema informático en términos de las interfaces de usuario proporcionadas por los programas de aplicación.

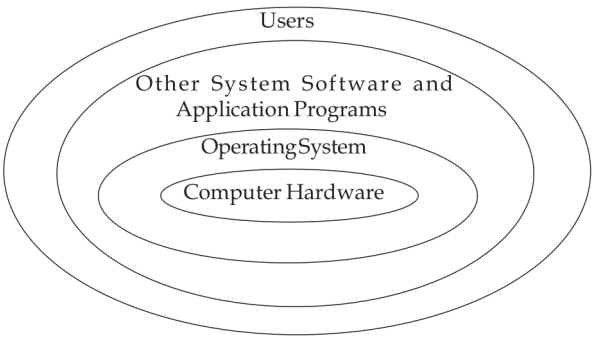

Fig. 01: Arquitectura de un sistema informático

Las capas del sistema operativo proporcionan varias facilidades y servicios que hacen que el uso de los recursos de hardware sea cómodo, eficiente y seguro. Un programador hace uso de estas facilidades al desarrollar una aplicación, y ésta, mientras se ejecuta, invoca los servicios necesarios para realizar determinadas funciones. En efecto, el sistema operativo oculta al programador los detalles del hardware y proporciona una interfaz cómoda para utilizar el sistema. Actúa como intermediario entre el hardware y sus usuarios, proporcionando una interfaz de alto nivel a los recursos de hardware de bajo nivel y facilitando al programador y a los programas de aplicación el acceso y uso de dichos recursos.

- **Gestionar los recursos de un sistema informático** El segundo objetivo importante de un sistema operativo es gestionar los distintos recursos del sistema informático. Esto implica realizar tareas como llevar un

registro de “quién está utilizando qué recurso”, conceder solicitudes de recursos, contabilizar el uso de los recursos y mediar en las solicitudes conflictivas de diferentes programas y usuarios.

Ejecutar un trabajo en un sistema informático suele requerir varios de sus recursos, como tiempo de CPU, espacio de memoria, espacio de almacenamiento de archivos, dispositivos de E/S, etcétera. El sistema operativo actúa como gestor de los distintos recursos de un sistema informático y los asigna a programas y usuarios específicos para que ejecuten sus trabajos con éxito. Cuando un sistema informático se utiliza para gestionar simultáneamente varias aplicaciones, puede haber muchas solicitudes de recursos, posiblemente conflictivas. En tal situación, el sistema operativo debe decidir .ª qué solicitudes se asignan recursos para que el sistema informático funcione de forma eficiente y justa (prestando la debida atención a todos los usuarios)". El reparto eficiente y justo de los recursos entre usuarios y/o programas es un objetivo clave de la mayoría de los sistemas operativos.

2. Principales Funciones de un Sistema Operativo

En la sección anterior se ha comentado claramente que el sistema operativo proporciona ciertos servicios tanto a los programas como a los usuarios de dichos programas. Los servicios específicos proporcionados diferirán, por supuesto, de un sistema operativo a otro, pero hay algunos tipos comunes de funciones que podemos identificar. Las principales funciones proporcionadas por la mayoría de los sistemas operativos son las siguientes:

- **Gestión de procesos** Un proceso es un programa en ejecución. El sistema operativo gestiona muchos tipos de actividades que van desde programas de usuario a programas de sistema como spooler de impresora, servidores de nombres, servidor de archivos, etc. Cada una de estas actividades se encapsula en un proceso. Un proceso incluye el contexto de ejecución completo (código, datos, PC, registros, recursos del sistema operativo en uso, etc.).

  Es importante señalar que un proceso no es un programa. Un proceso es sólo un instante de un programa en ejecución. Hay muchos procesos, pueden estar ejecutando el mismo programa. Las cinco actividades principales de un sistema operativo con respecto a la gestión de procesos son:

  - Creación y eliminación de procesos de usuario y de sistema;
  - Suspensión y reanudación de procesos;
  - Un mecanismo para la sincronización de procesos;
  - Un mecanismo para la comunicación de procesos; y
  - Mecanismo de gestión de bloqueos.

- **Gestión de la memoria**: Para ejecutar un programa, éste debe cargarse, junto con los datos a los que accede, en la memoria principal (al menos parcialmente). Para mejorar la utilización de la CPU y ofrecer un mejor tiempo de respuesta a sus usuarios, un sistema informático suele guardar varios programas en la memoria principal. El módulo de gestión de memoria de un sistema operativo se encarga de asignar y desasignar el espacio de memoria a los distintos programas que necesitan este recurso. La memoria principal es un gran conjunto de palabras o bytes. Cada palabra o byte tiene su propia dirección. La memoria principal proporciona almacenamiento al que puede acceder directamente la CPU. Es decir, para que un programa se ejecute, debe estar en la memoria principal. Las principales actividades de un sistema operativo con referencia a la gestión de la memoria son:

  - Hacer un seguimiento de ’qué parte de la memoria se está utilizando actualmente y por quién’;
  - Decidir ’qué proceso se carga en la memoria cuando el espacio de memoria está disponible; y asignar y desasignar espacio de memoria, según sea necesario.

- **Gestión de archivos**. Un archivo es una colección de información relacionada definida por su creador. Los ordenadores pueden almacenar archivos en el disco (almacenamiento secundario), que proporciona almacenamiento a largo plazo. Algunos ejemplos de medios de almacenamiento son la cinta magnética, el disco magnético y el disco óptico. Cada uno de estos medios tiene sus propias propiedades, como la velocidad, la capacidad, la velocidad de transferencia de datos y los métodos de acceso. Un sistema de archivos normalmente se organiza en directorios para facilitar su uso. Estos directorios pueden contener archivos y otras direcciones. Las cinco actividades principales de un sistema operativo en relación con la gestión de archivos son las siguientes:

- Creación y eliminación de archivos;
- La creación y eliminación de direcciones;
- El soporte de primitivas para manipular archivos y direcciones;
- La asignación de archivos al almacenamiento secundario; y
- La copia de seguridad de archivos en medios de almacenamiento estables.

- **Gestión de dispositivos**: Un sistema informático consta normalmente de varios dispositivos de E/S, como terminales, impresoras, discos y cintas. El módulo de gestión de dispositivos de un sistema operativo se encarga de controlar todos los dispositivos de E/S del ordenador. Realiza un seguimiento de las solicitudes de E/S de los procesos, emite comandos a los dispositivos de E/S y garantiza la correcta transmisión de datos a/desde un dispositivo de E/S. También proporciona una interfaz entre los dispositivos. También proporciona una interfaz entre los dispositivos y el resto del sistema que es simple y fácil de usar. A menudo, esta interfaz es independiente del dispositivo, es decir, la interfaz es la misma para todos los tipos de dispositivos de E/S.
- **Seguridad**: Los sistemas informáticos suelen almacenar grandes cantidades de información, parte de la cual es muy sensible y valiosa para sus usuarios. Los usuarios sólo pueden confiar en el sistema si los distintos recursos e información de un sistema informático están protegidos contra la destrucción y el acceso no autorizado. El módulo de seguridad de un sistema operativo se encarga de ello. Este módulo también garantiza que, cuando se ejecutan simultáneamente varios procesos disjuntos, uno de ellos no interfiera con los demás ni con el propio sistema operativo.
- **Interpretación de comandos**: Un intérprete de comandos es una interfaz del sistema operativo con el usuario. El usuario da órdenes que son ejecutadas por el sistema operativo (normalmente convirtiéndolas en llamadas al sistema). La función principal de un intérprete de comandos es obtener y ejecutar el siguiente comando especificado por el usuario. El intérprete de comandos no suele formar parte del núcleo, ya que un sistema operativo puede admitir varios intérpretes de comandos (shell, en terminología UNIX) y no es necesario que se ejecuten en modo núcleo. Separar el intérprete de comandos del núcleo tiene dos ventajas principales:
  - Si queremos cambiar el aspecto del intérprete de comandos, es decir, si queremos cambiar la interfaz del intérprete de comandos, podemos hacerlo si el intérprete de comandos está separado del núcleo. No podemos cambiar el código del núcleo, por lo que no podemos modificar la interfaz.
  - Si el intérprete de comandos es una parte del kernel; es posible que un proceso malicioso obtenga acceso a cierta parte del kernel que mostró, para evitar este feo escenario. Es ventajoso tener el intérprete de comandos separado del kernel.

Además de las funciones principales mencionadas, un sistema operativo también realiza algunas otras funciones como “llevar la cuenta de qué usuario (o procesos) utilizan cuánto” y “qué tipo de recursos informáticos, mantenimiento del registro de uso del sistema por parte de todos los usuarios” y “mantenimiento del reloj interno”.

3. Medición del Rendimiento del Sistema

La eficiencia de un sistema operativo y el rendimiento global de un sistema informático suelen medirse en función de los siguientes aspectos:

- **Rendimiento / Throughput:**: El *rendimiento* es la cantidad de trabajo que el sistema es capaz de realizar por unidad de tiempo. Se mide como el número de procesos que completa el sistema por unidad de tiempo. Por ejemplo, si se completan n procesos en un intervalo de t segundos, el rendimiento se calcula como n/t procesos por segundo durante ese intervalo. El rendimiento se mide normalmente en procesos/hora. Cabe destacar que el valor del rendimiento no depende sólo de la capacidad de un sistema, sino también de la naturaleza de los trabajos que procesa. Para procesos largos, el rendimiento puede ser de un proceso/hora; y para procesos cortos, el rendimiento puede ser de 100 procesos/hora.
- **Tiempo de respuesta / Turnaround time**: Desde el punto de vista de un usuario individual, un criterio importante es “cuánto tarda el sistema en completar un trabajo enviado por él/ella”. El tiempo de respuesta es el intervalo entre el momento en que un trabajo se envía al sistema para su procesamiento y el momento

en que se completa. Aunque un mayor rendimiento es deseable desde el punto de vista del rendimiento general del sistema, los usuarios individuales están más interesados en un mejor tiempo de respuesta para sus trabajos.

- **Tiempo de respuesta / Response time**: El *Turnaround time* no suele ser una medida adecuada para los sistemas interactivos, ya que en un sistema interactivo, un proceso puede producir algún resultado bastante pronto durante su ejecución y puede seguir ejecutándose mientras se envían los resultados anteriores al usuario. Por lo tanto, otra medida utilizada en el caso de los sistemas interactivos es el *tiempo de respuesta*, que es el intervalo desde el momento de la presentación de un trabajo al sistema para su procesamiento hasta el momento en que el sistema produce la primera respuesta para el trabajo.

  En cualquier sistema informático, es deseable maximizar el rendimiento y minimizar el tiempo de respuesta.

4. Gestión de Procesos

Un proceso es un programa secuencial en ejecución. Los componentes de un proceso son los siguientes

- El programa objeto a ejecutar (llamado *texto del programa* en UNIX);
- Los *datos* sobre los que se ejecutará el programa (obtenidos de un archivo o interactivamente del usuario del proceso);
- Los *recursos* que necesita el programa (por ejemplo, archivos que contengan la información necesaria); y El *estado / status* de ejecución del proceso.

Durante la vida de un proceso, su estado de ejecución puede estar en uno de cuatro estados (asociado a cada estado suele haber una cola en la que reside el proceso):

- **Ejecutándose / Executing**: el proceso se está ejecutando actualmente y tiene el control de una CPU;
- **En espera / Waiting**: el proceso puede ejecutarse, pero debe esperar a que haya una CPU disponible;
- **Bloqueado / Blocked**: el proceso está esperando en E/S, ya sea a que llegue la entrada o a que se envíe la salida;
- **Suspendido / Suspended**: el proceso puede ejecutarse, pero por alguna razón el sistema operativo no lo ha colocado en la cola de espera; y
- **Listo / Ready**: el proceso está en memoria, se ejecutará dado el tiempo de CPU.

4.1. Gestión de procesos en los primeros sistemas

En los primeros sistemas informáticos, un trabajo se ejecutaba normalmente de la siguiente manera:

- Un programador escribía primero el programa en papel.
- A continuación, se perforaba en tarjetas o cinta de papel junto con sus datos.
- La baraja de tarjetas o la cinta de papel que contenía el programa y los datos se entregaba en el mostrador de recepción del centro informático.
- Un operario cogía la baraja o la cinta de papel y la cargaba manualmente en el sistema desde el lector de tarjetas o el lector de cintas de papel. El operador también se encargaba de cargar cualquier otro recurso de software (como un compilador de lenguaje) o de configurar los dispositivos de hardware necesarios para la ejecución del trabajo. Antes de cargar el trabajo, el operador tenía que utilizar los interruptores del panel frontal del sistema informático para borrar la memoria principal y eliminar cualquier dato que quedara del trabajo anterior.
- A continuación, el operador ajustaba los interruptores adecuados en el panel frontal para ejecutar el trabajo.

- A continuación, se imprimía en la impresora el resultado de la ejecución del trabajo, que el operador llevaba al contador de recepción para que el programador pudiera recogerlo posteriormente.
- Había que repetir el mismo proceso para todos y cada uno de los trabajos que iba a ejecutar el ordenador. Este método de ejecución de trabajos se conocía como mecanismo de carga manual porque los trabajos tenían que ser cargados manualmente uno tras otro por el operador del ordenador en el sistema informático. Nótese que en este método, la transición de un trabajo a otro no era automática. La transición manual de un trabajo a otro hacía que se perdiera mucho tiempo de ordenador, ya que éste permanecía inactivo mientras el operador cargaba y descargaba los trabajos y preparaba el sistema para un nuevo trabajo. Para reducir este tiempo de inactividad del ordenador, se ideó un método de transición automática de un trabajo a otro. En este método, conocido como procesamiento por lotes, cuando finaliza un trabajo, el control del sistema se transfiere automáticamente al sistema operativo, que realiza automáticamente las tareas de mantenimiento (como borrar la memoria para eliminar los datos restantes del trabajo anterior) necesarias para cargar y ejecutar el siguiente trabajo. En el caso de los sistemas de procesamiento por lotes, los trabajos solían ejecutarse de la siguiente manera:
- Los programadores preparaban sus programas y datos en mazos de tarjetas o cintas de papel y los entregaban en el mostrador de recepción del centro informático.
- El operador recogía periódicamente todos los programas enviados, los agrupaba y los cargaba todos a la vez en el dispositivo de entrada del sistema.
- A continuación, el operario daba una orden al sistema para que empezara a ejecutar los trabajos.
- Los trabajos se cargaban automáticamente desde el dispositivo de entrada y el sistema los ejecutaba uno a uno sin intervención del operario. Es decir, el sistema leía el primer trabajo del dispositivo de entrada, lo ejecutaba, imprimía el resultado en la impresora y repetía estos pasos para cada uno de los trabajos siguientes hasta que se terminaban todos los trabajos del lote enviado.
- Cuando se procesaban todos los trabajos del lote enviado, el operador separaba la salida impresa de cada trabajo y los guardaba en el contador de recepción para que los programadores pudieran recogerlos más tarde.

5. Multiprogramación

En los sistemas de multiprogramación, la tarea en ejecución sigue ejecutándose hasta que realiza una operación que requiere esperar un evento externo (por ejemplo, la lectura de una cinta) o hasta que el planificador del ordenador cambia forzosamente la tarea en ejecución fuera de la CPU. Los sistemas multiprogramación están diseñados para maximizar el uso de la CPU. De hecho, dependiendo de la utilización de la CPU durante el curso del procesamiento, los trabajos se clasifican a grandes rasgos en los dos tipos siguientes:

- **Trabajos ligados a la CPU / CPU-bound jobs**: Estos trabajos realizan principalmente cálculos numéricos, con pocas operaciones de E/S. Se llaman así porque utilizan mucho la CPU durante su procesamiento. Se llaman así porque utilizan mucho la CPU durante su procesamiento. Los programas utilizados para cálculos científicos y de ingeniería suelen entrar en esta categoría de tareas.
- **Trabajos de E/S / I/O-bound jobs**: Estos trabajos normalmente introducen una gran cantidad de datos, realizan muy pocos cálculos y generan una gran cantidad de información. Esto se debe a que, durante su procesamiento, la utilización de la CPU es muy baja y la mayor parte del tiempo realizan operaciones de E/S. Los programas utilizados para aplicaciones comerciales de procesamiento de datos suelen entrar en esta categoría de trabajos.

5.1. Requisitos de los sistemas multiprogramación

Los sistemas multiprogramación tienen mejor rendimiento que los sistemas uniprogramación porque el tiempo de inactividad de la CPU se reduce drásticamente. Sin embargo, los sistemas multiprogramación son bastante sofisticados porque requieren las siguientes características adicionales de hardware y software:

- **Memoria grande / Large memory**: Para que la multiprogramación funcione satisfactoriamente, se necesita una memoria principal grande para dar cabida a un buen número de programas de usuario junto con el sistema operativo.
- **Memory protection / Memory protection**: Los ordenadores diseñados para multiprogramación deben proporcionar algún tipo de mecanismo de protección de memoria para evitar que un trabajo en una partición de memoria cambie información o instrucciones de un trabajo en otra partición de memoria. Por ejemplo, no queremos que el trabajo A destruya inadvertidamente algo en el trabajo B o el trabajo C, que son completamente independientes. En un sistema multiprogramación, esto se consigue mediante la función de protección de memoria, una combinación de hardware y software que impide que un trabajo se dirija más allá de los límites de su propia área de memoria asignada.
- **Conservación del estado del trabajo / Job status preservation**: En la multiprogramación, cuando un trabajo en ejecución se bloquea para el procesamiento de E/S, la CPU se retira de este trabajo y se entrega a otro que esté listo para su ejecución. Más adelante, se asignará la CPU al trabajo anterior para que continúe su ejecución. Hay que tener en cuenta que esto requiere preservar la información completa del estado del trabajo cuando se le quita la CPU y restaurar esta información antes de que se le devuelva la CPU de nuevo. Para permitir esto, el sistema operativo mantiene un Bloque de Control de Proceso (*PCB, Process Control Block*) para cada proceso cargado. En la Fig. 02 se muestra un bloque de control de procesos típico. Con esta disposición, antes de quitar la CPU a un proceso en ejecución, su estado se conserva en su PCB, y antes de que el proceso reanude la ejecución cuando se le devuelva la CPU más tarde, su estado se restaura desde su PCB. De este modo, el proceso puede continuar su ejecución sin ningún problema.

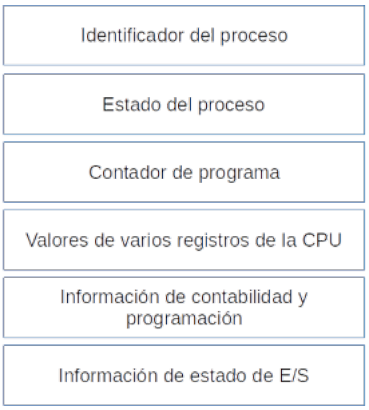

Fig. 02: Un típico bloque de control de procesos (PCB)

- **Combinación adecuada de trabajos / Proper job mix**: Se requiere una mezcla adecuada de trabajos ligados a E/S y trabajos ligados a CPU para solapar eficazmente las operaciones de la CPU y los dispositivos de E/S. Si todos los trabajos cargados necesitan E/S al mismo tiempo, la CPU volverá a estar ociosa. Por lo tanto, la memoria principal debe contener algunos trabajos ligados a la CPU y otros ligados a la E/S para que al menos un trabajo esté siempre listo para utilizar la CPU.
- **Programación de la CPU / CPU scheduling**: En un sistema multiprogramación, a menudo habrá situaciones en las que dos o más trabajos estarán en estado listo esperando que el CPU sea asignado para su ejecución. Cuando más de un proceso está en estado listo y la CPU queda libre, el sistema operativo debe decidir a cuál de los trabajos listos se le debe asignar la CPU para su ejecución. La parte del sistema operativo que se ocupa de esta decisión se denomina *planificador / scheduler* de la CPU, y el algoritmo que utiliza se denomina *algoritmo de planificación de la CPU*.

6. Multitarea / Multitasking

La multitarea es un método en el que varios procesos de tareas comparten recursos de procesamiento comunes, como una CPU. En el caso de un ordenador con una sola CPU, se dice que sólo una tarea se está ejecutando (*running*) en un momento dado, lo que significa que la CPU está ejecutando activamente instrucciones para esa tarea. La multitarea resuelve el problema programando qué tarea puede ser la que se esté ejecutando en un momento

dado, y cuándo le toca el turno a otra tarea en espera. El acto de reasignar una CPU de una tarea a otra se denomina cambio de contexto. Cuando los cambios de contexto se producen con suficiente frecuencia, se consigue la ilusión de paralelismo. Incluso en ordenadores con más de una CPU (llamados máquinas multiprocesador), la multitarea permite ejecutar muchas más tareas que CPUs hay.

Muchas personas no distinguen entre multiprogramación y multitarea porque ambos términos se refieren al mismo concepto. Sin embargo, algunas personas prefieren utilizar el término multiprogramación para los sistemas multiusuario (sistemas que son utilizados simultáneamente por muchos usuarios, como los sistemas mainframe y de clase servidor), y multitarea para los sistemas monousuario (sistemas que son utilizados por un solo usuario a la vez, como un ordenador personal o un ordenador portátil). Tenga en cuenta que, incluso en un sistema monousuario, no es necesario que el sistema trabaje sólo en una tarea a la vez. De hecho, un usuario de un sistema monousuario a menudo tiene múltiples tareas procesadas concurrentemente por el sistema. Por ejemplo, mientras se edita un archivo en primer plano, se puede dar un trabajo de clasificación en segundo plano. Del mismo modo, mientras se compila un programa en segundo plano, el usuario puede estar leyendo sus correos electrónicos en primer plano. De este modo, un usuario puede trabajar simultáneamente en varias tareas. En tal situación, el estado de cada una de las tareas se visualiza normalmente en la pantalla del ordenador dividiendo la pantalla en varias ventanas. En un sistema multitarea, el progreso de las distintas tareas puede verse en ventanas diferentes.

Por lo tanto, para aquellos que quieran diferenciar entre multiprogramación y multitarea, *la multiprogramación es la ejecución concurrente de múltiples trabajos (del mismo o de diferentes usuarios) en un sistema multiusuario, mientras que la multitarea es la ejecución concurrente de múltiples trabajos (a menudo denominados tareas del mismo usuario) en un sistema monousuario*.

7. Multihilos / Multithreading

Los hilos son una forma popular de mejorar el rendimiento de las aplicaciones. En los sistemas operativos tradicionales, la unidad básica de utilización de la CPU es un proceso. Cada proceso tiene su propio contador de programa, sus propios estados de registro, su propia pila y su propio espacio de direcciones (área de memoria que se le asigna). Por otro lado, en los sistemas operativos, con la facilidad de hilos, la unidad básica de utilización de la CPU es un hilo. En estos sistemas operativos, un proceso consiste en un espacio de direcciones y uno o más hilos de control como se muestra en la Fig 03 (a). Cada hilo de un proceso tiene su propio contador de programa, sus propios estados de registro y su propia pila. Pero todos los hilos de un proceso comparten el mismo espacio de direcciones. Por lo tanto, también comparten las mismas variables globales. Además, todos los hilos de un proceso también comparten el mismo conjunto de recursos del sistema operativo, como archivos abiertos, señales, información contable, etc. Debido a la compartición del espacio de direcciones, no hay protección entre los hilos de un proceso. Sin embargo, esto no es un problema. La protección entre procesos es necesaria porque diferentes procesos pueden pertenecer a diferentes usuarios. Pero un proceso (y, por tanto, todos sus subprocesos) siempre pertenece a un único usuario. Por lo tanto, la protección entre varios subprocesos de un proceso no es necesaria. Si se requiere protección entre dos hebras de un proceso, es preferible ponerlas en procesos diferentes, en lugar de ponerlas en un único proceso.

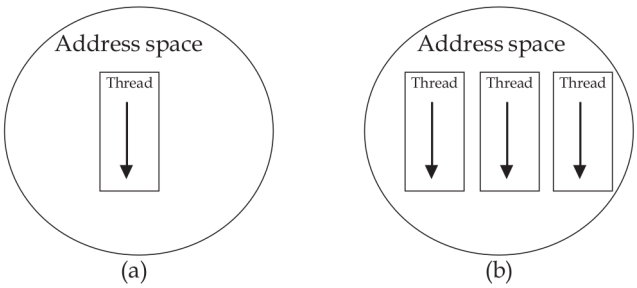

Figura 03: (a) Procesos monohilo y (b) multihilo

Un proceso monohilo corresponde a un proceso de un sistema operativo tradicional. Los hilos comparten la CPU del mismo modo que los procesos. En un momento dado, un hilo puede estar en cualquiera de los siguientes estados: en ejecución, bloqueado, listo o finalizado. Debido a estas similitudes, los hilos suelen considerarse miniprocesos. De hecho, en los sistemas operativos con la facilidad de hilos, un proceso que tiene un solo hilo corresponde a un proceso de un sistema operativo tradicional, como se muestra en la Fig. 03 (b). Los hilos suelen

denominarse *procesos ligeros / lightweight processes* y los procesos tradicionales, *procesos pesados / heavyweight processes*.

8. Multiprocesamiento

El multiprocesamiento es el uso de dos o más unidades centrales de procesamiento (CPU) dentro de un mismo sistema informático. El término también se refiere a la capacidad de un sistema para soportar más de un procesador y/o la capacidad de asignar tareas entre ellos. Hay muchas variaciones sobre este tema básico, y la definición de multiprocesamiento puede variar según el contexto, sobre todo en función de cómo se definan las CPU (varios núcleos en una sola matriz, varias matrices en un solo paquete, varios paquetes en una unidad del sistema, etc.). El multiprocesamiento a veces se refiere a la ejecución de múltiples procesos de software concurrentes en un sistema, en contraposición a un único proceso en un instante dado. Sin embargo, los términos multitarea o multiprogramación son más apropiados para describir este concepto, que se implementa principalmente en software, mientras que multiprocesamiento es más apropiado para describir el uso de múltiples CPUs de hardware. Un sistema puede ser tanto multiproceso como multiprogramación, sólo uno de los dos, o ninguno de los dos.

En un sistema multiproceso, todas las CPU pueden ser iguales, o algunas pueden reservarse para fines especiales. Una combinación de consideraciones de diseño de hardware y software del sistema operativo determinan la simetría (o la falta de ella) en un sistema dado. Por ejemplo, las consideraciones de hardware o software pueden requerir que sólo una CPU responda a todas las interrupciones de hardware, mientras que el resto del trabajo en el sistema puede distribuirse equitativamente entre las CPUs; o la ejecución de código en modo kernel puede estar restringida a sólo un procesador (ya sea un procesador específico, o sólo un procesador a la vez), mientras que el código en modo usuario puede ejecutarse en cualquier combinación de procesadores. Los sistemas multiproceso suelen ser más fáciles de diseñar si se imponen estas restricciones, pero tienden a ser menos eficientes que los sistemas en los que se utilizan todas las CPU. Los sistemas que tratan a todas las CPU por igual se denominan sistemas de multiprocesamiento simétrico (SMP). En los sistemas en los que no todas las CPU son iguales, los recursos del sistema se pueden dividir de varias formas, como el multiprocesamiento asimétrico (ASMP), el multiprocesamiento de acceso no uniforme a la memoria (NUMA) y el multiprocesamiento en clúster.

Los sistemas de multiprocesamiento son básicamente de dos tipos: sistemas muy acoplados y sistemas poco acoplados:

- **Sistemas multiprocesador estrechamente y débilmente acoplados /** *Tightly and Loosely Coupled Multiprocessing Systems* Los sistemas multiprocesador estrechamente acoplados contienen varias CPU conectadas a nivel de bus. Estas CPU pueden tener acceso a una memoria compartida central (SMP o UMA) o participar en una jerarquía de memoria con memoria local y compartida (NUMA). El IBM p690 Regatta es un ejemplo de sistema SMP de gama alta. Los procesadores Intel Xeon dominaban el mercado de los multiprocesadores para PC empresariales y fueron la única opción x86 hasta el lanzamiento de la gama de procesadores Opteron de AMD en 2004. Ambas gamas de procesadores tenían su propia caché integrada, pero proporcionaban acceso a la memoria compartida; los procesadores Xeon a través de una tubería común y los procesadores Opteron a través de rutas independientes a la RAM del sistema. Los multiprocesadores de chip, también conocidos como computación multinúcleo, implican más de un procesador colocado en un único chip y pueden considerarse la forma más extrema de multiprocesamiento estrechamente acoplado. Los sistemas mainframe con múltiples procesadores suelen estar estrechamente acoplados.
- **Sistemas multiprocesador poco acoplados /** *Loosely Coupled Multiprocessing Systems* Los sistemas multiprocesador poco acoplados (a menudo denominados clústeres) se basan en varios ordenadores autónomos de uno o dos procesadores interconectados mediante un sistema de comunicación de alta velocidad (Gigabit Ethernet es habitual). Un clúster Linux Beowulf es un ejemplo de sistema débilmente acoplado.

Los sistemas muy acoplados funcionan mejor y son físicamente más pequeños que los sistemas poco acoplados, pero históricamente han requerido mayores inversiones iniciales y pueden depreciarse rápidamente; los nodos de un sistema poco acoplado suelen ser ordenadores básicos baratos y pueden reciclarse como máquinas independientes cuando se retiran del clúster. El consumo de energía también es un factor a tener en cuenta. Los sistemas muy acoplados tienden a ser mucho más eficientes energéticamente que los clusters. Esto se debe a que se pueden realizar considerables ahorros diseñando los componentes para que trabajen juntos desde el principio en sistemas fuertemente acoplados, mientras que los sistemas poco acoplados utilizan componentes que no fueron necesariamente pensados específicamente para su uso en tales sistemas.

8.1. Diferencia entre multiprogramación y multiprocesamiento

La multiprogramación es la ejecución intercalada de dos o más procesos por un sistema informático con una sola CPU. Por otro lado, el multiprocesamiento es la ejecución simultánea de dos o más procesos por un sistema informático que tiene más de una CPU. Más concretamente, la multiprogramación consiste en ejecutar una parte de un programa, luego un segmento de otro, etc., en breves periodos de tiempo consecutivos. El multiprocesamiento, sin embargo, hace posible que el sistema trabaje simultáneamente en varios segmentos de uno o más programas.

8.2. Ventajas y limitaciones del multiprocesamiento

Los sistemas multiproceso presentan normalmente las siguientes ventajas:

- **Mejor rendimiento** / *Better Performance*: Debido a la multiplicidad de procesadores, los sistemas multiprocesador tienen mejor rendimiento que los sistemas de un solo procesador. Es decir, los múltiples procesadores de un sistema de este tipo pueden utilizarse adecuadamente para proporcionar tiempos de respuesta más cortos y un mayor rendimiento que un sistema de un solo procesador. Por ejemplo, si hay que ejecutar dos programas diferentes, dos procesadores son evidentemente más potentes que uno porque los programas pueden ejecutarse simultáneamente en procesadores diferentes.

  Mayor fiabilidad / *Better Reliability*: Debido a la multiplicidad de procesadores, los sistemas multiprocesador también tienen mejor fiabilidad que los sistemas de un solo procesador. En un sistema multiprocesador correctamente diseñado, si uno de los procesadores se avería, el otro u otros procesadores asumen automáticamente la carga de trabajo del sistema hasta que se efectúen las reparaciones. Así se puede evitar una avería completa de estos sistemas. Por ejemplo, si un sistema tiene 4 procesadores y falla uno, los 3 restantes pueden utilizarse para procesar los trabajos enviados al sistema. Así, todo el sistema funciona sólo un 25 % más lento, en lugar de fallar por completo. Esta capacidad de un sistema para seguir prestando un servicio proporcional al nivel de hardware sin fallos se denomina función de *degradación gradual* (*graceful degradation*).

Los sistemas multiproceso, sin embargo, requieren un sistema operativo muy sofisticado para programar, equilibrar y coordinar las actividades de entrada, salida y procesamiento de múltiples procesadores. El diseño de un sistema operativo de este tipo es una tarea compleja y laboriosa. Además, los sistemas multiprocesador son caros de adquirir y mantener. Además del elevado coste inicial, el funcionamiento y mantenimiento regular de estos sistemas también es un asunto costoso.

9. Tiempo Compartido / Time-Sharing

El tiempo compartido es el reparto de un recurso informático entre muchos usuarios mediante la multiprogramación y la multitarea. Este concepto, introducido en los años 60 y convertido en el modelo informático más destacado en los 70, representa un importante cambio tecnológico en la historia de la informática. Al permitir a un gran número de usuarios interactuar simultáneamente con un único ordenador, el tiempo compartido redujo drásticamente el coste de proporcionar capacidad informática, hizo posible que personas y organizaciones utilizaran un ordenador sin poseerlo y promovió el uso interactivo de los ordenadores y el desarrollo de nuevas aplicaciones interactivas. El tiempo compartido es un mecanismo que permite el uso interactivo simultáneo de un sistema informático por parte de varios usuarios, de forma que cada uno de ellos tiene la impresión de disponer de su propio ordenador. Para ello se utiliza la multiprogramación con un algoritmo especial de programación de la CPU.

9.1. Requisitos de los sistemas de multipropiedad

Los sistemas de multipropiedad suelen requerir las siguientes características adicionales de hardware y software:

- Un número de terminales conectados simultáneamente al sistema para que varios usuarios puedan utilizar simultáneamente el sistema en modo interactivo;

- Una memoria relativamente grande para soportar la multiprogramación;
- Mecanismo de protección de la memoria para evitar que las instrucciones y datos de un trabajo pasen a otros trabajos en un entorno de multiprogramación;
- Mecanismo de preservación del estado de los trabajos para conservar la información completa del estado de un trabajo cuando se le retira la CPU y restaurar esta información antes de que se le devuelva de nuevo la CPU;
- Un algoritmo especial de programación de la CPU que asigna un periodo muy corto de tiempo de CPU uno a uno a cada proceso de usuario de forma circular; y
- Un mecanismo de despertador para enviar una señal de interrupción a la CPU después de cada intervalo de tiempo.

9.2. Ventajas de los sistemas de tiempo compartido

Aunque los sistemas de tiempo compartido son complejos de diseñar, proporcionan varias ventajas a sus usuarios. Las principales ventajas de los sistemas de tiempo compartido son las siguientes:

- **Reduce el tiempo de inactividad de la CPU /** *Reduces CPU idle time*: Mientras un usuario se dedica a pensar o a teclear sus datos, un sistema de tiempo compartido puede dar servicio a muchos otros usuarios. De este modo, los sistemas de tiempo compartido ayudan a reducir en gran medida el tiempo de inactividad de la CPU, aumentando el rendimiento del sistema.
- **Proporciona ventajas de tiempo de respuesta rápido /** *Provides advantages of quick response time*: El algoritmo especial de programación de la CPU utilizado en los sistemas de tiempo compartido garantiza un tiempo de respuesta rápido a todos los usuarios. Esta característica permite a los usuarios interactuar con el sistema más rápidamente mientras trabajan en su problema. Por ejemplo, un sistema de tiempo compartido puede utilizarse eficazmente para la programación y depuración interactivas con el fin de mejorar la eficiencia de los programadores. Varios programadores pueden proceder simultáneamente paso a paso, escribiendo, probando y depurando partes de sus programas o probando varios enfoques para la solución de un problema. La mayor ventaja de un sistema de este tipo es que los errores pueden encontrarse, corregirse y el trabajo puede continuar inmediatamente para todos los usuarios simultáneos del sistema. Esto contrasta con un sistema por lotes en el que los errores se corrigen fuera de línea y el trabajo se vuelve a enviar para otra ejecución. En un sistema por lotes, el tiempo que transcurre entre el envío del trabajo y el retorno del resultado suele medirse en horas.
- **Ofrece buenas prestaciones informáticas a los pequeños usuarios /** *Offers good computing facility to small users*: Los pequeños usuarios pueden acceder directamente a hardware y software mucho más sofisticados de lo que podrían justificar o permitirse de otro modo. En los sistemas de tiempo compartido, se limitan a pagar una cuota por los recursos utilizados y se ven liberados de los problemas de hardware, software y personal asociados a la adquisición y mantenimiento de su propia instalación.

10. Gestión de Archivos

Un archivo es una colección de información relacionada. Cada archivo tiene un nombre, sus datos y atributos. El nombre de un archivo lo identifica unívocamente en el sistema y es utilizado por sus usuarios para acceder a él. Los datos de un archivo son su contenido. El contenido de un archivo es una secuencia de bits, bytes, líneas o registros, cuyo significado definen el creador y el usuario del archivo. Los atributos de un archivo contienen otra información sobre el mismo, como la fecha y hora de su creación, la fecha y hora del último acceso, la fecha y hora de la última actualización, su tamaño actual, sus características de protección, etc. La lista de atributos mencionados para un archivo varía considerablemente de un sistema a otro. El módulo de gestión de archivos de un sistema operativo se encarga de las actividades relacionadas con los archivos, como estructurarlos, acceder a ellos, nombrarlos, compartirlos y protegerlos.

10.1. Métodos de acceso a archivos

Para utilizar la información almacenada en un archivo, es necesario acceder a él y leerlo en la memoria del ordenador. Los dos métodos de acceso a archivos más comunes en el sistema operativo son el acceso secuencial y el aleatorio. A continuación se describen brevemente:

- **Acceso Secuencial /** *Sequential Access*: El acceso secuencial significa que se accede a un grupo de elementos (por ejemplo, datos en una matriz de memoria o un archivo de disco o en una cinta) en una secuencia predeterminada y ordenada. El acceso secuencial es a veces la única forma de acceder a los datos, por ejemplo si están en una cinta. También puede ser el método de acceso elegido, por ejemplo, si simplemente queremos procesar una secuencia de elementos de datos en orden.
- **Acceso aleatorio /** *Random Access*: Los archivos de acceso aleatorio están formados por registros a los que se puede acceder en cualquier secuencia. Esto significa que los datos se almacenan exactamente como aparecen en la memoria, con lo que se ahorra tiempo de procesamiento (porque no es necesaria la traducción) tanto cuando se escribe el archivo como cuando se lee. Los archivos aleatorios son una mejor solución a los problemas de las bases de datos que los archivos secuenciales, aunque tienen algunas desventajas. Por un lado, los archivos aleatorios no son especialmente transportables. A diferencia de los archivos secuenciales, no podemos echar un vistazo dentro de ellos con un editor, ni escribirlos de forma significativa en la pantalla.

Todos los sistemas operativos no admiten tanto los archivos de acceso secuencial como los de acceso aleatorio. Algunos de ellos sólo admiten archivos de acceso secuencial, mientras que otros sólo admiten archivos de acceso aleatorio, si bien hay algunos sistemas operativos que admiten ambos tipos. Aquellos que soportan archivos de ambos tipos, normalmente requieren que un archivo sea declarado como secuencial o aleatorio, cuando es creado; tal archivo puede ser accedido sólo de una manera consistente con su declaración. La mayoría de los sistemas operativos modernos sólo admiten archivos de acceso aleatorio.

10.2. Operaciones con archivos

Un sistema operativo proporciona un conjunto de operaciones para tratar con archivos y sus contenidos. Un conjunto típico de operaciones con archivos proporcionado por un sistema operativo puede ser el siguiente:

- **Crear /** *Create*: Se utiliza para crear un nuevo archivo.
- **Borrar /** *Delete*: Sirve para borrar un archivo existente que ya no se necesita.
- **Abrir /** *Open*: Esta operación se utiliza para abrir un archivo existente cuando un usuario quiere empezar a utilizarlo.
- **Cerrar /** *Close*: Cuando un usuario ha terminado de utilizar un archivo, éste debe cerrarse mediante esta operación.
- **Leer /** *Read*: Sirve para leer los datos almacenados en un archivo.
- **Escribir /** *Write*: Se utiliza para escribir nuevos datos en un archivo.
- **Buscar /** *Seek*: Esta operación se utiliza con archivos de acceso aleatorio para posicionar primero el puntero de lectura/escritura en un lugar específico del archivo, de forma que se puedan leer o escribir datos desde esa posición.
- **Obtener atributos /** *Get Attributes*: Se utiliza para acceder a los atributos de un archivo.
- **Establecer atributos /** *Set Attributes*: Sirve para cambiar los atributos de un archivo que puede configurar el usuario, como el modo de protección.
- **Renombrar /** *Rename*: sirve para cambiar el nombre de un archivo existente.
- **Copiar /** *Copy*: Se utiliza para crear una copia de un archivo, o para copiar un archivo a un dispositivo de E/S como una impresora o una pantalla.

11. Estructura del Sistema Operativo

En esta sección veremos çómo se unen los distintos componentes para formar un sistema operativo". Estos componentes se tratan a continuación:

11.1. Estructura en capas

Un diseño por capas de la arquitectura de un sistema operativo intenta conseguir robustez estructurando la arquitectura en capas con diferentes privilegios. La capa más privilegiada contendría el código que se ocupa de la gestión de interrupciones y el cambio de contexto, las capas superiores seguirían con los controladores de dispositivos, la gestión de memoria, los sistemas de archivos, la interfaz de usuario y, por último, la capa menos privilegiada contendría las aplicaciones. MULTICS es un ejemplo destacado de sistema operativo por capas, diseñado con ocho capas formando anillos de protección, cuyos límites sólo podían cruzarse utilizando instrucciones especializadas. Sin embargo, los sistemas operativos contemporáneos no utilizan el diseño por capas, ya que se considera demasiado restrictivo y requiere un soporte de hardware específico.

La mayoría de los sistemas operativos modernos organizan sus componentes en una serie de capas (niveles), cada una de ellas construida sobre capas inferiores. La capa inferior (capa 0) es el hardware, y la capa superior (capa) es la interfaz de usuario. El número de capas intermedias y su contenido varían de un sistema operativo a otro. La principal ventaja del enfoque por capas es la modularidad. Las capas se seleccionan de forma que cada una de ellas utilice las funciones y servicios que proporciona su capa inmediatamente inferior. Este enfoque simplifica enormemente el diseño y la implementación del sistema, ya que cada capa se implementa utilizando únicamente aquellas operaciones proporcionadas por su capa de nivel inmediatamente inferior.

11.2. Núcleo / Kernel

El kernel es el componente central de la mayoría de los sistemas operativos informáticos; es un puente entre las aplicaciones y el procesamiento real de datos que se realiza a nivel de hardware. Entre las responsabilidades del núcleo se incluye la gestión de los recursos del sistema (la comunicación entre los componentes de hardware y software). Normalmente, como componente básico de un sistema operativo, un kernel puede proporcionar la capa de abstracción de más bajo nivel para los recursos (especialmente procesadores y dispositivos de E/S) que el software de aplicación debe controlar para realizar su función. Normalmente pone estos recursos a disposición de los procesos de aplicación a través de mecanismos de comunicación entre procesos y llamadas al sistema.

Las tareas del sistema operativo se realizan de forma diferente en los distintos kernels, dependiendo de su diseño e implementación. Mientras que los kernels monolíticos ejecutan todo el código del sistema operativo en el mismo espacio de direcciones para aumentar el rendimiento del sistema, los microkernels ejecutan la mayoría de los servicios del sistema operativo en el espacio de usuario como servidores, con el objetivo de mejorar la mantenibilidad y modularidad del sistema operativo. Existe un abanico de posibilidades entre estos dos extremos.

11.3. Núcleo monolítico frente a micronúcleo (Microkernel)

Los dos modelos más utilizados para el diseño de núcleos en sistemas operativos son el núcleo monolítico y el micronúcleo. En un núcleo monolítico, todos los servicios del sistema operativo se ejecutan junto con el hilo principal del núcleo, por lo que también residen en la misma zona de memoria. Este enfoque proporciona un acceso al hardware rico y potente. Algunos desarrolladores, como Ken Thompson, desarrollador de UNIX, sostienen que es “más fácil implementar un núcleo monolítico” que los micronúcleos. Las principales desventajas de los kernels monolíticos son las dependencias entre los componentes del sistema, que un fallo en un controlador de dispositivo puede colapsar todo el sistema, y el hecho de que los kernels grandes pueden llegar a ser muy difíciles de mantener.

El enfoque del micronúcleo consiste en definir una abstracción sencilla sobre el hardware, con un conjunto de primitivas o llamadas al sistema para implementar servicios mínimos del SO como la gestión de memoria, la multitarea y la comunicación entre procesos. Otros servicios, incluidos los que normalmente proporciona el núcleo, como las redes, se implementan en programas de espacio de usuario, denominados servidores.

Los micronúcleos son más fáciles de mantener que los monolíticos, pero el gran número de llamadas al sistema

y de cambios de contexto puede ralentizar el sistema porque suelen generar más sobrecarga que las llamadas a funciones simples.

Un micronúcleo permite implementar la parte restante del sistema operativo como un programa de aplicación normal escrito en un lenguaje de alto nivel, y utilizar distintos sistemas operativos sobre el mismo núcleo inalterado. También es posible cambiar dinámicamente entre sistemas operativos y tener más de uno activo simultáneamente.

A medida que crece el núcleo informático, se hacen evidentes una serie de problemas. Uno de los más obvios es que aumenta la huella de memoria. Esto se mitiga hasta cierto punto perfeccionando el sistema de memoria virtual, pero no todas las arquitecturas informáticas tienen soporte de memoria virtual. Para reducir la huella del kernel, hay que realizar una edición exhaustiva para eliminar cuidadosamente el código innecesario, lo que puede resultar muy difícil con interdependencias no evidentes entre partes de un kernel con millones de líneas de código. A principios de los 90, debido a las diversas deficiencias de los núcleos monolíticos frente a los micronúcleos, prácticamente todos los investigadores de sistemas operativos consideraban obsoletos los núcleos monolíticos. Como resultado, el diseño de Linux como un núcleo monolítico en lugar de un micronúcleo fue el tema de un famoso debate entre los famosos científicos Linus Torvalds y Andrew Tanenbaum. Hay mérito en ambos lados del argumento presentado en el debate de Tanenbaum y Torvalds.

11.4. Módulos del sistema operativo residentes y no residentes

Con todas las funcionalidades de un sistema operativo implementadas, se convierte en un gran software. Obviamente, todas las funcionalidades de un sistema operativo no se necesitan todo el tiempo. Como la capacidad de la memoria principal de un sistema es limitada, es habitual mantener siempre en la memoria del sistema sólo una parte muy pequeña del sistema operativo y guardar la parte restante en un dispositivo de almacenamiento en línea, como un disco duro. Los módulos de un sistema operativo que se mantienen siempre en la memoria principal del sistema se denominan módulos residentes y los que se mantienen en el disco duro se denominan módulos no residentes. Los módulos no residentes se cargan en la memoria bajo demanda, es decir, cuando se necesitan para su ejecución.

No hay que confundir el núcleo del sistema con los modelos residentes del sistema operativo. Ambos no son necesariamente lo mismo. De hecho, para la mayoría de los sistemas operativos son diferentes. Los dos criterios siguientes determinan normalmente si un módulo concreto del sistema operativo debe ser residente:

- Su frecuencia de uso, y
- Si el sistema puede funcionar sin él.

12. Otros conceptos relacionados

A continuación se discuten brevemente otros conceptos importantes relacionados con los sistemas operativos:

12.1. Sistema operativo en tiempo real

Un Sistema Operativo en Tiempo Real (RTOS) es un sistema operativo (OS) destinado a servir peticiones de aplicaciones en tiempo real. Una característica clave de un RTOS es su nivel de consistencia en cuanto a la cantidad de tiempo que tarda en aceptar y completar la tarea de una aplicación; la variabilidad es el jitter. Un sistema operativo en tiempo real duro tiene menos jitter que un sistema operativo en tiempo real blando. El principal objetivo de diseño no es un alto rendimiento, sino garantizar una categoría de rendimiento suave o dura. Un RTOS que normalmente o generalmente puede cumplir un plazo es un SO en tiempo real blando, pero si puede cumplir un plazo de forma determinista es un SO en tiempo real duro. Un sistema operativo en tiempo real tiene un algoritmo avanzado de programación.

La flexibilidad del programador permite una orquestación más amplia de las prioridades de los procesos del sistema informático, pero un SO en tiempo real se dedica con más frecuencia a un conjunto reducido de aplicaciones. Los factores clave en un SO en tiempo real son una latencia mínima de interrupción y una latencia mínima de cambio de hilos, pero un SO en tiempo real se valora más por la rapidez o previsibilidad de su respuesta que por

la cantidad de trabajo que puede realizar en un periodo de tiempo determinado. Algunos ejemplos de este tipo de aplicaciones son:

- Una aeronave debe procesar los datos del acelerómetro dentro de un periodo determinado (digamos cada 20 milisegundos) que depende de las especificaciones de la aeronave. Si no lo hace, la aeronave podría desviarse de su rumbo correcto o incluso estrellarse.
- No responder a tiempo a una condición de error en una central térmica de un reactor nuclear podría provocar una fusión.
- No responder a tiempo a una condición de error en la cal de montaje de una fábrica automatizada podría dar lugar a que varias unidades de producto tuvieran que desecharse en última instancia.
- Una solicitud de reserva de billete en un sistema informatizado de reservas ferroviarias debe procesarse dentro de la percepción de los pasajeros de un tiempo razonable.

12.2. Sistemas operativos distribuidos

Un sistema operativo distribuido es la agregación lógica del software del sistema operativo en una colección de nodos informáticos independientes, conectados en red, comunicados y diseminados espacialmente. Cada uno de los nodos del sistema contiene un subconjunto de software discreto del sistema operativo global agregado. Cada subconjunto de software a nivel de nodo es una composición de dos proveedores de servicios distintos. El primero es un núcleo mínimo ubicuo, o micronúcleo, situado directamente sobre el hardware de cada nodo. El micronúcleo sólo proporciona los mecanismos necesarios para la funcionalidad de un nodo. El segundo es un conjunto de componentes de gestión del sistema de nivel superior, que proporciona todas las políticas necesarias para las actividades individuales y colaborativas de un nodo. Esta colección de componentes de gestión se encuentra inmediatamente por encima del micronúcleo, y por debajo de cualquier aplicación de usuario o API que pueda residir en niveles superiores.

Estas dos entidades, el micronúcleo y la colección de componentes de gestión, trabajan juntas. Apoyan el objetivo del sistema global de integrar sin fisuras todos los recursos conectados a la red y la funcionalidad de procesamiento en un sistema eficiente, disponible y unificado. Esta integración sin fisuras de los nodos individuales en un sistema global se conoce como transparencia, o imagen de sistema único; describe la ilusión proporcionada a los usuarios de la apariencia del sistema global como una entidad computacional singular y local. Los sistemas operativos utilizados habitualmente para los sistemas informáticos distribuidos pueden clasificarse a grandes rasgos en dos tipos: sistemas operativos de red y sistemas operativos distribuidos. Las tres características más importantes utilizadas habitualmente para diferenciar estos dos tipos de sistemas operativos son la imagen del sistema, la autonomía y la capacidad de tolerancia a fallos. Estas características se explican a continuación:

- **Imagen del sistema /** *System Image*: La característica más importante utilizada para diferenciar entre los dos tipos de sistema operativo es la imagen del sistema informático distribuido desde el punto de vista de sus usuarios. En el caso de un sistema operativo de red, los usuarios ven el sistema informático distribuido como un conjunto de máquinas distintas conectadas por un subsistema de comunicación. Es decir, los usuarios son conscientes de que se están utilizando varios ordenadores. Por otro lado, un sistema operativo distribuido oculta la existencia de múltiples ordenadores y proporciona una única imagen del sistema a sus usuarios. Es decir, hace que una colección de máquinas conectadas en red aparezca ante sus usuarios como un uniprocesador virtual proporcionando un tipo de interfaz de usuario similar al que ofrece el sistema operativo centralizado.
- **Autonomía /** *Autonomy*: En un funcionamiento en red, cada ordenador del sistema informático distribuido tiene su propio sistema operativo local (los sistemas operativos de los distintos ordenadores pueden ser iguales o diferentes), y no existe esencialmente ningún tipo de coordinación entre los ordenadores, salvo la regla de que cuando dos procesos de distintos ordenadores se comunican entre sí, deben utilizar un protocolo de comunicación acordado mutuamente. Cada ordenador funciona independientemente de los demás, en el sentido de que cada uno toma decisiones independientes sobre la creación y finalización de sus propios procesos y la gestión de los recursos locales. Cabe señalar que, debido a la posibilidad de que existan diferencias en los sistemas operativos locales, la llamada al sistema desde distintos ordenadores del mismo sistema informático distribuido puede ser diferente en este caso.

Por otro lado, con un sistema operativo distribuido, existe un único sistema operativo para todo el sistema y cada ordenador del sistema informático distribuido ejecuta una parte de este sistema operativo global. El sistema operativo distribuido entrelaza estrechamente todos los ordenadores del sistema informático distribuido en el sentido de que trabajan en estrecha cooperación entre sí para la utilización eficiente y eficaz de los diversos recursos del sistema. Es decir, los procesos y varios recursos se gestionan globalmente (algunos recursos se gestionan localmente). Además, en todos los ordenadores del sistema informático distribuido existe un único conjunto de llamadas al sistema válidas en todo el mundo.

- **Capacidad de tolerancia a fallos** / *Fault tolerance capability*: Un sistema operativo de red ofrece poca o ninguna capacidad de tolerancia a fallos, en el sentido de que si el 10 % de las máquinas de todo el sistema informático distribuido se caen en un momento dado, al menos el 10 % de los usuarios no podrán continuar con su trabajo. Por otro lado, con un sistema operativo distribuido, la mayoría de los usuarios normalmente no se ven afectados por las máquinas averiadas y pueden seguir realizando su trabajo con normalidad, con sólo una pérdida del 10 % en el rendimiento de todo el sistema informático distribuido. Por lo tanto, la capacidad de tolerancia a fallos de un sistema operativo distribuido suele ser muy alta en comparación con la de un sistema operativo de red.

En resumen, tanto los sistemas operativos de red como los sistemas operativos distribuidos trabajan con varios ordenadores interconectados mediante una red de comunicaciones. En el caso de un sistema operativo de red, el usuario ve el sistema como un conjunto de ordenadores distintos, pero en el caso de un sistema operativo distribuido, el usuario ve el sistema como un "uniprocesador virtual".