GRADO EN INGENIERÍA INFORMÁTICA

Módulo de Formación Básica

SISTEMAS OPERATIVOS

D. Ricardo González García

viu Universidad Internacional de Valencia

Edita
Universidad Internacional de Valencia

Grado en

Ingeniería Informática

Sistemas operativos

Módulo de Formación Básica

6ECTS

D. Ricardo González García

Índice

Leyenda

Glosario
Términos cuya definición correspondiente está en el apartado "Glosario".

Tema 1.
Introducción a los Sistemas Operativos

Tenemos que decir que el **sistema operativo** es el más importante de los programas de sistemas y posee tres objetivos fundamentales, el primero de ellos es facilitar que los usuarios puedan usar los recursos del ordenador; el segundo, administrar los recursos disponibles en el ordenador, para poder ejecutar las **aplicaciones** y los **programas** de los usuarios y; el tercero, servir de interfaz para que los usuarios puedan interactuar con el ordenador, solicitar información de este y entregar programas para que, al ser ejecutados, se satisfagan las necesidades de información de los usuarios.

Los **programas** que vamos a usar en un ordenador se pueden clasificar en **dos grandes familias o grupos**: en el primer grupo están los programas o aplicaciones de usuario que son programas que le brindan resultados directos a los usuarios de los ordenadores, para satisfacer sus necesidades de información. Ejemplos de este tipo de programa son: los editores de texto, los programas de correo electrónico, las hojas de cálculo, los juegos, las calculadoras, agendas y otras herramientas de productividad.

En el segundo grupo se encuentran los **Programas de Sistema** que son programas que sirven de soporte para el funcionamiento del ordenador y que se usan para apoyar o complementar la ejecución de otros programas. En este grupo están los sistemas operativos, pero también otros elementos como: los cargadores, la BIOS, los drivers y controladores, las librerías graficas como OPENGL, los entornos

de escritorios, los sistemas de ventanas, los compiladores, los administradores y las herramientas de configuración; todos estos programas, librerías y herramientas nos ayudan a manejar el ordenador, pero no dan respuesta directa a los requerimientos de información de los usuarios. En este sentido, el sistema operativo es el principal y más importante de los programas de sistema que está en el ordenador.

1.1. Funciones principales de un Sistema Operativo

Para comprender de la mejor manera posible qué es un sistema operativo, debemos identifica claramente cuáles son sus funciones principales y cómo lo usamos, en este sentido, es posible resumir estas funciones principales en tres grandes grupos de actividades que pasamos a describir a continuación:

1.1.1. Facilitar o permitir que los programas usen los recursos del ordenador

Varios de los componentes de hardware que posee el ordenador poseen un nivel de complejidad importante en su operación, por ejemplo, para poder escribir una letra en una pantalla del ordenador, lo que es muy común, se deben encender algunos píxeles en un patrón especifico en la pantalla, pero si los píxeles adecuados no se encienden, es posible que una letra "b" se convierta en una "p" o en una "o".

Si el usuario, cada vez que necesitara escribir una letra, debiera saber qué píxeles en la pantalla encender y cuáles no, seguramente mandaría muy pocos mensajes. Es por ello, que el sistema operativo contiene una serie de rutinas que el usuario invoca para imprimir una letra y que se encargan de identificar donde está el cursor en la pantalla del ordenador en ese momento, es decir, por donde se está escribiendo en la pantalla en ese instante; luego, se encargan de encender los píxeles adecuados para representar una letra y, también, de dejar el espacio adecuado para que la siguiente letra no esté pegada a la anterior, de forma que se pueda entender y leer adecuadamente el mensaje.

Otro ejemplo es cuando queremos almacenar en el computador un archivo que contiene los nombres y los teléfonos de las personas que vamos a invitar a una reunión. Queremos almacenar la lista porque podemos usarla luego para otras futuras reuniones, entonces le decimos al ordenador que la guarde en una carpeta reuniones, dentro de otra carpeta donde tenemos información personal, ahora, en qué parte del disco duro se almacena realmente esa información y cómo podemos recuperarla rápidamente cuando la volvamos a necesitar, es algo que realmente no sabemos exactamente cómo se hace. Es el sistema operativo el encargado tanto de guardar adecuadamente en el disco duro del ordenador los datos, como de recuperarlos luego cuando los volvamos a necesitar. Estos son dos de entre los muchos servicios que el sistema operativo brinda, para que sea mucho más sencillo para los usuarios utilizar el hardware del ordenador.

1.1.2. Brindar una interfaz para que el usuario pueda manejar el ordenador

El sistema operativo debe brindar un espacio o ventana a través de la cual el usuario le indique qué es lo que desea hacer, tal como se ilustra en la Figura 1.1. Para que el usuario pueda especificar qué información requiere o qué programas se van a ejecutar, esta interfaz puede consistir de una interfaz

de texto en la que al usuario se le abre un espacio en una ventana para que en ella escriba comandos de textos con las instrucciones, rutinas y programas que desea ejecutar, o también puede estar compuesta por elementos gráficos y ventanas en las que, mediante una serie de iconos, elementos gráficos, ventanas y aplicaciones, el usuario puede, al hacer clic con el ratón, seleccionar la información que requiere, o solicitar la ejecución de determinados programas, o abrir ciertos ficheros de datos.

Todos estos son elementos que permiten que el usuario pueda expresarle al ordenador qué es lo que desea. Recordemos que el ordenador realmente habla es un lenguaje electrónico en el cual la presencia o ausencia de electricidad es la que marca el inicio de acciones y del flujo de información, pero a nosotros como seres humanos se nos dificulta hablar y entender este lenguaje de electricidad o incluso un lenguaje binario de ceros y unos, en el que se marca la presencia o no de electricidad en el dispositivo, por eso requerimos que estas señales sean traducidas a señales textuales, visuales o auditivas que sí podamos entender y con las cuales podamos interactuar para hacer uso del ordenador.

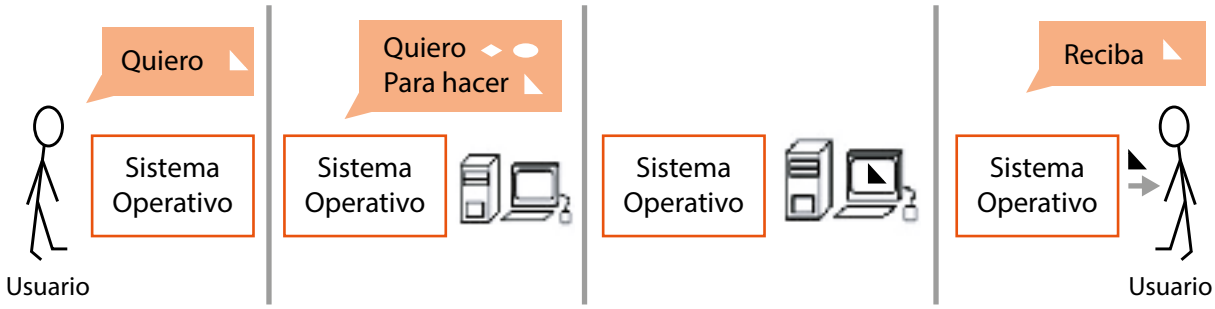


Figura 1.1. El Sistema Operativo como una interfaz para manejar el ordenador. Fuente: elaboración propia.

1.1.3. Administrar los recursos del ordenador

El ordenador contiene una cantidad de recursos que deben ser administrados cuidadosamente, en especial cuando hay recursos que son compartidos por varios programas o usuarios. En este sentido, el sistema operativo no solo debe permitir que los programas y **procesos** usen adecuadamente estos recursos, sino que también debe proteger o aislar la ejecución de procesos, de forma que lo que se haga sea hecho de forma segura, donde un programa no entorpezca o moleste el trabajo que están haciendo otros programas, o incluso que un programa no le llegue a quitar los recursos que otro programa necesita para su ejecución o, si lo hace, que esto sea por una razón importante y válida.

Por ejemplo, uno de los recursos importantes del ordenador es la **CPU** o unidad central de proceso, que es como el cerebro del ordenador. El sistema operativo debe saber en todo momento qué programa es el que se está ejecutando en la CPU y después de que el programa que se esté ejecutando termine, el sistema operativo debe identificar qué otro programa será tomado para continuar con su ejecución; esta es una actividad que el sistema operativo va realizando, de forma automática, en el ordenador en cada momento. Las instrucciones y programas que se encargan de estas tareas de forma automática forman parte del sistema operativo del computador y, más específicamente, del despachador y del **planificador de procesos**.

Otro de los recursos del ordenador es la **memoria**, cada programa usa una porción de memoria para almacenar los datos que va procesando y los resultados que va obteniendo. El sistema operativo es el

programa que se encarga de administrar estos espacios de memoria y de asignar más memoria cuando un proceso lo solicite, así como de verificar que un programa P1 no esté usando algún espacio de memoria que corresponda a otro programa Pk, porque de ser así, el programa P1 puede, al modificar uno de estos espacios de memoria, cambiar o entorpecer el trabajo que el programa Pk está haciendo con sus datos, lo que puede llegar a ocasionar que el programa Pk no pueda continuar su ejecución, o incluso que la finalice de forma anormal.

Como se ha comentado, un sistema operativo debe poder manejar los programas de los usuarios y los recursos que estos requieran para ejecutarse. Para lograr realizar todas estas actividades se debe contar son una serie de manejadores como son: el manejador de procesos, el manejador de memoria, el manejador de dispositivos y el manejador de ficheros o archivos. Estos componentes, junto con una interfaz de usuario, son los que permiten al sistema operativo realizar su trabajo de administración, control y gestión de sus usuarios y los recursos del ordenador.

Durante este curso, estaremos tratando cada uno de estos componentes para comprender mejor cómo trabajan y cómo permiten que los usuarios tengamos una mejor experiencia cuando trabajemos con los recursos de un ordenador.

1.2. Formas de trabajo en un sistema operativo

Los primeros ordenadores que existieron eran equipos muy simples y poco fiables que apenas alcanzaban a hacer algunas decenas o cientos de operaciones por segundo. En estos primeros ordenadores no existía la noción de sistema operativo, tal como hoy lo conocemos, por lo que tanto los usuarios como los programadores debían tomar el control de todas las operaciones del ordenador para ejecutar un programa a la vez, indicándole al ordenador qué pasos debía seguir para ejecutar cada programa, suministrando tanto los programas como los datos que estos programas requirieran y llevando el mismo el orden y el control de todos los aspectos del uso del ordenador.

Como la ejecución de estos primeros programas podía tomar una cantidad importante de tiempo, los usuarios no solo entregaban los programas, sino que, también muchas veces, los conjuntos de datos de entrada que estos programas debían procesar e indicaban cómo debían obtener los resultados. A esta unidad de trabajo que contenía programas, datos de entrada y especificaciones de la salida de los resultados que el computador podía tomar para ejecutar sin que mediara la intervención del usuario, se le dio el nombre de unidad de **Trabajo** o **Job**. A medida que los ordenadores fueron avanzando tecnológicamente, tuvieron la capacidad de ir haciendo más operaciones, al punto que podían ir procesando aplicaciones a una velocidad mucho mayor, a la que los usuarios podían usar para entregarles cada uno de sus trabajos. En este punto surge la idea de entregarle al ordenador paquetes completos con la especificación de varias actividades o trabajos por hacer, es aquí donde nace el procesamiento de trabajos por lotes.

1.2.1. Procesamiento por Lotes o Batch

En la medida en que los ordenadores estuvieron en disposición de atender rápidamente los trabajos de los usuarios, estos se dieron cuenta de que podían hacer actividades más complejas e interesantes en las que, incluso, la salida de un trabajo podía llegar a ser la entrada del próximo trabajo a realizar.

Al poder estructurar programas de esta forma, es posible combinar una serie de programas simples para automatizar actividades más complejas y de más alto nivel. Sin embargo, también podía ocurrir que la ejecución de un programa no diera el resultado esperado y entonces se debía seguir un curso de acción diferente con los datos obtenidos. Todo esto implica el hecho de coordinar un conjunto elaborado de especificaciones para que el ordenador pueda procesar diferentes programas y unidades de información, con el objetivo de cumplir los requerimientos de una aplicación en particular. A este tipo de procesamiento en el que un conjunto de diferentes programas procesaba una serie de datos con el objetivo de lograr obtener información útil al final del proceso, de forma automática y sin la intervención directa de los usuarios, se le llamó **Procesamiento por Lotes** o *batch*.

En este tipo de procesamiento, el usuario entrega sus programas, los datos iniciales y un conjunto de instrucciones de procesamiento al ordenador para que sea el ordenador el que realice todo el trabajo de ejecución por su cuenta y de forma automática, para que el usuario pueda, más tarde, volver a ver los resultados de los trabajos que ha entregado para su ejecución, tal como se ilustra en la Figura 1.2.

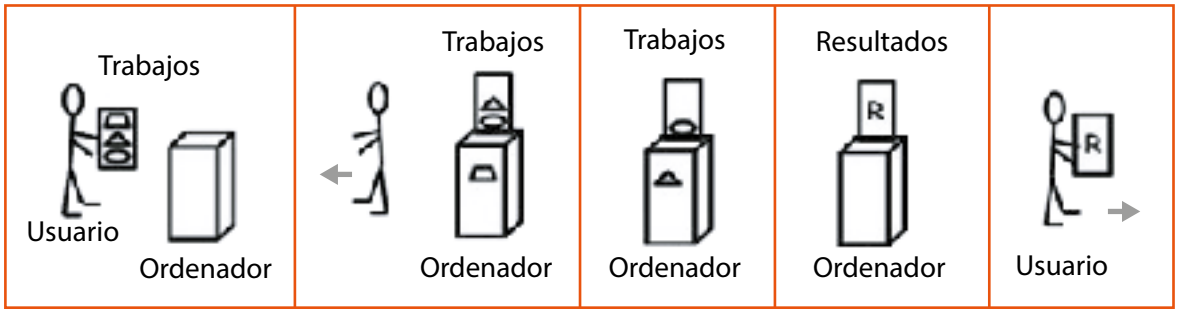


Figura 1.2. En el procesamiento por lotes el usuario entrega un lote de actividades que el ordenador ejecuta de forma autónoma y cuando los resultados están disponibles el usuario vuelve a recogerlos. Fuente: elaboración propia

1.2.2. Procesamiento interactivo o en línea (on line)

Existen algunos programas en los que las actividades que se realizan son muy cambiantes y requieren de la constante participación de los usuarios para ingresar datos o para verificar que realmente el programa está haciendo lo que se supone que debe hacer, como se ilustra en figura 1.3.

También existen casos en los que es difícil predecir todos los rumbos de ejecución que puedan ocurrir en un programa. En cualquiera de estas situaciones se requiere que, durante la ejecución de los programas, los usuarios deban estar alimentando nuevos datos, viendo los resultados parciales que se van obteniendo e indicando posibles cambios en el procesamiento que se está realizando para ajustarlo a las necesidades de información que existan. Todo esto implica que al usuario no siempre le basta con entregarle al computador los programas y datos para regresar luego a ver los resultados que se obtuvieron. En cambio, hay situaciones en las que la interacción del usuario con los programas es una necesidad; esto, unido a la introducción de los primeros teclados y pantallas en los ordenadores, permitió la creación del **Procesamiento Interactivo** o En Línea, en el que el usuario constantemente recibe información del computador a través de una pantalla y está en la capacidad de introducir nuevos datos o cambiar el flujo de ejecución de los programas al ingresar comandos mediante un teclado, ratón, una pantalla táctil o uno de los muchos dispositivos que están disponibles en el mercado para la introducción de datos en el ordenador.

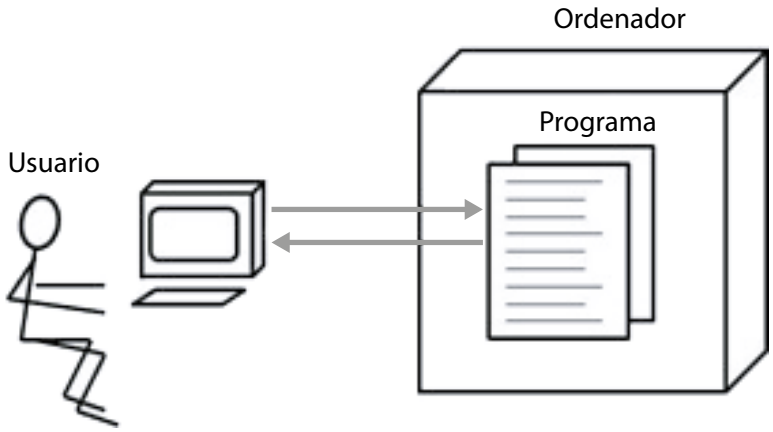


Figura 1.3. Sistema de Procesamiento Interactivo. Fuente: elaboración propia.

1.2.3. Sistemas Multiprogramados

En la medida en que los computadores fueron incrementando su capacidad de procesamiento, estuvieron en la capacidad de tomar un grupo o lote de programas e irlo ejecutando de forma secuencial, uno después de otro. Sin embargo, en muchos casos, uno de los programas debía esperar a leer datos en un disco de almacenamiento o esperar la entrada de datos de un usuario que aún estaba pensando qué información ingresar. En estos casos, el ordenador podía llegar a estar ocioso, sin hacer nada, hasta que se solventara el problema y se pudieran entregar los datos que se requirieran para continuar la ejecución del programa. Pero, en lugar de no hacer nada, una mejor opción es detener momentáneamente la ejecución del programa que está a la espera de algo, almacenar todo el trabajo que se haya estado haciendo hasta ese momento de forma que no se pierda el trabajo realizado, para que se pueda continuar luego, justo en el punto en el que se había quedado.

Finalmente, se puede tomar otro programa y empezarlo a ejecutar, de forma que este nuevo programa pueda ir avanzando en su ejecución, al menos hasta que el primero de los programas pueda estar listo para volver a continuar. De esta forma, se crearon los primeros **Sistemas Multiprogramados** en los que el ordenador estaba en la capacidad de contener varios programas en diferentes estados de ejecución, tal como se observa en la figura 1.4. En esta situación, cuando uno de los programas no pueda continuar, se sigue con otro programa, de forma que el ordenador pueda estar ocupado la mayor parte del tiempo. De esta manera, la totalidad de los trabajos asignados se pueden ejecutar en un menor lapso de tiempo al aprovechar de una mejor forma los recursos disponibles en el ordenador.

Sin embargo, el hecho de tener varios programas ejecutándose implica que el ordenador debe estar atento de qué va haciendo cada programa y garantizar que cada uno de ellos cuenta con los recursos que requiere para su ejecución. Además, la ejecución de un programa no debe perjudicar la ejecución de ninguno de los otros programas que están contenidos en el ordenador, es decir, que el sistema debe garantizar que cada programa se va a ejecutar de forma muy similar a como lo haría si estuviera él solo en el ordenador, sin ningún otro programa que lo moleste.

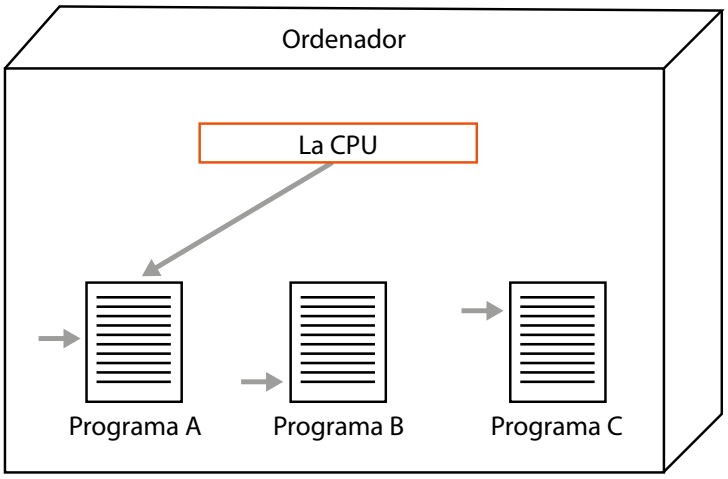


Figura 1.4. Sistemas Multiprogramados. Fuente: elaboración propia.

1.2.4. Sistemas Multiusuarios

Una vez que un computador puede contener varios programas al mismo tiempo, el siguiente paso es la posibilidad de tener diferentes programas pero que pertenezcan a diferentes usuarios. Los **Sistemas Multiusuarios** son aquellos en los que diferentes usuarios pueden estar ejecutando sus programas, tal como se muestra en la figura 1.5, de forma que a cada usuario se le brida la ilusión de que los recursos del ordenador están a su disposición.

Para poder lograr esto, es importante que cada usuario cuente con un espacio propio y se le asignen los recursos que se requieran para que pueda ejecutar sus programas, pero los espacios de cada usuario deben controlarse, de forma que la ejecución de un programa de un usuario no afecte negativamente la ejecución de programas que puedan ejecutar otros usuarios. Esto implica que el sistema operativo debe poder identificar a cada usuario de forma diferente, repartir los recursos que administra entre estos usuarios y, cuando un usuario termine, reasignar los recursos que se liberen de la forma adecuada para que todos los usuarios sientan que se les está dando el tratamiento adecuado a sus programas.

Además, se hace necesario separar o aislar la ejecución de cada programa para evitar que la ejecución de un programa genere efectos colaterales negativos en la ejecución de programas de otros usuarios; en otras palabras, se duplican o se incrementan los controles que un sistema mutiprogramado debe tener para agregar el hecho de trabajar ahora con programas pertenecientes a diferentes usuarios.

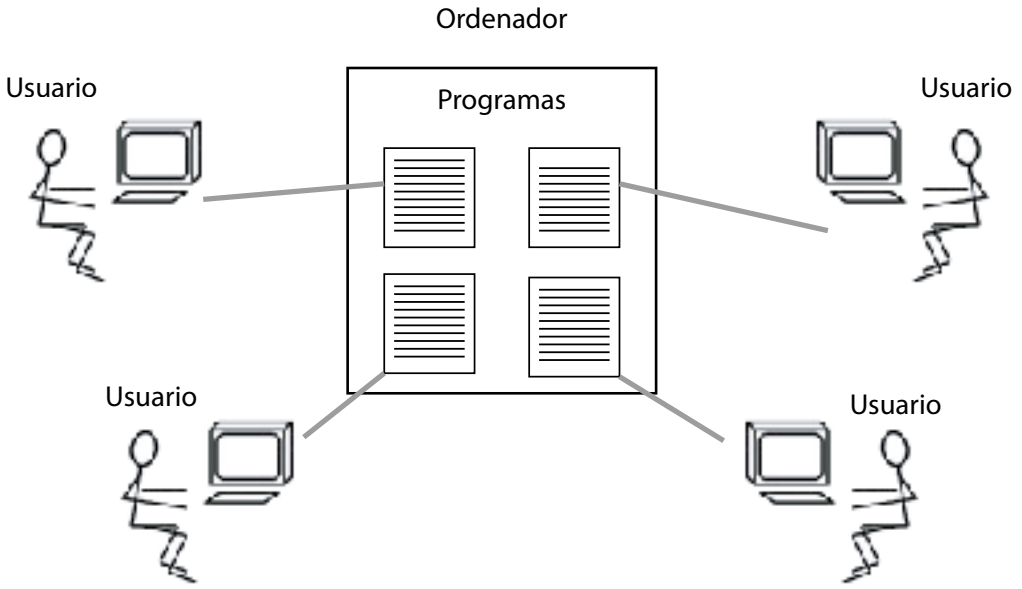


Figura 1.5. Sistemas Multiusuario. Fuente: elaboración propia.

Además de esto, en un sistema multiusuario, diferentes usuarios pueden tener diferentes perfiles de uso que estén asociados a tener permisos sobre diferentes recursos del ordenador. Por ejemplo, un usuario administrador puede requerir acceso prácticamente a todos los recursos del sistema, mientras que un usuario normal puede requerir solo algunos recursos con los que cuente el sistema y, a un usuario invitado, es decir, un usuario ocasional, se le pueden asignar una cantidad mínima de recursos, solo suficientes para que pueda ejecutar algunos programas sencillos. Bajo este mismo enfoque, si diferentes usuarios solicitan un mismo recurso, tendrán prioridad en la asignación de estos recursos los usuarios que tengan más importancia en el sistema.

1.2.5. Sistemas de Tiempo Compartido o Time Sharing

Originalmente, los programas se ejecutaban uno cada vez y se pasaba al siguiente programa una vez que el primero terminaba, o cuando era suspendido, porque estaba a la espera de algún recurso que aún no tenía. Bajo este esquema, aunque podían existir diferentes usuarios ejecutando sus programas, empezaron a ocurrir inconvenientes cuando algunos programas tardaban mucho tiempo en ejecutarse y retrasaban de forma importante el inicio de la ejecución de los otros.

Para solventar este tipo de situaciones y ser un poco más justo con el reparto de los recursos del ordenador, se crearon los **Sistemas de Tiempo Compartido**. En este tipo de sistemas, varios usuarios podían estar ejecutando sus programas porque lo que hace el sistema operativo es repartir **el tiempo de ejecución** en intervalos que pueden ser iguales para cada uno de los programas que se ejecutan, tal como se puede apreciar en la figura 1.6, para que todos puedan ir avanzando al menos un poco, de manera tal que externamente parece que todos se están ejecutando de forma más o menos simultánea, aunque realmente lo que ocurre es que el sistema operativo toma un programa, lo ejecuta por una porción de tiempo que recibe el nombre de **quantum** y, luego, se suspende ese programa para tomar otro, ejecutarlo a su vez por un quantum de tiempo y seguir así con todos los programas restantes. Una vez que se ha llegado al final de la lista de programas, se vuelve a comenzar con el primero.

Si el quantum de tiempo no es muy grande, se atiende a todos los procesos del sistema en unos pocos segundos, por lo que la visión que cada usuario tiene es que el ordenador le está dando servicio todo el tiempo, aunque realmente esté compartiendo su tiempo con otros programas. Este tipo de sistema es de especial utilidad cuando en el sistema hay muchos programas interactivos que requieren constantemente de la participación de sus usuarios para introducir datos, analizar resultados y controlar los procesos que se están ejecutando, ya que en este tipo de sistema no es conveniente que un programa deba esperar mucho a que otro de los programas culmine su ejecución.

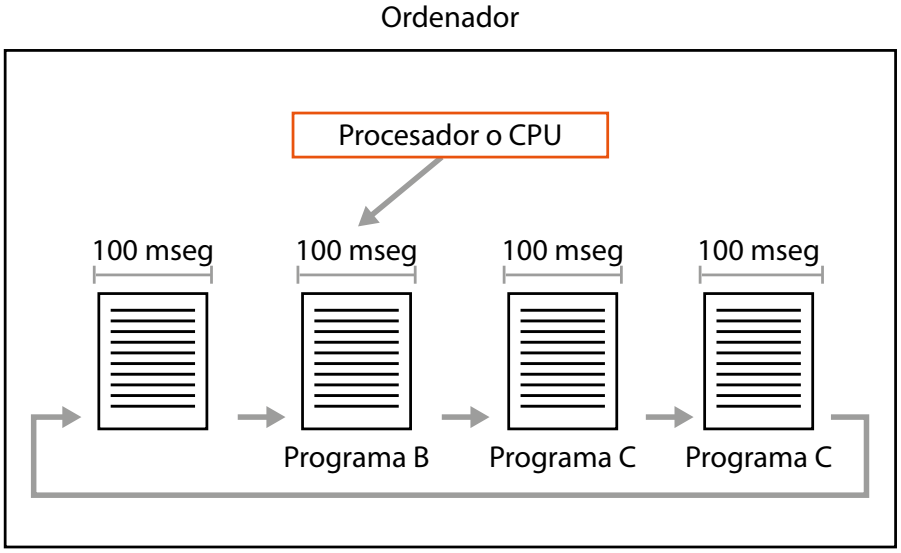


Figura 1.6. Sistemas de Tiempo Compartido. Fuente: elaboración propia.

1.2.6. Sistemas Multiprocesadores y Paralelismo

Buscando que los ordenadores tuvieran cada vez una mayor capacidad de procesamiento para poder hacer más tareas en un menor tiempo, han sido utilizadas una serie de estrategias Una de ellas ha sido usar componentes más veloces y de mayor capacidad, sin embargo, no siempre es posible conseguir componentes más rápidos; esto es especialmente cierto con los procesadores o CPUs, por lo que una forma alternativa ha sido que los ordenadores tengan más de una CPU. Un ordenador con más de una CPU recibe el nombre de un **Sistema Multiprocesador**.

Cuando se le coloca más de una CPU a un ordenador, es posible tener dos o más programas ejecutándose exactamente al mismo tiempo, como se puede ver en la figura 1.7, pero cuando esto ocurre, hay que garantizar que los dos programas sean totalmente independientes el uno del otro y que no requieran ningún recurso común que sea difícil de compartir.

En un computador con una sola CPU, a dos programas que se estén ejecutando más o menos de forma simultánea se les llama **Programas Concurrentes**. Aquí estamos indicando que es de forma más o menos simultánea porque, en realidad, una CPU solo puede estar ejecutando un programa a la vez. Sin embargo, si uno de los programas se detiene sin haber aun terminado su ejecución, puede ser sacado de la CPU junto con todo su estado de ejecución para atender a otro programa hasta que el primero esté listo para continuar; es por esto que es posible tener en una misma CPU dos o más programas en diferentes estados de ejecución. Sin embargo, si el ordenador posee varias CPUs, es

posible que cada CPU esté ejecutando un programa diferente; en este caso se dice que los programas se están ejecutando en **paralelo**.

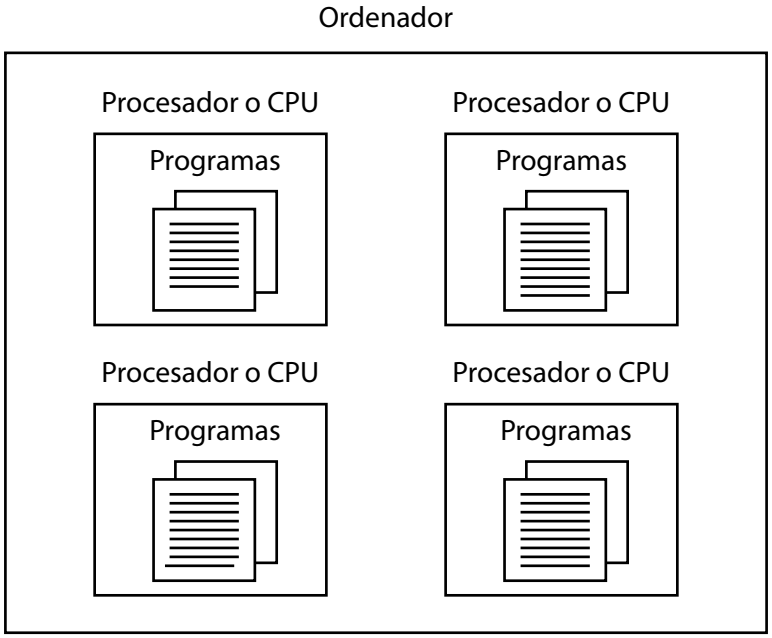


Figura 1.7. Sistema Multiprocesador. Fuente: elaboración propia.

1.2.7. Sistemas en Red

Con el surgimiento de las redes de computadores y, luego, de Internet, cada vez fue más importante que los ordenadores pudieran conectarse a otros ordenadores para compartir información y recursos. Dentro de este esquema, ha surgido el **Modelo Cliente/Servidor** en el que un ordenador, el **Servidor**, posee una serie de recursos que puede compartir con una serie de usuarios en otros ordenadores. A los equipos de estos otros usuarios se les da el nombre de **Clientes** y son los equipos que requieren y solicitan la información que el Servidor puede contener.

El surgimiento de este nuevo modelo de trabajo ha cambiado los requerimientos de los sistemas operativos porque permite que el Servidor pueda atender a una mayor cantidad de usuarios, muchos de los cuales pueden, incluso, estar geográficamente bastante lejos del servidor. Esto ha exigido de forma importante a los servidores que deban hacer un uso más eficiente de los recursos para poder atender de forma adecuada a una cantidad cada vez más grande de usuarios, pero también ha cambiado la forma en la que los ordenadores de los clientes pueden trabajar, dado que ya no requieren que todos los servicios estén de forma local en el cliente, ya que este puede tener acceso a recursos y servicios que pueden estar ahora en uno o varios servidores remotos.

Hoy en día, ya que prácticamente todos los ordenadores están conectados a alguna red, es muy difícil que un ordenador pueda satisfacer todos los requerimientos de un usuario sin estar conectado a la Internet, ya que esta nos permite consultar cualquier duda mediante un navegador web, acceder a datos en cualquier parte del mundo y comunicarnos e intercambiar información con cualquier persona; desde la que está justo a nuestro lado, hasta la que puede estar al otro lado del mundo, con la misma facilidad y prácticamente con el mismo costo.

1.2.8. Sistemas Distribuidos

Los sistemas en red han tenido un éxito impresionante, pero para usarlos tenemos que saber que la información que estamos buscando está en otro lugar, e incluso debemos identificar con qué servidor en específico queremos conectarnos. Esto lo hacemos, por ejemplo, al escribir una url en un navegador web para consultar una información, donde la primera parte de la url identifica al dominio y al ordenador específico, es decir, dónde está ubicado el servidor con el que nos queremos comunicar. Sin embargo, en algunos sistemas quisiéramos poder acceder a la información, sin tener que estar al tanto de si la información está en el mismo ordenador que estamos usando o si está en un servidor a miles de kilómetros de distancia. Aquellos sistemas en los que es totalmente transparente el uso de los recursos, sin importar si son locales o no, se les llamada **Sistemas Distribuidos**. Este tipo de sistemas son básicamente sistemas en red donde los usuarios pueden estar usando recursos de otros ordenadores sin saber que lo están haciendo, y accediendo a ellos de la misma forma a como lo hacen con los recursos locales de su propio ordenador.

1.3. Arquitectura o estructura general de un Sistema Operativo

La arquitectura de un sistema establece cuáles son los componentes constitutivos de ese sistema y cómo estos componentes interactúan y se relacionan para que el sistema pueda cumplir con su objetivo.

Los sistemas operativos, en general, realizan dos tipos de tareas diferentes, aquellas que son importantes y delicadas que tienen que ver con la gestión de recursos compartidos, donde si un programa solicita o toma demasiados recursos, puede afectar la operación de otros programas que están en el mismo ordenador. Estas funciones deben ser realizadas por módulos del sistema operativo muy bien programados, que verifiquen y garanticen que lo que hace un programa no afecta negativamente a los otros programas. Estas funciones son ejecutadas solo por rutinas del sistema operativo que no representan ningún riesgo, ya que se ha probado que hacen un uso adecuado de los recursos y no favorecen a ningún programa en particular que no lo requiera. Estas funciones corren en lo que se llama **Modo Supervisor** o **Modo Kernel**, lo que les permite tener acceso a la mayoría de los componentes y recursos del ordenador.

Por otra parte, hay servicios y programas que pueden ejecutar los usuarios para hacer labores de menor importancia o de menor riesgo cuya ejecución, en general, no debe impactar negativamente en el funcionamiento de otros programas. Estos programas, que pueden pertenecer o no al sistema operativo, corren en lo que se llama **Modo Usuario** y, por tanto, no tienen acceso a las estructuras de datos y a los recursos críticos y delicados del ordenador.

En este ambiente, todos los programas que corren en modo supervisor o **kerne**l pueden ser agrupados en lo que se llama el núcleo o kernel del sistema operativo. Para acceder a los servicios del núcleo del sistema operativo, el mismo sistema operativo establece una serie de controles para que los usuarios puedan utilizar estos servicios, que incluyen el hecho de hacer una llamada al sistema. Esto evita que sean programas elaborados por los usuarios los que tengan acceso a las áreas críticas o delicadas y, de esta forma, no se pueda hacer un uso indebido o no adecuado de los recursos del ordenador.

El usuario puede interactuar directamente con el sistema operativo a través de dos mecanismos, mediante una **Interfaz de Comandos** o **Shell**, que posee una serie de programas incluidos y que permite la creación de nuevos programas para aumentar los servicios que el mismo sistema operativo puede ofrecer. También es posible solicitar algunos servicios del sistema operativo a través de una serie de llamadas que reciben el nombre de **llamadas al sistema**, estas llamadas conforman un API a través del cual los programas de usuarios pueden solicitar determinados servicios del sistema operativo para poder funcionar y obtener los recursos que necesita para su ejecución.

Estos dos componentes se encargan de dos de las principales funciones del sistema operativos; el **Shell**, de interactuar directamente con el usuario a través de un lenguaje de comandos; y el kernel permite que los programas de los usuarios puedan tener acceso a los servicios que ofrece el sistema operativo y el hardware del ordenador a través de las llamadas al sistema que, si bien son invocada por los programas de los usuarios, realmente están implementadas en el kernel y allí están en la capacidad de acceder a las estructuras de datos y los recursos compartidos y de manejo delicado que están dentro del mismo sistema operativo.

Entre los recursos que debe administrar todo sistema operativo se encuentran los procesos, la memoria, los dispositivos y los archivos; cada uno de estos recursos son administrados por un manejador particular. Veamos cuáles son las funciones generales que cada uno de estos manejadores debe implementar.

El manejador de procesos es el componente encargado de la ejecución de los programas de usuarios, esto implica dos grandes funciones: a cada programa se le debe garantizar contar con los recursos que este requiera y un ambiente adecuado para su correcta ejecución, esto es parte de lo que debe hacer el Planificador de Procesos. Por otra parte, hay un componente que se encarga de manejar todos los programas que se estén ejecutando en el ordenador y de seguir una serie de políticas para identificar qué programa se debe ejecutar en un momento dado, y cuál o cuáles deben ser los siguientes programas a ejecutarse una vez que el programa actual termine o no pueda continuar con su ejecución. Esto es parte de lo que debe realizar el **Despachador**.

El manejador de memoria es el componente encargado de ofrecer y administrar el espacio que los programas requieran para almacenar, tanto las instrucciones que conforman el código mismo del programa, como el espacio para las estructuras y los datos que el programa deba procesar, así como también el espacio donde el ambiente de ejecución del programa deba ser preservado cada vez que un proceso deba abandonar la CPU.

El manejador de dispositivos es el componente que se encarga de administrar y facilitar el uso de los dispositivos de almacenamiento y de los dispositivos de entrada y salida en el ordenador. Cada dispositivo puede tener algunas particularidades de uso dependiendo de sus características operativas y tecnológicas. El trabajo del manejador de dispositivos es el de unificar el funcionamiento de estos dispositivos diferentes para que los usuarios puedan acceder a la información de cada dispositivo, de forma más o menos uniforme, ocultando la mayoría de los detalles de operación de estos componentes, de forma que se le facilite al usuario la labor de consulta y almacenamiento de información en todos estos dispositivos.

El manejador de archivos es el componente del sistema operativo que se encarga del almacenamiento persistente de la información del usuario en el ordenador. Si bien es cierto que mientras un programa está ejecutándose, prácticamente toda la información que necesita está en la memoria principal del ordenador, más específicamente en la memoria RAM. Esta memoria tiene la particularidad de que, si bien pueden almacenar y entregar datos muy rápidamente, también está el hecho de que solo puede mantener la información mientras el ordenador esté encendido. Una vez que el ordenador se apaga, la información que contiene la memoria principal se pierde. Para evitar que la información de los usuarios se pierda y deba ser regenerada cada vez que se deba encender el ordenador, han sido diseñados los dispositivos de almacenamiento persistente o permanente que, mediante mecanismos electromagnéticos u ópticos, pueden guardar y mantener almacenada la información aún después de que el computador esté apagado. Como la cantidad de información que cada usuario puede almacenar en un ordenador es considerable, se requiere de algunos mecanismos de organización y estructuración de forma que los usuarios puedan encontrar la información que requieran de forma rápida y eficiente, independientemente de la cantidad de información que esté almacenada. La labor del manejador de archivos es la de estructurar la información almacenada, de forma que pueda ser fácilmente accedida y utilizada por los usuarios del sistema.

Los sistemas operativos y los diferentes componentes que los conforman se pueden organizar de diferentes formas para poder ejecutar sus funciones principales; en este sentido, históricamente han sido definidas una serie de modelos arquitectónicos para la construcción de sistemas operativos, a continuación, describiremos algunos de los modelos de arquitectura de sistema operativos más utilizados.

1.3.1. Modelo de Arquitectura Monolítica

La palabra Monolito significa una sola piedra, por ello, **la arquitectura monolítica** es aquella en la que el sistema operativo se construye de un todo en una sola pieza uniforme, en la que no se diferencian elementos distintivos, tal como se presenta en la figura 1.8. Un sistema operativo monolítico es aquel construido a partir de un único programa que hace todo. Este enfoque en el que no hay una estructura clara en el sistema, donde todo se interconecta y se integra con todo, fue el enfoque que se usó en los primeros sistemas operativos. Con sistemas pequeños, ese tipo de arquitectura puede llegar a conseguir resultados importantes, pero a medida que los sistemas se hacían más complejos e incluían más funciones, la falta de estructura hace que sea mucho más difícil mantener el código de los sistemas e integrar nuevos componentes.

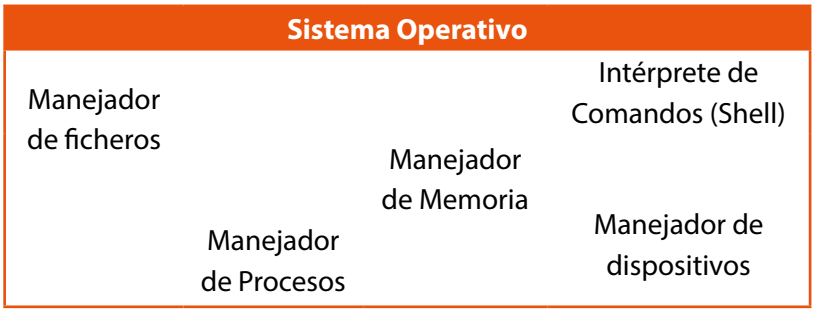


Figura 1.8. Arquitectura Monolítica en la que los componentes del sistema operativo están todos contenidos en una misma estructura común. Fuente: elaboración propia.

En la medida en la que la complejidad de los sistemas operativos fue aumentando, estos se volvieron programas más modulares, formados por diferentes componentes independientes, que se interrelacionan siguiendo algunos esquemas de integración bien definidos.

1.3.2. Modelo de Arquitectura de Kernel

Los modelos de **arquitectura de kernel** son modelos en los que los componentes más delicados del sistema operativo y, en especial aquellos que permiten compartir recursos, son agrupados en un núcleo o kernel del sistema operativo, concentrándose allí las funciones, los recursos y las operaciones más importantes del ordenador.

La ejecución de las rutinas que se encuentra en el kernel se podrá hacer solo en modo supervisor con el fin de evitar que un usuario, sin el permiso o autorización adecuada, pueda acceder o modificar las estructuras de datos asociadas a la administración de recursos del ordenador de forma incorrecta, ya que esto puede representar un riesgo para la operación de todos los sistemas del equipo.

1.3.3. Modelos de Micro Kernel

El modelo de **Micro Kernel** es una variación del modelo de kernel, tal como se muestra en la figura 1.9. En este modelo se ha realizado un estudio exhaustivo para identificar y reducir al mínimo las funciones más básicas que deben estar en el kernel, de forma que sea lo más pequeño y eficiente posible, conteniendo solo aquellos componentes que deben correr en modo kernel, dejando que el resto de los servicios que un sistema operativo puede ofrecer y que no requieren acceder a componentes delicados que puedan ocasionar efectos adversos a otros programas, se ejecuten en modo usuario.

De ser necesario, cualquier programa, componente o servicio puede invocar a los servicios del micro kernel para que sea el kernel el que realice, en nombre de los programas de los usuarios, las actividades más delicadas en el sistema. Al ser una rutina del kernel la que accede a los recursos internos y delicados del sistema operativo y no una rutina programada por el usuario, se reduce de forma importante el riesgo de que las rutinas del usuario realicen una tarea de forma incorrecta y dañen no solo el funcionamiento de su programa, sino también el funcionamiento de otros programas que se estén ejecutando en el ordenador.

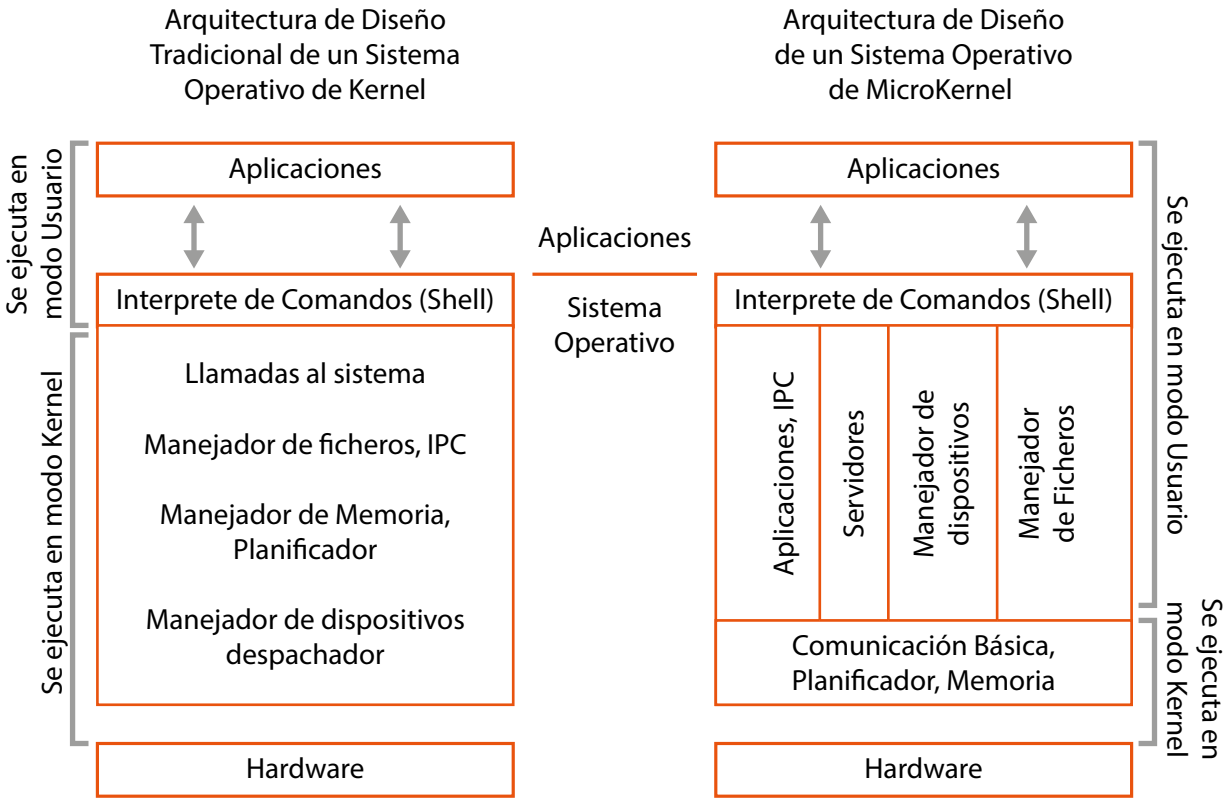


Figura 1.9. Comparación de una arquitectura de Kernel y una arquitectura de Micro Kernel. Fuente: adaptado de Wooptoo (2008).

1.3.4. Modelo Jerárquico o de Capas

Las arquitecturas jerárquicas por capas o anillos son arquitecturas basadas en el enfoque de divide y vencerás (*divide and conquer*), en el que hay una capa básica inicial donde están implementadas las funciones más básicas del sistema y, de allí en adelante, el sistema se organiza en una serie de capas concéntricas.

Cada capa usa los servicios que le ofrecen las capas inferiores y construye, a su vez, servicios que ofrece a las capas superiores del sistema, tal como se muestra en la figura 1.10.

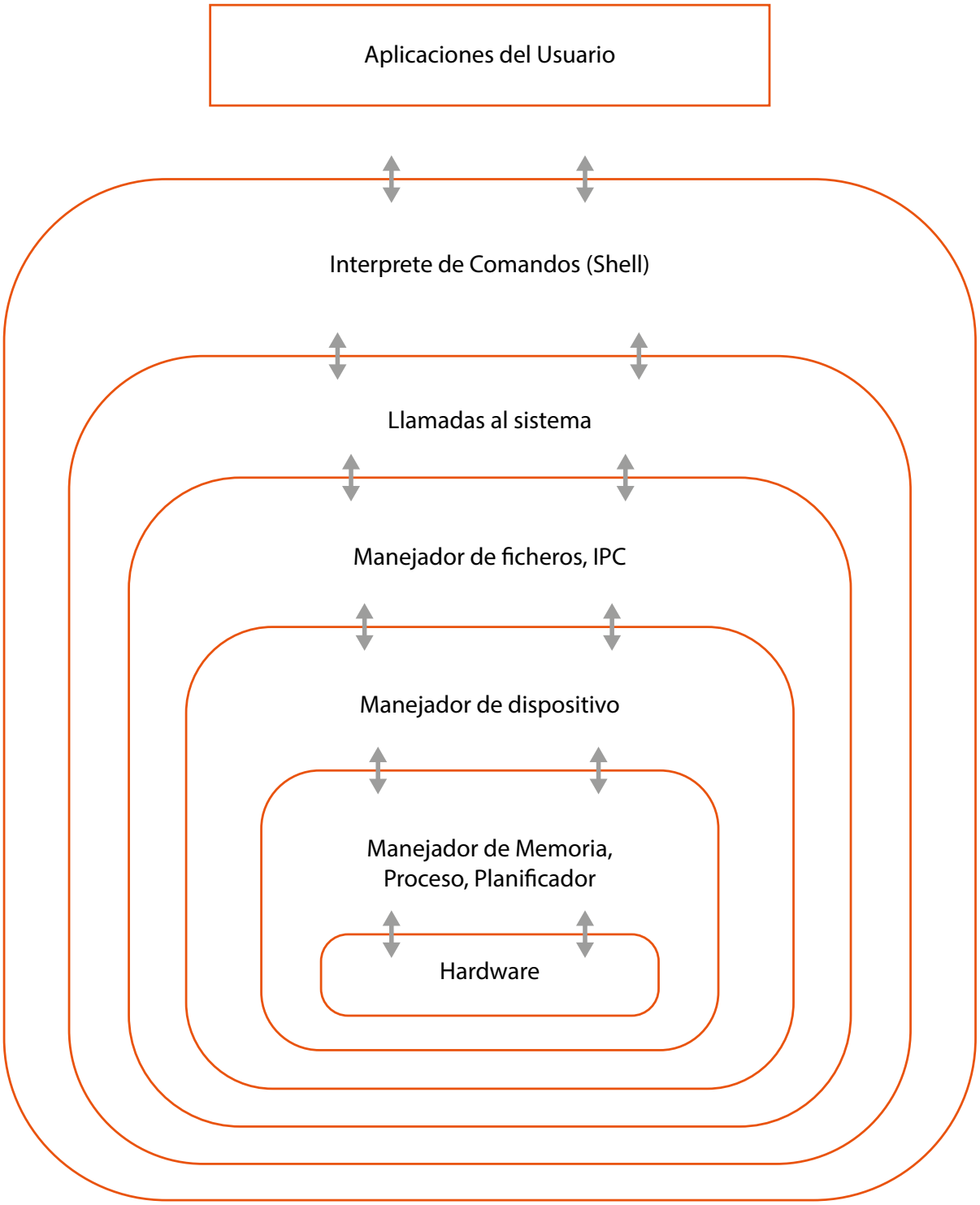


Figura 1.10. Modelo Jerárquico o por Capas de un Sistema Operativo. Fuente: adaptado de P. de Miguel y F. Pérez (2016).

1.3.5. Modelos en Red y Distribuidos

Las arquitecturas en red o las de sistemas distribuidos se basan en el hecho de que los diferentes manejadores que conforman un sistema operativo pueden ejecutarse en más de un ordenador y, por ello, los recursos que se usan pueden estar entonces repartidos en una o varias redes de ordenadores.

En este modelo de arquitectura, desde uno de los ordenadores de la red se puede tener acceso a los recursos y servicios que están disponibles en otros equipos, como se aprecia en la figura 1.11 Cuando el uso de los recursos y servicios lo hace el usuario siendo consciente de que los recursos que requiere están en otros equipos físicos diferentes al suyo, se dice que estamos trabajando en un **Sistemas en Red**. Si los usuarios están en disposición de usar los recursos del sistema de forma totalmente transparente de donde estén realmente ubicados y, además, les es posible solicitar, usar y manejar los recursos de la misma forma, ya sea que estén locales a su ordenador o funcionen en otros ordenadores de la red, se dice entonces que estamos ante un **Sistema Distribuido**. En ambos sistemas hay recursos que están en la red, pero en los sistemas distribuidos los recursos se deben poder acceder, usar y administrar de forma totalmente transparente a su ubicación física en la red.

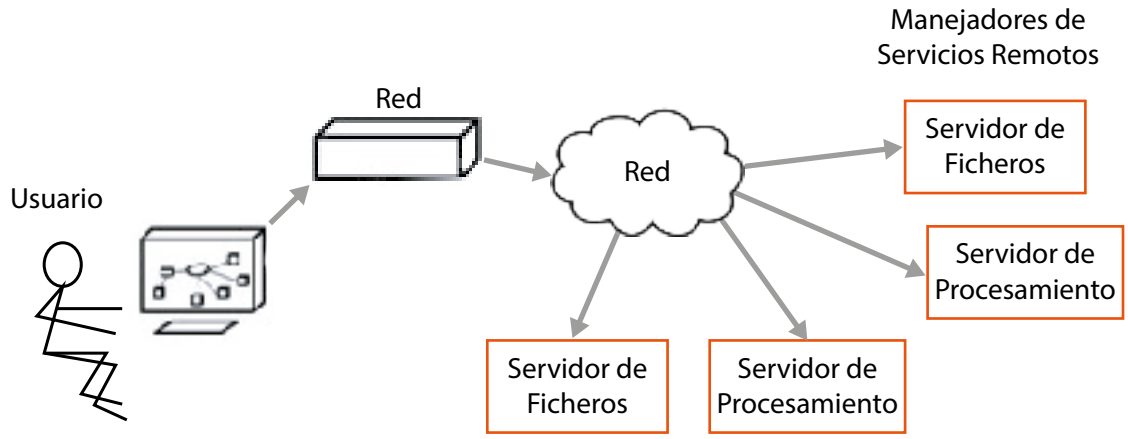


Figura 1.11. Modelos de Arquitecturas en Red y de Sistemas Distribuidos. Fuente: elaboración propia.

1.4. El sistema operativo como una máquina extendida

El ordenador es un equipo electrónico especializado con un gran potencial que puede hacer muchísimas cosas, pero al que hay que indicarle claramente y sin ambigüedades qué debe hacer y en qué momento debe hacerlo. Aunque el usuario, en general, puede tener una buena idea de qué es lo que quiere hacer, es posible que no conozca muchos detalles respecto a cómo se debe realizar una tarea de forma específica.

Veamos un ejemplo para entender más claramente esta situación. A casi todos nos gusta comer una chocolatina y esto además es bastante sencillo, ¿cierto? Pero, ¿realmente sabemos lo que ocurre cuando comemos una chocolatina? Veamos; primero, quitamos el envoltorio y le damos un mordisco; luego, empezamos a masticar y nuestra saliva se empieza a mezclar con la chocolatina; la subida de temperatura al estar en nuestra boca a 37°C, junto con la masticación, hace que el chocolate comience a derretirse, formándose una especie de pasta, lo que va haciendo que se liberen los carbohidratos, las grasas y una gran cantidad de otras sustancias como la teobromina y la feniletilamina. En este proceso, la amilasa que es una encima que está en nuestra boca, se adhiere a los carbohidratos y los descompone en hidratos de carbono más pequeños para que nuestro cuerpo pueda asimilarlos. Luego, tragamos y el chocolate va al estómago; después, al intestino donde, gracias a la acción del ácido clorhídrico y a otras enzimas, los carbohidratos siguen degradándose hasta formar glucosa, que es uno de los componentes que pueden tomar nuestras células para producir energía.

Pero, ¿a qué viene toda esta historia de reacciones químicas? Pues imagine que usted debiera recordar cada uno de estos detalles químicos y fisiológicos cada vez que quisiera comerse una chocolatina. ¿Cuantas podría comerse?, quizás ninguna. Es nuestro cuerpo, mediante los sistemas: nervioso, endocrino y digestivo, quien, automáticamente, se encarga de manejar todos los detalles sobre qué se debe hacer para obtener la energía de una chocolatina, y lo hace de una forma tan simple que hasta un niño puede hacerlo. De esta misma forma, podemos aprovechar los recursos de un ordenador sin tener la necesidad de conocer todos los detalles de su operación gracias a las funciones y servicios que nos ofrece el sistema operativo. Aunque la verdad es que mientras más información tengamos de su funcionamiento, mejor podremos sacar provecho de él.

1.4.1. Extendiendo las funciones del Hardware

Aunque un ordenador puede hacer muchas cosas, cuando agregamos el sistema operativo y algunos otros programas especializados, potenciamos lo que el ordenador puede hacer a un nivel superior, se habla entonces de tener una **máquina extendida**. Para entender mejor este concepto, pensemos en nosotros mismos como seres humanos, en este sentido tenemos una serie de habilidades para hablar, caminar, conversar con otras personas; luego, en nuestra casa aprendemos un conjunto de habilidades adicionales, aprendemos el respeto, a escuchar a los demás, aprendemos a compartir. Posteriormente, vamos a la escuela y aprendemos una serie adicional de habilidades, aprendemos matemáticas, historia, geometría, para luego ir a la universidad donde finalmente aprendemos habilidades específicas de una profesión; nos volvemos profesionales. Cuando después de graduados vamos a un trabajo, nosotros somos seres humanos extendidos, con las habilidades que nos ha dado el aprendizaje que recibimos en casa, más lo que aprendimos en la escuela, más la formación profesional que adquirimos en la universidad. Gracias al uso de todos estos aprendizajes podemos hacer, como personas, muchas más cosas y actividades de las que podemos hacer simplemente como seres humanos sin ninguna formación. De esta misma manera, un ordenador es un dispositivo que permite almacenar información en su disco duro, sin embargo, el sistema operativo brinda todo un sistema de abstracción, basado en el uso de directorios y ficheros, para que la información de los usuarios pueda almacenarse y recuperarse de las unidades de almacenamiento del ordenador de forma más eficiente y natural para el usuario, para quien es mucho más fácil recordar que la información que busca está en una carpeta de nombre Documentos, dentro de la carpeta Administración y, allí, en el documento Gastos del mes de marzo, en la cuarta página, a recordar que los datos que busca están el cilindro 32, pista 7, sector 24, 12 registros más abajo del primer bloque de información del fichero. Aunque esta segunda manera es realmente la forma en la que puede estar almacenado el dato, es mucho más fácil para el usuario recordar la organización de carpetas y ficheros. En este ejemplo las capacidades de almacenamiento del ordenador son extendidas, con un esquema organizativo, basado en carpetas, ficheros y documentos que facilita al usuario recordar la ubicación de sus recursos y tener un mejor control de las informaciones que requiera.

1.5. Interacción del Sistema Operativo con el hardware del ordenador

Entre las principales actividades que debe realizar el sistema operativo se encuentra la de permitir que los programas puedan usar de forma eficiente el hardware del ordenador y, para ello, ofrece una serie

de manejadores y servicios que los programas pueden invocar para poder usar los recursos físicos del ordenador. Veamos algunos ejemplos de estas interacciones.

1.5.1. La ejecución de programas dentro de la CPU

Una de las funciones del sistema operativo es permitir la ejecución de programas del usuario, pero que implica ejecutar un programa en un ordenador. Un programa es un conjunto estructurado y ordenado de instrucciones que, al ejecutarse, satisfacen alguna necesidad de información de un usuario; pero qué significa que un programa se esté ejecutando.

El programa, para ejecutarse, debe estar cargado en la memoria del ordenador. A partir de la memoria, la CPU va tomando instrucción a instrucción del programa y las va ejecutando. Dentro de la CPU hay un conjunto de registros que son pequeñas áreas de almacenamiento que permiten usar los componentes de hardware del procesador de forma más rápida y eficiente, tal como se observa en la figura 1.12.

Uno de estos registros, que está en la CPU, es el PC (*Program Counter*) o **Contador de Programa**, que almacena la dirección de la próxima instrucción a ejecutar. En los ordenadores también suele haber una serie de registros de propósito general, donde se van almacenando los resultados parciales de los cálculos, comparaciones y operaciones que va haciendo la CPU mientras ejecuta un programa; además, hay también un conjunto de registros que permite acceder a la memoria principal o memoria RAM del ordenador. Todo esto conforma lo que se llama el entorno de ejecución de un programa.

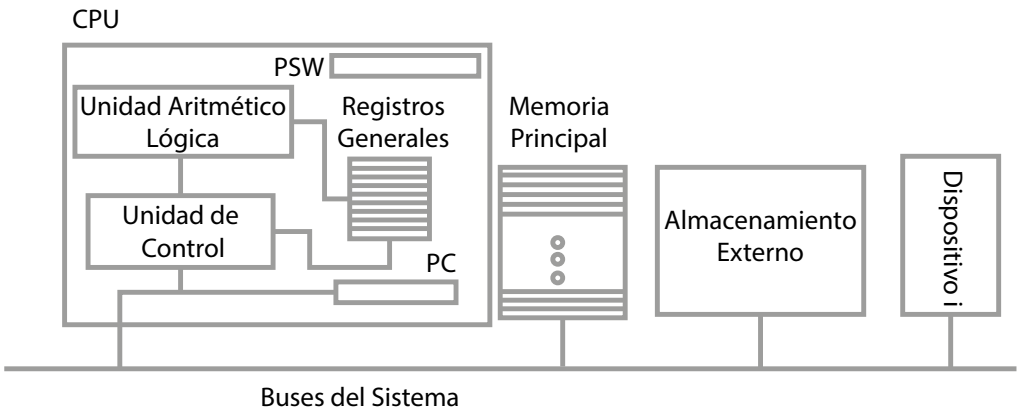


Figura 1.12. Arquitectura general de un ordenador, su CPU y algunos de los registros que contiene. Fuente: adaptado de A. Tanenbaum (2009).

Ahora, un ordenador tiene una asombrosa capacidad de trabajo o procesamiento; si hablamos de un ordenador que trabaja a 2,5 GHz, esto implica que posee un reloj que cambia a razón de 2.500.000.000 veces por segundo. El reloj es como el corazón del computador, lleva el ritmo y coordina todos los componentes de hardware del ordenador para que, al estar sincronizados y engranados todos estos componentes, el ordenador pueda efectuar operaciones conjuntas a una velocidad especifica.

Si un ordenador puede hacer una operación simple en cada cambio de reloj, entonces puede correr casi cualquier programa en solo fracciones de segundo, pero lo que ocurre es que los programas requieren datos o requieren que el usuario introduzca valores por el teclado para luego procesarlos.

La velocidad a la que un usuario puede escribir, o la velocidad a la que puede obtenerse datos de un disco, es mucho más lenta que la velocidad del procesador, es por esto que cuando un programa que se está ejecutando debe buscar alguna información fuera de la CPU, entonces debe dejar de ejecutarse, y mientras se consiguen los datos, la CPU pasa a ejecutar otro programa. Pero antes de dejar la CPU, el programa que se está ejecutando debe salvaguardar su estado de ejecución; esto implica guardar los contenidos de casi todos los registros de la CPU para que, cuando vuelva a ejecutarse, pueda continuar en la CPU justo en la instrucción o en el lugar en el que se había quedado, como si nada hubiera ocurrido y no se hubiera interrumpido su ejecución. El proceso de salvaguarda del estado de ejecución de un programa al almacenar los contenidos de los registros de la CPU cada vez que se suspende la ejecución de un programa, la selección de otro programa para ejecutarse, la carga de los últimos contenidos de los registros del nuevo programa, así como la actualización del registro PC con la dirección de la siguiente instrucción del nuevo programa a ejecutar, son parte del control y gestión que hace el sistema operativo sobre los componentes de hardware del ordenador para que los programas de los usuarios puedan ejecutarse de forma adecuada y eficiente.

1.5.2. Manejo de eventos asíncronos a través del mecanismo de interrupciones

En un programa tradicional, el programador va agregando instrucciones que se van ejecutando en una secuencia para generar un comportamiento específico. Sin embargo, hay ocasiones en las que no es posible prever de qué forma debe responder el programa en todo momento, ya que no todos los eventos que ocurren durante la operación del ordenador pueden ser previstos de forma exacta. Además de esto, puede llegar a ser difícil anticipar en un programa en qué momento van a ocurrir determinadas situaciones. Esto puede pasar especialmente cuando hay que atender situaciones externas al programa mismo que se está ejecutando. Por ejemplo, si un programa está esperando que el usuario ingrese un valor por el teclado, quizás el usuario deba pensar qué número va a colocar, pero es solo después de que el usuario haya ingresado ese número cuando el programa puede continuar con sus cálculos para poder brindar la información final que el usuario requiere.

Una posible estrategia para trabajar con esta situación consiste en que el programa esté constantemente verificando si el usuario ha presionado ya alguna tecla; de ser así, entonces se le entrega el número introducido al programa, pero con muy alta probabilidad el programa puede tener que estar preguntando varias decenas, miles o millones de veces antes de que el usuario tenga tiempo de introducir el número en el teclado. A esta estrategia se la denomina **Espera Ocupada** o ***Polling***. Pero en esta estrategia es más el tiempo que se pierde preguntando si un determinado evento ha ocurrido, que el tiempo que se pasa realmente procesando o generando información útil.

Una estrategia alternativa, puede consistir en contar con un mecanismo, similar a una alarma, que nos avise en el momento en el que el usuario escriba un número o presione la tecla *Enter* y, mientras esto ocurre, la CPU puede ir a ejecutar otro programa. En el momento que el usuario introduzca el dato que el programa está esperando, entonces el mecanismo avisará a la CPU mediante algún tipo de señal o alarma de que el evento que ha estado esperando ya ocurrió, y que debe ir a atenderlo. Para manejar estos casos, el ordenador cuenta con un mecanismo para el manejo de eventos asíncronos, este es el **Sistema Manejador de Interrupciones y Excepciones**. Esto le permite al ordenador lidiar o manejar situaciones asíncronas que no se sabe realmente en qué momento van a ocurrir, por lo que

este se ha convertido en uno de los mecanismos más importantes del sistema operativo para hacer una adecuada gestión de los recursos del ordenador.

1.5.3.Máquinas Virtuales

El concepto de máquinas virtuales o *Virtual Machines* se ha puesto de moda en los últimos años como una forma de aprovechar mejor los recursos de un ordenador y, a la vez, poder ejecutar sobre él varios sistemas de forma segura, flexible y escalable.

Para comprender adecuadamente este término; primero, tenemos que indicar que una máquina virtual no es una máquina real, pero al final quien va a ejecutar los programas es, en último término, una máquina real que brinda servicio a más máquinas virtuales, y es en estas máquinas virtuales donde realmente se ejecutan los programas de los usuarios.

Actualmente, se usa el concepto de máquinas virtuales con diferentes objetivos; en algunos casos, se instalan diferentes sistemas, cada uno en su máquina propia, para evitar que la acción de un sistema pueda interferir o incluso llegar a bloquear la operación de otro. Una forma sencilla de independizarlos totalmente es colocar las aplicaciones en ordenadores diferentes, cada una con sus propios espacios y recursos, pero el hecho de tener varias máquinas físicas puede ser costoso, no solo por el costo de compra de los equipos, sino también porque la administración de varios equipos es más compleja que la administración de uno solo. Además de esto, en muchos casos las aplicaciones no llegan a aprovechar adecuadamente la totalidad de los equipos.

Por estas razones, una mejor opción parece ser tener una sola máquina poderosa y dividirla en varias máquinas virtuales; de esta forma, se puede tener cada sistema en un ambiente separado con su propio sistema operativo y sus recursos, de forma de aislar las aplicaciones diferentes, pero con la flexibilidad de poder reconfigurar estas máquinas, con el fin de poder repartir de otra forma los recursos del equipo para brindar más recursos a las aplicaciones que así lo requieran y menos recursos a las aplicaciones que sean menos demandantes, esto permitirá una mejor utilización de los recursos de hardware del ordenador.

Aunque el uso de máquinas virtuales ha tenido un auge importante en años recientes, lo cierto es que no es un concepto nuevo. Los primeros usos de este tipo de tecnología se remontan a los años 70, al tiempo de los Mainframes cuando la empresa IBM, para sus sistemas 360, diseñó el sistema VM con el objetivo de permitir compartir computadores Mainframes que eran bastantes costosos. Inicialmente, los sistemas OS/360 eran básicamente sistemas de procesamiento por lotes o *batch* pero, cada vez más, los usuarios estaban interesados en procesar sus aplicaciones en línea y requerían de un sistema de tiempo compartido; las primeras versiones de estos sistemas no eran muy eficientes y consumían muchos recursos.

Como una posible solución, un grupo de investigación de IBM en Cambridge, Massachusetts, desarrolló una idea basada en la observación de que un sistema de tiempo compartido proporciona dos tipos de servicio, la multiprogramación para poder ejecutar varios programas, y los servicios de una máquina extendida para que la gestión del equipo sea más fácil para los desarrolladores y administradores de sistemas. La mayoría de los sistemas crean una base de maquina extendida y, sobre ella, implementan la multiprogramación; la idea novedosa de este grupo fue poner como base un

sistema de multiprogramación para compartir los recursos de hardware del ordenador real, definiendo un conjunto de subsistemas entre los que se reparten los recursos del hardware real y, luego, sobre cada subsistema que se ve como una máquina diferente, se implementa una máquina extendida; a cada uno de los subsistemas se le considera una máquina virtual independiente. Esta idea permitía que en un subsistema se pudiera instalar un sistema operativo tradicional de procesamiento por lotes, mientras que en otro subsistema de la misma máquina se podría instalar un sistema operativo que podía ser diferente, con lo que sería posible atender a usuarios interactivos con los recursos de un mismo ordenador, los cuales podrían ser compartidos por usuarios que trabajaran sobre sistemas con características diferentes, a cada uno de los cuales se le considera una máquina virtual.

Un sistema operativo tradicional se ejecuta sobre una máquina real, es decir, una máquina con un hardware específico que el sistema operativo aprovecha y administra a través de una serie de driver y administradores. En este sentido, una máquina virtual va a ser una máquina que no es real. Pero, como podemos correr programas sobre una máquina que no es real, lo que debemos hacer es agregar un conjunto de programas especiales para crear una máquina extendida que permita hacer que la máquina real se comporte como una máquina virtual, que incluso puede ser bastante diferente de la máquina real. Eso lo podemos hacer simulando o emulando el comportamiento de esa máquina virtual. Un esquema general en el que se describe esta solución arquitectónica se puede apreciar en la figura 1.13. En ella, sobre el hardware real se implementa un sistema que permite la creación de una o varias máquinas virtuales; sobre este sistema se definen las diferentes máquinas virtuales, en cada una de las cuales se crea o emula una instancia de un hardware virtual y sobre este se ejecuta un sistema operativo para dar soporte a las aplicaciones de los usuarios. Como cada aplicación está en su propia máquina virtual, los problemas que puedan ocurrir en una aplicación dentro de una máquina virtual A, no podrán afectar la ejecución de las aplicaciones que están en las otras máquinas virtuales del ordenador, ya que están en espacios separados e independientes.

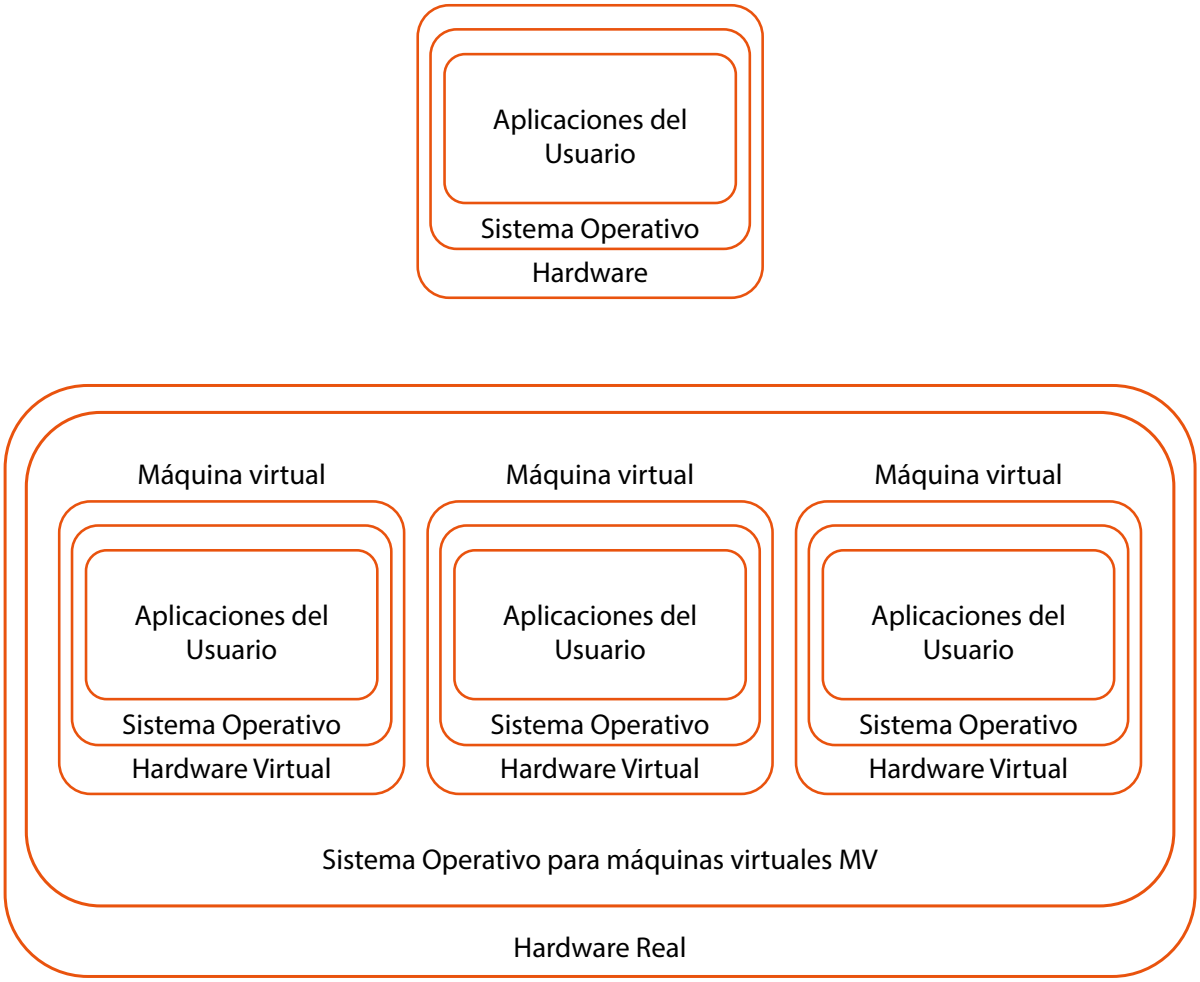


Figura 1.13.Esquema de la arquitectura de un sistema operativo tradicional en la figura de arriba y, debajo, la arquitectura general de un sistema de máquinas virtuales. Fuente: adaptado de R. Vanover (2017).

Actualmente, el esquema de máquinas virtuales ha evolucionado de forma importante, los usuarios pueden definir en sus equipos máquinas virtuales que les permiten trabajar con sistemas e incluso con sistemas operativos que no se ajustan a la arquitectura real del ordenador donde se está trabajando; para ello, se definen capas de software para envolver un equipo y producir o emular el comportamiento de otro.

Mediante esta estrategia podemos ejecutar en un ordenador que use el sistema operativo Windows una máquina virtual y, sobre esta máquina virtual, ejecutar un sistema operativo como MacOs o Linux que, originalmente, pudiera no ser compatible con Windows; de esta forma se puede trabajar con aplicaciones que han sido diseñadas para otras plataformas diferentes a las del equipo con el que contamos.

Este concepto ha sido extendido incluso a ciertos lenguajes como ha sido el caso de Java, donde los programas desarrollados se ejecutan sobre una máquina virtual java (*JVM*) y, al tener implementaciones de JVM para diferentes sistemas y arquitecturas, entonces es posible ejecutar el mismo programa Java en una gran variedad de plataformas, haciendo realidad el slogan que llegó a usar *Sun Microsystem*, la empresa que creo y difundió el Java, indicando que sus programas en Java

cumplían con la característica "*Write once, run anywhere*", que significa que el programa se escribirá una sola vez y podrá ser ejecutado luego en cualquier otro dispositivo que implemente una máquina virtual JVM.

1.6. Interacción del usuario con el sistema operativo

Existen dos grandes formas en las que los usuarios pueden interactuar con el sistema operativo del ordenador, la forma más amigable para el usuario es emplear una interfaz gráfica en la que el usuario interactúa con el sistema operativo haciendo clic sobre iconos y elementos gráficos en la pantalla del ordenador, en la figura 1.14 podemos apreciar interfaces graficas de diferentes sistemas operativos y algunos de los elementos gráficos con los que los usuarios pueden interactuar. Mediante este tipo de interfaz, es bastante sencillo hacer ciertos procesos, pero para hacer procesos complejos de varios pasos, hace falta hacer varias acciones sobre la interfaz.

Figura 1.14: Interfaces gráficas de diferentes sistemas operativos donde el usuario interactúa con el sistema haciendo clic en diferentes iconos, menús y elementos gráficos. Fuente: elaboración propia.

Otra opción es usar la interfaz de comandos desde una ventana de texto como se muestra en la figura 1.15, este esquema permite que el usuario escriba comandos de texto que pueden ser interpretados por el Shell para invocar los servicios del sistema operativo.

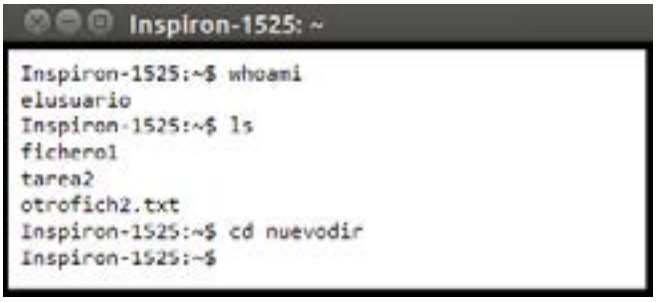


Figura 1.15: Interfaces de texto mediante las cuales el usuario puede ingresar comandos y ver las respuestas de estos comandos para poder consultar información del sistema. Fuente: elaboración propia.

Si bien el uso del *Shell* permite que los usuarios y administradores ingresen comandos para solicitar información o ejecutar acciones sobre el sistema operativo que le ayudan a manejar y gestionar el ordenador, aún más importante es la posibilidad de combinar estos comandos para desarrollar programas que le permitan automatizar ciertas actividades que, de otra forma, tendría que ir haciéndolas una a una, requiriendo gran atención de su parte. Esto se logra mediante los lenguajes de *Script* o *Shell Scripts*.

Los *Shell Script* son programas informáticos diseñados para combinar diferentes comandos del *Shell* de Sistema operativo de manera que puedan ser ejecutados de forma automática por el intérprete de comandos o *Shell*, teniendo la posibilidad de hacer que la salida de un comando pueda ser redireccionada para poder ser automáticamente la entrada de un segundo comando; también es posible hacer que la ejecución de un comando genere un fichero de salida que luego pueda ser usado por otros programas posteriores para obtener de ellos información que les ayude a realizar sus actividades, esto permite crear nuevas funcionalidades combinando y componiendo programas y comandos ya existentes en el sistema.

Algunas de las operaciones que pueden ser realizadas mediante programas de *Scripts del Shell* contemplan actividades de ejecución de programas, manipulación de archivos, la impresión de texto, entre muchas otras. Con este mecanismo también es posible combinar diferentes comandos y llamadas al sistema de forma de poder automatizar tareas complejas de consultas de datos y administración de elementos del sistema. Este tipo de herramienta está presente en prácticamente todos los sistemas operativos, por ejemplo en la forma del *Command Shell* o *cmd* y los ficheros .bat en los sistemas Windows, archivos de comandos de *bash*, *born*, *korn*, *cshell* o sh en los sistemas Linux/ Unix y *Applescript* en los sistemas Mac OS y OS X.

Otra opción es usar lenguajes de programación tradicionales que hagan llamadas al sistema operativo, pero esta opción puede llegar a ser un poco más limitada en cuanto a tomar la salida de un comando para introducirla como la entrada de un comando posterior. Aunque esto se puede realizar, es posible que requiera de una participación mayor del programador para lograr este efecto mediante instrucciones y construcciones del lenguaje. Todo con el fin de desarrollar nuevos programas y funcionalidades para el sistema.

1.7. Evolución histórica de algunos conceptos asociados a los Sistemas Operativos

Cuando aparecieron los primeros computadores, los sistemas operativos no existían y, en aquel entonces, los usuarios debían cargar sus programas y ejecutarlos directamente en el ordenador, muchas veces ajustando interruptores, moviendo perillas y otros elementos de hardware para especificar lo que querían que sus programas hicieran. En ese momento, un ordenador contenía solo un programa a la vez. A medida que los computadores se fueron haciendo más rápidos y con mayor capacidad, surge el concepto del procesamiento por lotes o batch, en este tipo de procesamiento los operadores cargaban varios trabajos o Jobs en el ordenador para que se fueran ejecutando uno a la vez. En este punto, los programas también habían avanzado y se hacían cada vez más grandes, por lo que se requería de algún apoyo de software para hacer actividades cada vez más complejas; allí surgen los primeros monitores que permiten hacer algunos manejos automáticos en el ordenador y que no son gobernados directamente por los programas de los usuarios; de esta manera surgen los primeros programas de sistema que terminan dando forma a los primeros sistemas operativos.

El siguiente avance ocurre cuando los sistemas operativos controlan ordenadores más poderosos, en cuyas memorias pueden residir varios programas a la vez; es entonces cuando surge el concepto de multiprogramación. Con CPUs que son cada vez más rápidas, el ordenador puede tomar un programa y empezarlo a ejecutar hasta que termine o hasta que se detenga, a la espera de alguna entrada o

salida de datos; en ese momento, en lugar de quedar ocioso y sin hacer nada, el ordenador puede, mientras el programa espera por un dato, suspender su ejecución para iniciar la ejecución de algún otro programa que sí pueda continuar. Así, en un mismo instante de tiempo, en el ordenador pueden estar varios programas, cada uno en un estado diferente de ejecución. Cuando alguno deba pararse, otro programa toma su lugar y continúa, pero para poder realizar esto, el ordenador debe salvaguardar todo el contexto de ejecución del programa que se está ejecutando, antes de cargar el siguiente.

Para entender mejor por qué debe ser así este proceso, imaginemos que estamos haciendo una tarea de sistemas operativos en una mesa en nuestra casa. En esa mesa tenemos un cuaderno con apuntes, un libro abierto en el capítulo 2, unas hojas donde vamos resolviendo los problemas y ejercicios que vamos consiguiendo y, finalmente, otro grupo de hojas donde vamos escribiendo la versión final de la tarea que vamos a entregar. Después de que llevamos un rato trabajando, recibimos una llamada de un compañero de clase que nos recuerda que tenemos una tarea de matemáticas que debemos entregar para mañana mismo. Al escuchar esto, tomamos la decisión de que debemos hacer primero la tarea de matemáticas, ya que es más urgente, pero no podemos simplemente tirar todo lo que está en la mesa para empezar a hacer la tarea de matemáticas, porque perderíamos todo lo que llevamos hecho de la tarea de sistemas operativos. Lo que hacemos entonces es lo siguiente: colocamos una marca en el libro de sistemas operativos, en la parte donde nos hemos quedado leyendo para poder continuar luego desde allí, también colocamos un lápiz en la **página** del cuaderno de apuntes y allí dentro incluimos, de forma ordenada, las hojas con las preguntas que llevamos de la tarea, así como las hojas donde hemos estado apuntando resultados y garabateando posibles soluciones. Luego, colocamos todos estos elementos en un estante o una gaveta y, una vez que tenemos la superficie de la mesa limpia, empezados a sacar los libros, cuadernos y hojas para la tarea de matemáticas, comenzamos a trabajar en esta nueva tarea y, una vez que la terminamos, guardamos todo el material de matemáticas. A continuación, abrimos la gaveta, sacamos nuevamente los libros, cuadernos y hojas de sistemas operativos y los colocamos de la misma forma en la que estaban antes de que paráramos. Para esto, es de mucha ayuda la marca en el libro y el lápiz que dejamos justo allí donde nos habíamos quedado trabajando la última vez. A partir de esta situación, continuamos con la tarea de sistemas operativos como si la anterior tarea de matemáticas nunca hubiera ocurrido.

Fíjense, entonces, que en la misma mesa estuvimos haciendo dos actividades diferentes; primero, la de sistema operativos; luego, la de matemáticas y, finalmente, volvimos a la de sistemas operativos, pero en instantes diferentes de tiempo. Cuando cambiamos de una tarea a la otra, organizamos todo para que, al terminar la segunda tarea, pudiéramos volver a retomar lo que estábamos haciendo en la primera, esto es equivalente a salvaguardar el contexto de ejecución de un programa que se está ejecutando para, luego, regresar a él más tarde y continuar su ejecución como si la interrupción nunca hubiera ocurrido.

Tan pronto como los ordenadores se hicieron lo suficientemente rápidos y potentes, fue posible que varios usuarios pudieran trabajar de forma más o menos simultánea en un mismo ordenador, incluso los equipos llegaron a tener tal capacidad que podían tener varias decenas de usuarios trabajando casi al mismo tiempo. En ese momento los ordenadores ya podían procesar tanta información que para los usuarios era como si cada uno estuviera trabando solo en el computador.

En los años 60, ya se contaba con grandes ordenadores para el procesamiento comercial de información en grandes empresas y bancos, aunque la miniaturización de los componentes electrónicos da como resultado la construcción de ordenadores cada vez más pequeños, rápidos y potentes. Aún los equipos eran costosos y en las empresas los usuarios debían compartir un ordenador central para procesar los requerimientos de información de los diferentes departamentos. Con la introducción de dispositivos como los teclados y las pantallas, surgieron los primeros sistemas interactivos en los que los usuarios utilizaban el teclado para directamente ingresar información, y el monitor para ver en su pantalla los resultados de la ejecución de los programas. En este mismo periodo, surgen los primeros sistemas de tiempo compartido, en los cuales un ordenador muy rápido dedica un espacio de tiempo limitado a atender a cada uno de sus usuarios; una vez vencido este tiempo, pasa a otorgarle una cantidad de tiempo equivalente al siguiente usuario, y así sucesivamente hasta que vuelve de nuevo con el primero.

A mediados de los 80, aparecen los primeros ordenadores personales con gran éxito comercial, esto generó toda una revolución ya que permitió que cualquier persona pudiera tener y operar un computador personal a un costo relativamente modesto. Estos sistemas primero fueron un tanto primitivos con interfaces de comandos de texto, como los grandes computadores, pero en esta misma década empiezan a aparecer las primeras interfaces gráficas comerciales y los primeros equipos de uso personal con capacidad multimedia; destaca aquí la aparición de los equipos Apple y, en especial, su modelo Macintosh, así como la IBM PC con sus sistemas de operación DOS y Windows. A partir de este punto, se popularizó el uso de los ordenadores, las redes e Internet. El surgimiento del **Movimiento del Software Libre** y la creación de sistemas de libre distribución como Linux también marcaron un hito importante en el que la colaboración de usuarios alrededor de mundo permitió contar con sistemas cada vez más funcionales y robustos. Estos sistemas, junto con el uso de dispositivos móviles, han estado cambiando el mundo de la tecnología y permitiendo que, cada vez, podamos estar más conectados enviando, manejando y recibiendo información en todo momento y lugar. Nada de esto sería posible, tal como lo hacemos hoy en día, sin el apoyo de los sistemas operativos que nos facilitan la utilización, la coordinación y la gestión de los ordenadores, dispositivos móviles, portátiles y servidores que forman parte del ecosistema tecnológico en el que vivimos hoy en día.

Tema 2.
Gestión de Procesos

Usamos los ordenadores para ejecutar programas, pero en el momento en que estos programas están ejecutándose en el ordenador pasan a ser algo más que solo programa. A este nuevo elemento que vamos a tener dentro del ordenador le vamos a dar el nombre de **Proceso**. Pero realmente, ¿qué es un proceso? Vamos a dar una primera definición de un proceso como la imagen de un programa que está en ejecución, esta es una definición que nos sirve como un punto de inicio. Pero veamos todo lo que requiere un programa para poder ejecutarse en un ordenador.

Lo primero que debemos tener en cuenta son las instrucciones del programa, a esto es a lo que se le llama el programa ejecutable, que debe estar escrito en código de máquina, es decir, escrito de forma que el ordenador pueda entenderlo y ejecutarlo directamente; este componente también recibe el nombre de **segmento de texto**. Pero las instrucciones no son lo único que le hace falta a un programa para poder ejecutarse, un programa también requiere de espacios de memoria donde se almacenarán los datos que va a usar, de hecho, realmente se usan, al menos, dos áreas de memoria; la primera es donde se almacenarán las variables y estructuras de datos que serán usadas durante la ejecución del programa, a esta área se le suele llamar **Pila** o ***Stack***; y una segunda que recibe el nombre de ***Heap***.

Veamos primero cómo funciona la Pila. Cuando un programa invoca a una rutina o función para ejecutar una sección de código independiente, también se le crea una nueva área dentro de la pila que contendrá las variables locales a esta función o procedimiento, por lo que la pila será un área que podrá crecer dinámicamente a medida que un programa se va ejecutando y va invocando funciones y procedimientos.

La otra área de memoria que podrá usar un programa en ejecución corresponderá al espacio donde puede solicitar la creación de estructuras dinámicas de datos, esto dependerán de la ejecución del programa. Esta área recibe el nombre de Heap y es donde las llamadas al sistema como malloc del lenguaje C o la creación dinámica de objetos con el constructor new en Java, toman el espacio que ocupan las estructuras dinámicas como pilas, colas, listas y objetos para poder existir en el ordenador. Pero esto aún no es suficiente para que el programa esté efectivamente ejecutándose. La CPU, para ejecutar un programa, necesita saber cuál es la instrucción que está ejecutando y cuál es la próxima instrucción que debe ejecutarse; estas dos informaciones son almacenadas en registros internos de la CPU en el registro IR (*Instruction Register*) o **Registro de Instrucción**, se almacena la instrucción que se está ejecutando y, en el registro PC (*Program Counter*) o Contador de Programa, es donde se almacena la dirección de la próxima instrucción a ejecutar.

Adicionalmente a esto, la CPU cuenta con una serie de registros de propósito general, donde van quedando algunos de los resultados parciales de la ejecución de las instrucciones del programa, al menos hasta que estos resultados son almacenados en variables en la memoria principal del ordenador. Además de esto, hay otra serie de datos de ejecución como, por ejemplo, en qué estado está el proceso en ese momento, si está ejecutándose, si está listo, bloqueado, suspendido o en algún otro estado. Parte de estos datos son almacenados en la PCB (*Process Control Block*) o **Bloque de Control de Proceso**, que es una estructura de datos del Sistema Operativo que registra los detalles de la ejecución de los programas. Todas estas informaciones que están almacenadas en registros de la CPU, junto con los espacios de la memoria RAM y las instrucciones del programa mismo, forman parte de lo que representa la ejecución de un programa y, por tanto, son una parte fundamental de lo que es un proceso en el ordenador, tal como se puede ver en la figura 2.1.

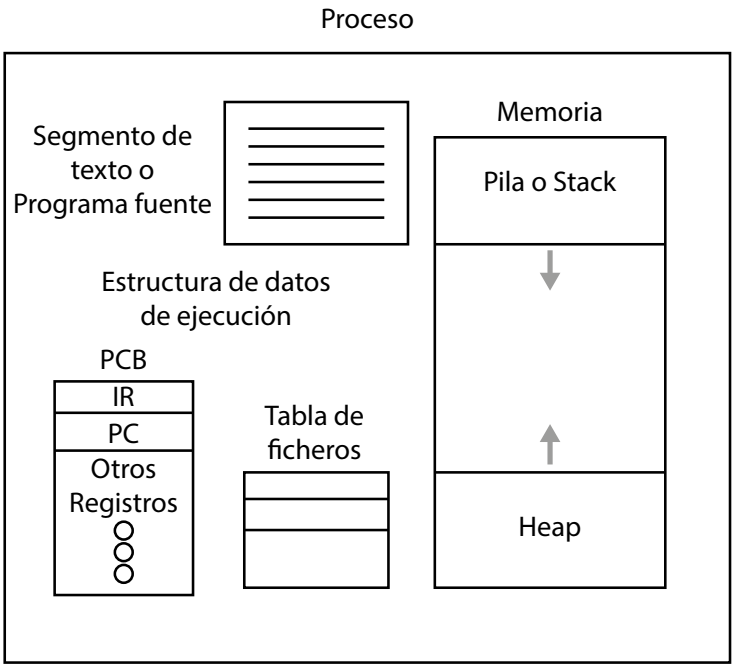


Figura 2. 1. Elementos que constituyen un proceso en el ordenador. Fuente: elaboración propia.

2.1. Diagrama de estados de un proceso

Cuando un programa llega al computador para ser ejecutado, se debe crear un proceso; para ello, se le asignan recursos para su ejecución: se compila para generar un programa ejecutable, se le asigna memoria para almacenar los datos y variables que usará para hacer su trabajo, se le asigna un segmento de pila para registrar las llamadas a subrutinas y funciones que realice y se le crea su PCB. En estos momentos y mientras el sistema operativo termina de asignar todos los recursos para que el programa se empiece a ejecutar, se dice que el proceso está en estado de *Iniciado*.

Una vez que el sistema operativo le ha asignado todos los elementos que requiere para ejecutarse, el proceso entonces es pasado a la cola de *Listos,* donde están todos los procesos que cuentan con todos los recursos para ejecutarse, con la excepción de la asignación a una CPU. Eventualmente, el Sistema Operativo, y más específicamente el Planificador, seleccionará uno de los procesos de la cola de *Listos* para pasarlo al estado de *En Ejecución* al asignarlo a una de las CPUs que puedan estar disponibles en el ordenador. Una vez que un proceso está en el estado de *Ejecutando* y termina todas sus instrucciones, pasa al estado de *Terminado* y los recursos que tenía asignados se liberan para que puedan ser usados por otros procesos en el sistema.

Cuando un proceso se está ejecutando y una de sus instrucciones requiere que se lea un dato de una unidad de disco o del teclado, en ese momento se puede generar la solicitud de información, pero conseguir esta información usualmente requerirá algo de tiempo para poder hacerse efectiva; entonces, en ese momento el proceso se pasa del estado de *Ejecutando* a un nuevo estado que llamaremos *Bloqueado*, porque el proceso no podrá continuar su ejecución, hasta que no se consigan los datos que han sido solicitados. Cuando esto ocurra, la unidad de disco o el teclado, según corresponda, lanzará una interrupción para notificar que poseen ya los datos solicitados; en ese momento, el planificador, al atender la interrupción, podrá modificar el estado del proceso para

pasarlo a la cola de Listos, de forma tal que pueda competir por la CPU con el resto de los procesos para poder continuar su ejecución.

Esta descripción ha especificado los estados más simples por los que puede pasar un proceso durante su ciclo de ejecución, tal como se puede apreciar en la figura 2.2.

Figura 2. 2: Diagramas Simple de los estados de un Proceso. Fuente:elaboración propia.

Hay ocasiones en las que los recursos del ordenador son escasos o en los que hay muchos procesos, algunos de los cuales pueden estar bloqueados, por lo que, al menos por algunos momentos, no pueden continuar su ejecución, pero como estos procesos están ocupando un espacio y mantienen unos recursos reservados, otros procesos que pudieran utilizar dichos recursos se ven imposibilitados de hacerlo y, por ello, de iniciar o avanzar en su ejecución. En estos casos, el planificador puede decidir tomar algunos de estos procesos bloqueados para colocarlos en un nuevo estado de **Suspendido**.

Los procesos suspendidos son aquellos a los cuales se les ha elegido como víctimas para sacarlos del ordenador temporalmente, con el objetivo de liberar los recursos que están usando, de forma que otros procesos, que sí puedan ejecutarse, puedan tomarlos para avanzar en su ejecución. Posteriormente, cuando existan más recursos disponibles, los procesos suspendidos pueden volver a ser colocados en el ordenador en el estado en el cual estaban antes de ser suspendidos. Usualmente, estos procesos suelen ser tomados de la cola de procesos bloqueados, ya que estos aún no pueden continuar, o de la cola de procesos listos en el caso de que exista una necesidad apremiante de liberar recursos para la atención de nuevos procesos o para culminar algún proceso ya existente. Al agregar el estado de suspendido, el diagrama de estados de un proceso se amplía para pasar a ser como se muestra en la figura 2.3.

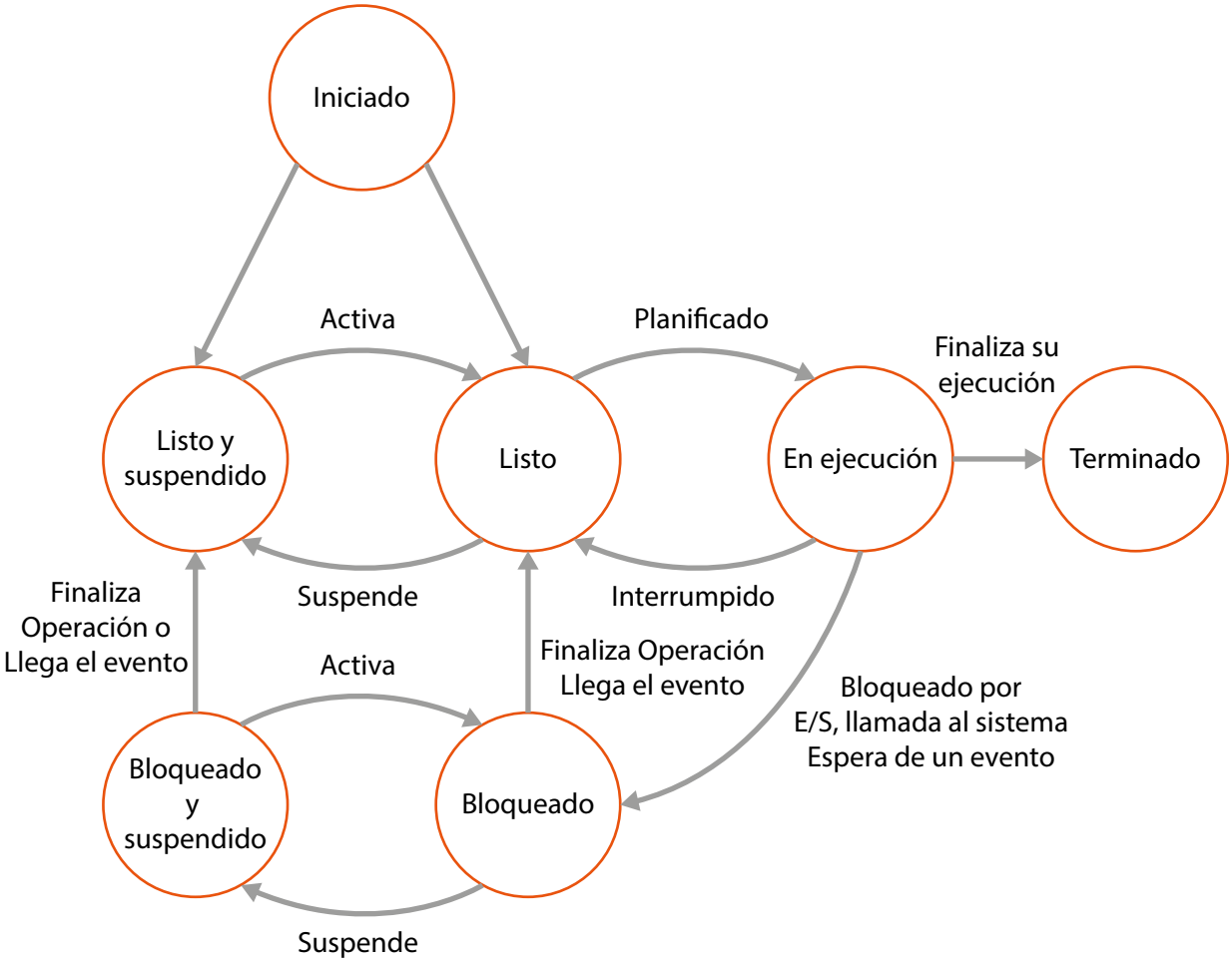


Figura 2.3. Diagrama de estados de un proceso considerando la suspensión de procesos cuando hay problemas de espacio en el ordenador. Fuente: elaboración propia.

En un ordenador tradicional con una sola CPU y un solo núcleo, solo debe haber un proceso en el estado de *En Ejecución*; sin embargo, es posible que existan varios procesos en el estado de *Listos* o de *Bloqueados*, en estos casos los procesos son organizados en *Colas* asociadas a cada uno de estos estados. Estas colas podrán ser recorridas por los programas del sistema operativo y, en particular, por el planificador de procesos y por el despachador para seleccionar qué proceso deben cambiar de estado, según las instrucciones del programa y la disponibilidad de los recursos del ordenador.

Para entender mejor estos estados y sus funciones, veamos cómo es el cambio de estados al que es sometido un proceso P a lo largo de su ciclo de vida dentro del ordenador. Cuando el proceso P ingresa en el ordenador, comienza en el estado *Iniciado*, una vez que se inicializan todas sus estructuras de datos y se le brindan todos los recursos que este necesita para iniciar su ejecución, pasa a la cola de procesos *Listos*, allí permanece hasta que es seleccionado por el planificador del procesador para iniciar su ejecución y, cuando esto ocurre, el proceso P es pasado al estado de en ejecución y allí permanece ejecutando instrucciones hasta que ocurra uno de los siguientes eventos: si el sistema operativo interrumpe la ejecución de P porque otro proceso es seleccionado para ejecutarse, entonces el proceso P es pasado a la cola de *Listos*; si el proceso P debe realizar una operación de entrada/salida para almacenar o leer un dato, que es parte de su funcionamiento, entonces el proceso es detenido y

colocado en la lista de *Bloqueados*, ya que el proceso P no podrá continuar su ejecución hasta que el dato que va a leer o escribir pueda ser obtenido o almacenado adecuadamente en algún dispositivo. Cuando esto pase, el dispositivo correspondiente avisará al sistema mediante una interrupción y, en ese momento, el proceso P será sacado de la cola de *Bloqueados* para pasar a la cola de procesos *Listos*, ya que el recurso por el que estaba esperando ya está disponible y, entonces, en cualquier momento puede reanudar su ejecución. Finalmente, si el proceso P está en el estado de *En Ejecución* y termina con todas sus instrucciones de procesamiento, este debe ejecutar una última instrucción que es una llamada al sistema para indicar que el proceso ha terminado, en este caso, la ejecución del proceso P es suspendida, el proceso es pasado al estado de *Terminado* y, finalmente, los recursos que poseía durante su ejecución son liberados de forma que puedan ser utilizados ahora por otros procesos en el ordenador.

Si el proceso P está en el estado de *Bloqueado* y el ordenador no tiene suficientes recursos para ejecutar adecuadamente todos los procesos que posee en ese momento, entonces el sistema operativo puede decidir suspender al proceso y lo coloca en la cola de *Suspendidos*; en este caso, se salvaguarda todo el estado de ejecución del proceso, pero se le quitan todos los recursos que se le habían asignado para que estos recursos estén disponibles para el resto de los procesos. Cuando otros procesos terminen y continúen liberando recursos, será posible, entonces, que al proceso P se le vuelvan a asignar los recursos que se le desasignaron y, en este caso, el proceso volverá a la cola de *Bloqueados*.

En caso de que no queden procesos bloqueados y el ordenador no posea aún suficientes recursos, se podrán suspender algunos procesos que estén en la cola de *Listos*, siempre que esto ayude a que el ordenador pueda disponer de los recursos para ejecutar de forma más eficiente los procesos que quedan en el sistema. En estos casos, se puede definir un estado diferente que lleva el nombre de *Suspendidos Listos*, para almacenar los proceso suspendidos que proviene de la cola de Listos, ya que estos, al volvérseles a asignar los recursos que necesitan, deben volver a la cola de *Listos* y no a la cola de *Bloqueados*, como habíamos descrito antes.

Usualmente, en el computador hay varios procesos en diferentes estados de ejecución y entonces surge la pregunta: cuando un proceso termina, ¿quién o qué decide cuál será el próximo proceso que se ejecutará? Esta y otras tareas son las que hace el Planificador de Procesos.

En algunos de estos estados solo puede estar un proceso a la vez, este puede ser el caso del estado de *En Ejecución*, ya que este proceso está en la CPU del ordenador; sin embargo, hay otros estados en los que puede haber varios procesos, como es los estados de *Listo*, *Bloqueados* y *Suspendidos*; en esto casos, los procesos se organizan en listas de procesos que son usualmente estructuradas en listas que enlazan las PCB de los procesos, como se observa en la figura 2.4 que, como habíamos comentado, son las estructuras de datos donde se almacenan varios de los datos que corresponden al estado de ejecución de cada proceso.

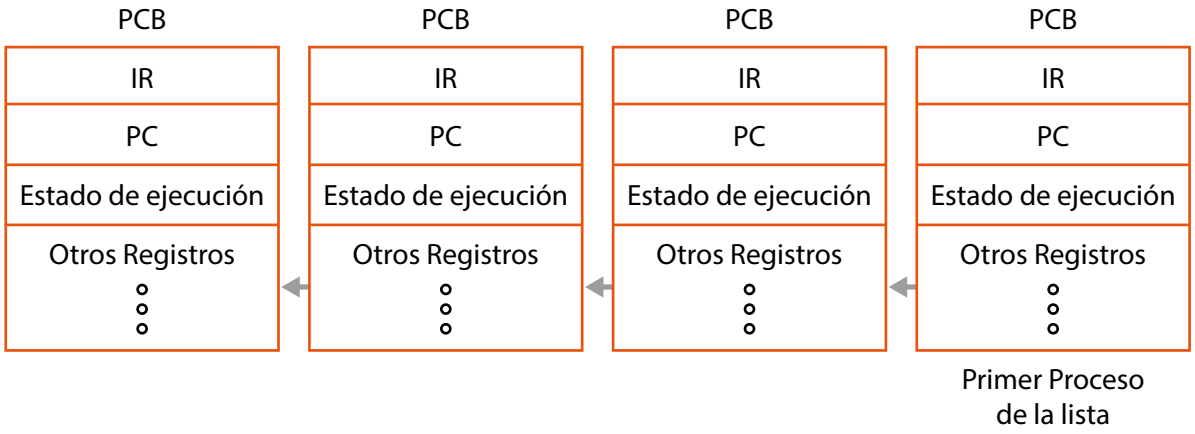


Figura 2. 4. Lista de procesos enlazados por sus PCB o Bloques de Control de Procesos. Fuente: elaboración propia.

Los procesos son movidos de un estado a otro por un componente del sistema operativo que lleva el nombre de despachador, quien se encarga de mirar las colas y actualizar el estado del proceso y otros campos de la PCB. La decisión de qué proceso debe ejecutarse y cuál debe detenerse, en función de una serie de normas y políticas, la realiza otro componente del sistema operativo que lleva el nombre de **Planificador de Procesos** o *Scheduler*.

2.2. El Planificador de procesos

La planificación es la forma en la que el sistema operativo identifica y gestiona los diferentes procesos que van a ser ejecutados en el ordenador; los organiza y, entre ellos, selecciona aquellos que se deban ejecutar en un cierto orden. Existen muy diferentes formas de planificar la ejecución de los procesos en base a diferentes enfoques; para identificar estos aspectos se utilizan los conceptos de **Política** y **Mecanismos**. Se le llama política a la estrategia que se utiliza para seleccionar cuál será el próximo proceso a ser colocado en ejecución. A la forma en la que esta estrategia es implementada, es a lo que se le llama un Mecanismo.

2.3. Políticas para la elección del próximo proceso a ejecutar

Las políticas establecen una serie de lineamientos que son usados para identificar y definir cuál será el próximo proceso que se ejecutará en el sistema.

Los mecanismos que se usan para implementar las políticas, a su vez se pueden dividir en dos grandes familias, los mecanismos expropiativos o apropiativos son aquellos en los que, si hay un proceso ejecutándose y llega otro proceso que se considera que es más importante, entonces se suspende la ejecución del proceso inicial que se estaba ejecutando para pasar a ejecutar el nuevo considerado de mayor importancia. Otra posibilidad es que las políticas o mecanismo sean no expropiativos o no apropiativos, en dicho caso, si un proceso comienza a ejecutarse, no podrá ser detenido y continuará ejecutándose, a menos que voluntariamente desee parar, hasta que termine,

o hasta que sea interrumpido, ya sea porque debe realizar una operación de E/S o porque alguna otra fuente de interrupciones o excepciones lo detiene.

2.3.1. Métricas de desempeño

Una métrica es un indicador o variable que proporciona información cuantitativa sobre el desempeño de un proceso de interés. Cuando un proceso se ejecuta en un ordenador muchas veces, se desea estimar que tan bien lo está haciendo; para evaluar o medir este desempeño es posible definir una serie de métricas como el tiempo de respuesta, el tiempo de ejecución, el *throughput*, entre otros. Pasemos a describir cada una de estas métricas y a identificar cómo medirlas.

Tiempo de ejecución: es el tiempo que transcurre desde que un proceso se inicia hasta que está totalmente culminado.

Tiempo de Espera: es la suma de los tiempos que transcurren desde que un proceso está listo para ejecutarse hasta que, efectivamente, comienza o continúa su ejecución dentro del procesador.

Tiempo de Respuesta: es la cantidad de tiempo que transcurre desde que el usuario introduce un requerimiento hasta que se obtiene la primera respuesta a este requerimiento. No es necesario que esté totalmente culminado.

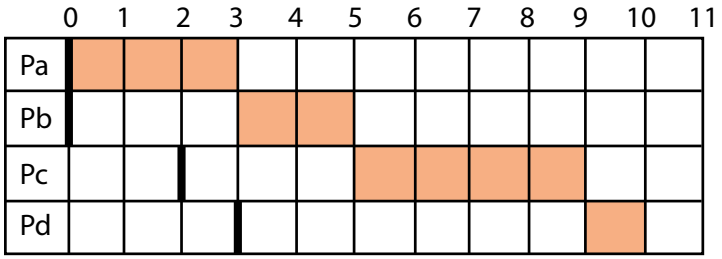
Throughput: es una medida de la cantidad de trabajo que puede hacer un procesador o un ordenador en una unidad de tiempo.

Estas métricas nos servirán de ayuda para identificar que tan bien se está ejecutando un proceso o un conjunto de procesos en el ordenador y que tan buenas o acertadas pueden llegar a ser alguna de las políticas y mecanismo que se usen en el sistema operativo.

Existe un gran conjunto de diferentes políticas para la selección del próximo proceso a ejecutar en un sistema, cada una con objetivos que se ajustan a requerimientos diferentes. A continuación, vamos a describir algunas de las políticas más usadas en la planificación de procesos.

2.3.2. PEPS: Primero en Entrar Primero en Salir (FCFS First Come First Serve)

La política PEPS o FCFS es una de las más tradicionales y simples; en ella, se cuenta con una cola de procesos en la que el primer proceso que entre en la fila será el primer proceso que sea seleccionado para ejecutarse y, el segundo en llegar, será el segundo en ejecutarse, y así sucesivamente. Se sigue el mismo esquema de una fila de atención al cliente, como suele ocurrir en la fila de un banco o de un supermercado, en la que el primer cliente que llega a la fila queda de primero y es atendido antes que las demás personas que puedan ir llegar luego a la fila. De allí en adelante, los procesos serán atendidos siguiendo estrictamente el orden en el que llegaron a la cola. A continuación, en la figura 2.5, se muestra un ejemplo de cómo es el orden de ejecución de un conjunto de procesos usando la política PEPS o FCFS.



Proceso	Tiempo de llegada	Tiempo de ejecución
Pa	0	3
Pb	0	2
Pc	2	4
Pd	3	1

Tiempo de ejecución
Pa = 3 - 0 = 3
Pb = 5 - 0 = 5
Pc = 9 - 2 = 7
Pd = 10 - 3 = 7

Tiempo promedio de ejecución

$$\frac{3 + 5 + 7 + 7}{4} = 5{,}5$$

Tiempo de espera
Pa = 0
Pb = 3
Pc = 3
Pd = 6

Tiempo promedio de espera

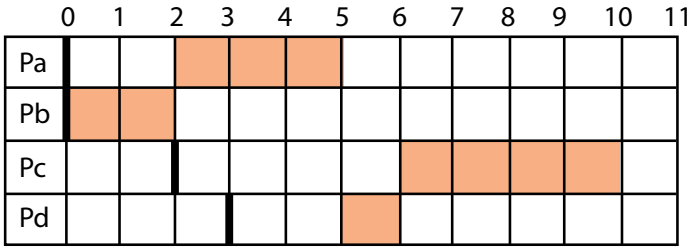
$$\frac{0 + 3 + 3 + 6}{4} = 3$$

Figura 2. 5. Ejemplo de aplicación de la Política PEPS o FCFS. Fuente: elaboración propia.

Esta es una política bastante tradicional y fácil de implementar, pero no siempre muestra los mejores resultados.

2.3.3. TMCP: Trabajo Más Corto Primero (SJF Shortest Job First)

En esta política, los procesos se ordenan tomando primero el trabajo más corto, es decir, aquel que pueda terminar antes. Esta política tiende a mejorar los valores de las métricas de desempeño de los sistemas de forma importante ya que, al dejar pasar primero los programas más cortos, esto hace que su tiempo de ejecución sea menor. Al ser estos tiempos menores, se reducen de forma importante los tiempos promedio de ejecución de todos los procesos que se estén trabajando, tal y como se puede apreciar en la figura 2.6.



Proceso	Tiempo de llegada	Tiempo de ejecución
Pa	0	3
Pb	0	2
Pc	2	4
Pd	3	1

Tiempo de ejecución
Pa = 5 - 0 = 5
Pb = 2 - 0 = 2
Pc = 10 - 2 = 8
Pd = 6 - 3 = 3

Tiempo promedio de ejecución

$$\frac{5 + 2 + 8 + 3}{4} = 4{,}5$$

Tiempo de espera
Pa = 2
Pb = 0
Pc = 4
Pd = 2

Tiempo promedio de espera

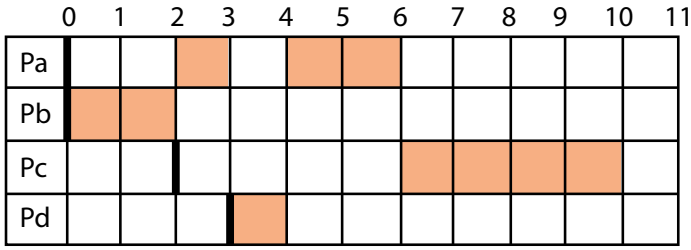
$$\frac{2 + 0 + 4 + 2}{4} = 2$$

Figura 2. 6. Ejemplo de aplicación de la Política TMCP o SJF. Fuente: elaboración propia.

La política del trabajo más corto se puede combinar con una política expropiativa para crear una nueva política.

2.3.4. TMCPA: Trabajo Más Corto Primero Apropiativa

En esta política, al igual que en la anterior, se le da prioridad a la ejecución del proceso al que le reste menos tiempo para ejecutarse, con la diferencia de que, si hay un trabajo ejecutándose y uno más corto se hace presente, entonces al proceso más largo se suspende, al menos temporalmente, para pasar a ejecutar el proceso al que le quede menos tiempo para culminar. Un ejemplo de la ejecución de esta política se muestra en la figura 2.7.



Proceso	Tiempo de llegada	Tiempo de ejecución
Pa	0	3
Pb	0	2
Pc	2	4
Pd	3	1

Tiempo de ejecución
Pa = 6 - 0 = 6
Pb = 2 - 0 = 2
Pc = 10 - 2 = 8
Pd = 4 - 3 = 1

Tiempo promedio de ejecución

$$\frac{6 + 2 + 8 + 1}{4} = 4{,}25$$

Tiempo de espera
Pa = 3
Pb = 0
Pc = 4
Pd = 0

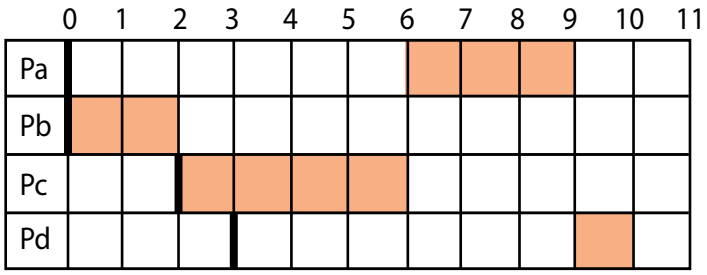
Tiempo promedio de espera

$$\frac{3 + 0 + 4 + 0}{4} = 1{,}75$$

Figura 2. 7. Ejemplo de aplicación de la Política TMCP Apropiativa. Fuente: elaboración propia.

2.3.5. Prioridades (Priority)

Hay ocasiones en que se desea dar preferencia a la ejecución de ciertos procesos sobre los otros; para implementar esto es posible definir una serie de diferentes niveles de prioridad según la importancia que se defina para cada proceso. En este esquema de prioridades se le asigna un número entero o prioridad a cada uno de los procesos y se selecciona el próximo proceso a ser ejecutado organizando la lista de procesos en función del nivel de prioridad que les ha sido asignado. En el ejemplo que presentamos en la figura 2.8, para el nivel de prioridad un número más bajo significará que el proceso tiene una mayor prioridad y, por tanto, debería ejecutarse primero.



Proceso	Tiempo de llegada	Tiempo de ejecución	Prioridad
Pa	0	3	3
Pb	0	2	1
Pc	2	4	2
Pd	3	1	4

Tiempo de ejecución
Pa = 9 - 0 = 9
Pb = 2 - 0 = 2
Pc = 6 - 2 = 4
Pd = 10 - 3 = 7

Tiempo de espera
Pa = 6
Pb = 0
Pc = 0
Pd = 6

Tiempo promedio de ejecución

$$\frac{9 + 2 + 4 + 7}{4} = 5{,}5$$

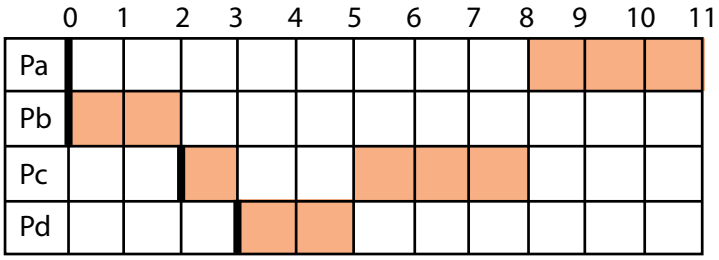
Tiempo promedio de espera

$$\frac{6 + 0 + 0 + 6}{4} = 3$$

Figura 2. 8. Ejemplo de aplicación de la Política de Prioridades. Fuente: elaboración propia.

2.3.6. Prioridades Apropiativas

En este esquema se le asigna una prioridad a cada proceso, la cual se usa para decidir qué procesos se ejecutarán primero en la CPU, pero en esta política, si ya un proceso se está ejecutando cuando llega un proceso de más alta prioridad, entonces el proceso que se está ejecutando es suspendido y el nuevo proceso con más prioridad se pasa a ejecutar. Esta política permite que los procesos prioritarios sean atendidos tan pronto como sea posible; sin embargo, puede llegar a ocurrir que procesos de baja prioridad nunca lleguen a ejecutarse si constantemente están llegando procesos de más alta prioridad. Un ejemplo del uso de esta política se puede apreciar en la figura 2.9.



Proceso	Tiempo de llegada	Tiempo de ejecución	Prioridad
Pa	0	3	4
Pb	0	2	2
Pc	2	4	3
Pd	3	2	1

Tiempo de ejecución
Pa = 11 - 0 = 11
Pb = 2 - 0 = 2
Pc = 8 - 2 = 6
Pd = 5 - 3 = 2

Tiempo de espera
Pa = 8
Pb = 0
Pc = 2
Pd = 0

Tiempo promedio de ejecución

$$\frac{11 + 2 + 6 + 2}{4} = 5{,}25$$

Tiempo promedio de espera

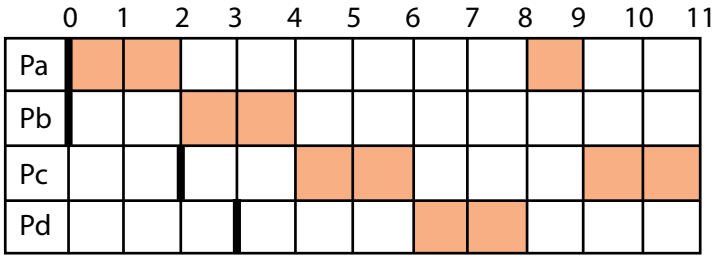
$$\frac{8 + 0 + 2 + 0}{4} = 2{,}5$$

Figura 2. 9. Ejemplo de aplicación de la Política de Prioridades Apropiativas. Fuente: elaboración propia.

2.3.7. Round Robin

Prácticamente todas las políticas anteriores se basan en favorecer la ejecución de algunos procesos, pero sacrificando o retardando la ejecución de otros que, por alguna razón, se consideran menos importantes. En este sentido, es deseable en algunos casos contar con alguna política que sea más justa y que trate de forma más uniforme a los procesos en el sistema. Para ello, surgió la política de Round Robin que divide el uso de la CPU en periodos de tiempo iguales, a los cuales se les da el nombre de *quantum* y reparte un quantum a cada proceso que desea ser ejecutado siguiendo un orden particular. Después de transcurrido el quantum, se detiene al proceso en ejecución y se pasa a ejecutar el próximo proceso de la lista. Una vez que se ha recorrido toda la lista, se vuelve a empezar con el primer proceso de la lista nuevamente.

A continuación, en la figura 2.10, mostramos un ejemplo del uso de Round Robin usando un tamaño de quantum de 2 unidades de tiempo.



Proceso	Tiempo de llegada	Tiempo de ejecución
Pa	0	3
Pb	0	2
Pc	2	4
Pd	3	2

Tiempo de ejecución
Pa = 9 - 0 = 9
Pb = 4- 0 = 4
Pc = 11 - 2 = 9
Pd = 8 - 3 = 5

Tiempo promedio de ejecución

$$\frac{9 + 4 + 9 + 5}{4} = 6{,}75$$

Tiempo de espera
Pa = 6
Pb = 2
Pc = 5
Pd = 3

Tiempo promedio de espera

$$\frac{6 + 2 + 5 + 3}{4} = 4$$

Figura 2. 10. Ejemplo de aplicación de la Política Round Robin. Fuente: elaboración propia.

En los esquemas de planificación de procesos que hemos visto hasta ahora, existe una única cola de donde se eligen los procesos que se ejecutarán. Sin embargo, también existe la posibilidad de definir esquemas en los que exista más de una cola de procesos. En estos esquemas, es posible que la política que se use en cada una de las colas para seleccionar un proceso a ejecutarse sea la misma, pero también es posible que cada una de las colas pueda usar una política diferente.

2.3.8. Colas Multiniveles

En los esquemas de colas multinivel existe más de una cola y, en cada una de las colas, se puede definir una política específica para seleccionar a cuál de los procesos le toca el turno de ejecutarse en la próxima oportunidad. En el esquema más simple, los procesos que entran en una cola permanecen allí todo el tiempo y se define un conjunto de prioridades o reglas para identificar de cuál de las colas se tomará un proceso para ejecutarse.

Veamos un ejemplo para comprender mejor el uso de las colas multinivel. Se definen dos colas; una primera, para procesos que hacen operaciones de Entrada/Salida, es decir, procesos en los que los programas hacen un cálculo, luego interactúan, leen o escriben un dato en un dispositivo y, finalmente, realiza un cálculo antes de terminar su ejecución. Se presenta también el uso de una segunda cola de procesos *batch*, en la que los procesos solo realizan una serie de cálculos y luego terminan su ejecución.

En este esquema, se les da prioridad a los procesos con Entrada/Salida, ya que los procesos *batch* fueron dejados por sus usuarios y no los están esperando; lo que sí pueden estar haciendo los usuarios de los procesos que hacen E/S. En la cola de procesos de E/S se eligen los procesos a ejecutar primero en

función de la política PEPS o FCFS; en las colas de procesos *batch* se elige al siguiente proceso usando la política de TMCP o SJF. A continuación, ilustramos esta situación con un conjunto de proceso, tal como se muestra en la figura 2.11, adicionalmente la figura 2.12 permite apreciar cómo es el esquema de colas multinivel para estos procesos.

Proceso	Tiempo de Llegada	Tiempo de Ejecución en la CPU	Tiempo de Ejecución de E/S	Tiempo de Ejecución en la CPU
P_ESa	3	3	4	2
P_ESb	2	2	5	3
P_Bc	0	4		
P_Bd	3	6		

Figura 2. 11. Ejemplo de procesos para un esquema de colas multinivel. Fuente: elaboración propia.

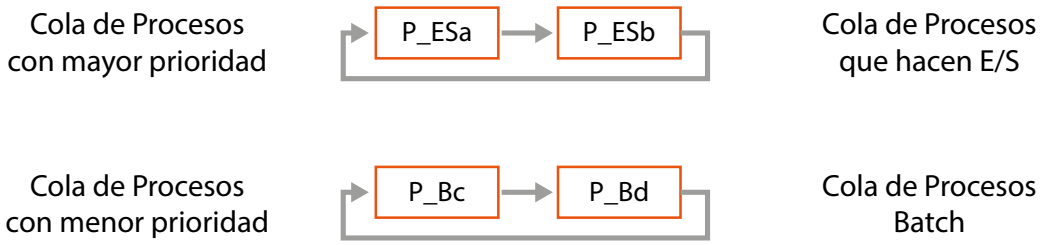


Figura 2. 12. Ejemplo en el que están armadas las colas multinivel. Fuente: elaboración propia.

A partir de estos datos, la ejecución de este conjunto de procesos se realizará como se presenta en la figura 2.13.

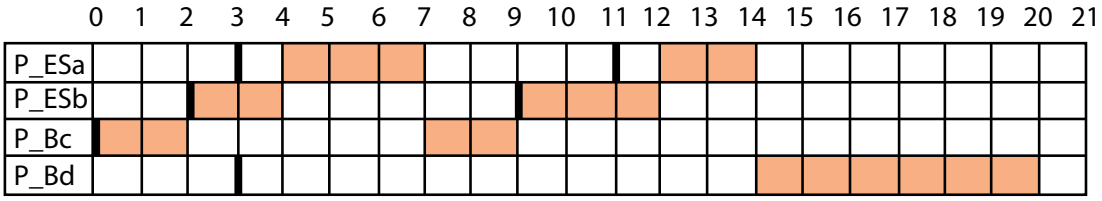


Figura 2. 13. Ejecución de dos procesos con E/S y dos procesos Batch usando un esquema de colas multinivel. Fuente: elaboración propia.

En este caso, los procesos que hacen E/S son los prioritarios, pero en el instante 0 no hay ninguno de estos procesos disponible para su ejecución, por eso se inicia la ejecución de la carga de trabajo del sistema con el proceso P_Bc, que es de la cola de los procesos *batch*. Esta segunda cola se va a atender siguiendo la política del trabajo más corto. En el instante 0 el trabajo más corto, y realmente el único disponible, es el proceso P_Bc, por lo que este comienza a ejecutarse. En el instante 2, llega el primer proceso que hace E/S, que es el proceso P_ESb, y desaloja al proceso P_Bc de la CPU porque, como dijimos antes, los procesos con E/S son prioritarios.

La cola de procesos con E/S se atiende usando la política FCFS o PEPS, por ello, el proceso P_ESb se ejecuta hasta que termina de ejecutar su primer grupo de operaciones en la CPU que, como lo indica la figura 2.12, es de 2 unidades de tiempo, lo que ocurre en el tiempo 4, para este momento hay ya otro proceso (P_ESa) que hace E/S y que está listo para ejecutarse desde el instante 3, por lo que, en el instante 4, el proceso P_ESa comienza su ejecución hasta el instante 7. En ese momento, al terminar

su primer grupo de operaciones en la CPU, suspende su ejecución. En este instante 7 no hay ningún proceso que haga E/S disponible para su ejecución, situación que seguirá así hasta el instante 9, por eso se pasa a ejecutar un proceso de la segunda cola, es decir, de la cola de procesos *batch* y aquí ya están disponibles los procesos P_Bc y P_Bd para ejecutarse, pero el más corto de ambos es el proceso P_Bc, por lo que es este proceso el que se ejecuta primero. En el instante 9 coinciden la terminación del proceso P_Bc con la llegada del proceso P_ESb; por estas dos razones, el proceso P_Bc abandona la CPU y se comienza a ejecutar el proceso P_ESb por 3 unidades de tiempo hasta que finaliza su ejecución; en este instante 12, el proceso P_ESa ya ha finalizado su operación de E/S y está listo para ejecutarse, por lo que se le asigna la CPU hasta el instante 14, donde finaliza su ejecución.

Ya finalizados todos los procesos de la cola de procesos con E/S, solo resta ejecutar los procesos de la cola de procesos *batch*, de esta cola el proceso P_Bc ya finalizó su ejecución en el tiempo 9, por lo que solo resta ejecutar el proceso P_Bd, que continúa ejecutándose hasta el instante 20, cuando finaliza su ejecución; con esto se termina la ejecución de todos los procesos considerados en este caso de estudio.

2.3.9. Colas Multiniveles con retroalimentación

En el esquema de colas multinivel que acabamos de ver, una vez que un proceso entra en una de las colas, permanece allí hasta finalizar su ejecución. Sin embargo, un enfoque interesante podría consistir en que los procesos que entren en una de las colas, puedan cambiarse de una cola a otra en función de cual fuera su comportamiento en el sistema, esto permitiría favorecer ciertos comportamientos y penalizar otros. Para apreciar un ejemplo del uso de las colas multinivel con retroalimentación, miremos el conjunto de procesos que se describen en la figura 2.14.

Proceso	Tiempo de Llegada	Tiempo de Ejecución en la CPU	Tiempo de Ejecución de E/S	Tiempo de Ejecución en la CPU
Pa	0	1	4	5
Pb	2	2	5	3
Pc	0	4		
Pd	3	6		

Figura 2.14. Conjunto de procesos para mostrar un ejemplo del uso de colas multinivel con retroalimentación. Fuente: elaboración propia.

Veamos cómo es la ejecución de este conjunto de procesos suponiendo que contamos con un sistema que posee dos colas. La cola 1 es la de más alta prioridad. Los procesos en la cola 1 se ejecutan siguiendo una variación de la política de Round Robin con un quantum de 3 unidades de tiempo en la que un proceso que se ejecute por un máximo de 3 unidades de tiempo es pasado automáticamente a la cola 2 de menos prioridad, ya que es considerado un proceso con muchas operaciones de CPU. Si un proceso que está en la en la cola 1 se ejecuta por un periodo menor a las 3 unidades de tiempo, se considera un proceso que interactúa frecuentemente y permanece en esta cola 1. Los procesos de la cola 1 tendrán prioridad sobre los procesos de la cola 2 y siempre se ejecutarán de forma preferente. Los procesos de la cola 2 se ejecutan siguiendo la política de trabajo más corto TMCP o SJF. Finalmente, si un proceso que está en la cola 2, cuando se ejecuta, consume menos de tres unidades de tiempo, entonces es promovido a la cola 1 para que continúe allí su ejecución la próxima vez. Cualquier

proceso que llegue al sistema inicial en la cola 1 y podrá ser llevado a la cola 2 en función de cuál es su comportamiento al ejecutarse. Para apreciar adecuadamente este proceso, veamos en las figuras 2.15 y 2.16 cómo se van armando estas dos colas al irse ejecutando los procesos. En la figura 2.17 podremos observar cómo es la ejecución de los cuatro procesos que se describen en este ejemplo.

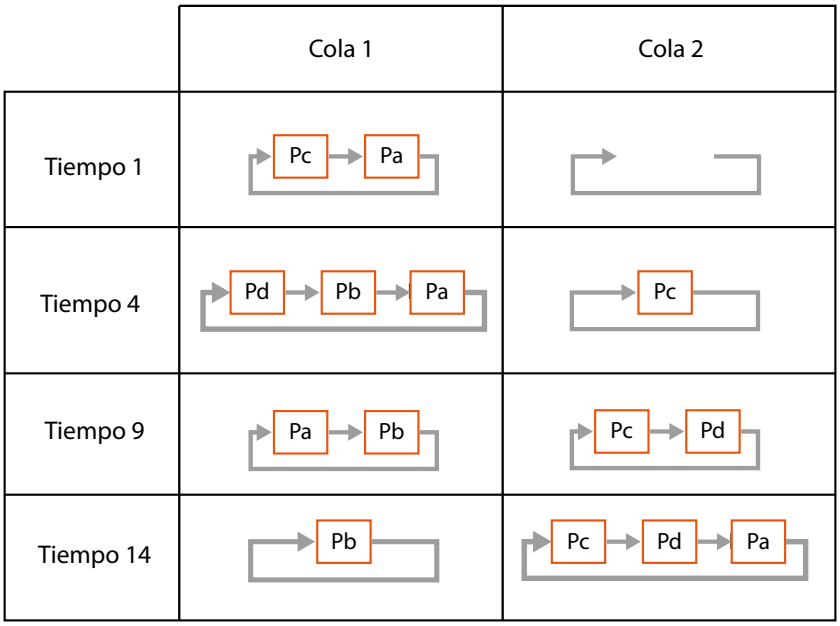


Figura 2. 15. Movimiento de los procesos en las colas en el esquema de colas multinivel con retroalimentación hasta el instante de tiempo 14. Fuente: elaboración propia.

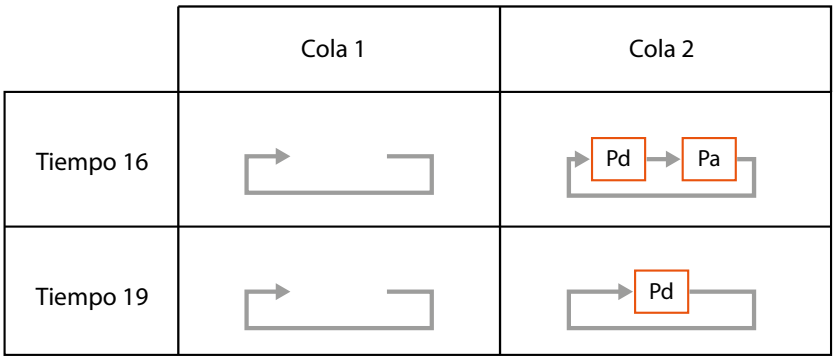


Figura 2. 16. Movimiento de los procesos en las colas en el esquema de colas multinivel con retroalimentación desde el instante de tiempo 16 al 19. Fuente: elaboración propia.

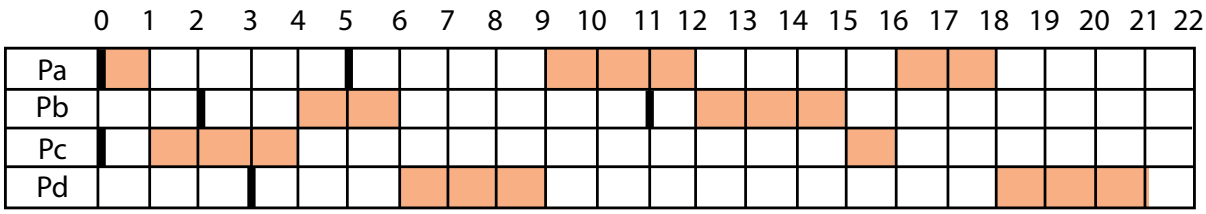


Figura 2. 17. Ejemplo de ejecución de cuatro procesos usando un esquema de dos colas multinivel con retroalimentación. Fuente: elaboración propia

Aquí podemos apreciar como en el instante 0 llegan los procesos Pa y Pc al sistema y son ingresados ambos en la cola 1; el proceso Pa inicia su ejecución por ser el primero en la lista. Aunque los procesos Pa y Pc llegan exactamente en el mismo instante de tiempo, al ser iguales se puede elegir el que tenga un menor identificador para desempatar la decisión de qué proceso iniciar primero.

En el instante 1 el proceso Pa interrumpe su ejecución por una operación de E/S y como no consumió su quantum de 3 unidades de tiempo, permanece entonces en la cola 1. En ese instante, comienza su ejecución el proceso Pc que se ejecutará por 3 unidades de tiempo, después de las cuales, en el instante 4, es suspendida su ejecución y es pasado a la cola 2 por haber consumido por completo su quantum de 3 unidades de tiempo. En los instantes 2 y 3 llegan los procesos Pb y Pd, siendo ingresados a la cola 1. En el instante 4, el proceso Pa no está aún listo, por lo que se inicia el proceso Pb por haber llegado antes que el proceso Pd, que también está listo para ejecutarse en ese instante. En el instante 5, el proceso Pa termina su operación de E/S, pero aún no se ejecuta debido a que continúa en ejecución el proceso Pb. En el instante 6, el proceso Pb se detiene para hacer una operación de E/S e inicia su ejecución el proceso Pd; este se ejecuta por 3 unidades de tiempo para ser detenido y pasado a la cola 2; en ese momento, que corresponde al instante 9, el proceso Pa continúa su ejecución y la finaliza en 3 unidades de tiempo, lo que nos ubica en el instante 12; para ese momento, en el instante 11, el proceso Pb ha terminado su operación de E/S y queda listo para iniciar su ejecución; en el instante 12, al terminársele el quantum al proceso Pa, continua su ejecución el proceso Pb por 3 unidades de tiempo hasta finalizar su ejecución, en el instante 15. Al terminar el proceso Pb, la cola 1 queda sin procesos disponibles para ejecutar, por lo que se empiezan a ejecutar los procesos de la Cola 2, comenzando por el proceso Pc, que es el proceso al que le queda un menor tiempo de ejecución que termina en una unidad de tiempo; después, le toca el turno al proceso Pa, al que le quedan solo dos unidades de tiempo de ejecución y, finalmente, el proceso Pd, al que le quedan 3 unidades de tiempo. Con esto finaliza la ejecución de todos los procesos considerados en este ejemplo.

2.4. Concurrencia y Sincronización

En un ordenador pueden existir diferentes procesos, ejecutándose de forma más o menos simultánea en un instante de tiempo. A este tipo de procesos se le suele dar el nombre de procesos **Concurrentes**. Este término se usa no solo para referirse a procesos con eventos que ocurren exactamente al mismo tiempo, sino más bien para referirse a procesos que poseen dos o más eventos que pueden ser independientes, cuyo orden de ejecución no es siempre el mismo y que puede variar según una serie de factores que no son siempre iguales, por lo que es difícil predecir el orden relativo en el que ocurrirán en cada ejecución. Se habla de concurrencia porque en algunos ordenadores, si hay un solo procesador, será difícil que dos eventos pasen exactamente al mismo tiempo, pero aun así, se pueden presentar casos en los que la ejecución de procesos concurrentes puede generar algunos inconvenientes, como se podrá percibir con el siguiente ejemplo.

Imaginemos que tenemos un local para eventos que posee una entrada y una salida para personas que van al lugar, tal como se aprecia en la figura 2.18. Asumimos que hay torniquetes en la entrada y en la salida que controlan la cantidad de personas que van ingresando y saliendo del local; tanto la entrada como la salida están conectadas a una oficina interna, donde se almacena la cantidad de personas que hay en todo momento en el local. Imaginemos que, en un momento dado, hay 20 personas dentro del local y sale una persona, quedan entonces 19; sale otra, quedan 18, entra una más y vuelven a ser 19,

pero ahora imaginemos que, exactamente al mismo tiempo, está entrando una persona y saliendo otra. Justo en ese instante, la entrada mira qué cantidad de personas hay en el local y ve el número19; en ese mismo momento, la salida mira cuántas personas están en el local y también ve el número 19; ahora la entrada incrementa ese número en 1 (20) y lo guarda en la variable que almacena la cantidad de personas en el local, y solo instantes después la salida toma el valor 19 que había visto y le resta 1 (18) para guardarlo en la variable que almacena la cantidad de personas que hay en el local, quedando registrado que hay 18 personas, en lugar de las 19 que realmente hay. Fíjese que, si ambos lados miran el valor central, pero es la salida la que resta y actualiza primero el valor y, luego, solo algunos instantes después la entrada incrementa el valor leído y lo almacena, entonces el resultado es que estarán registradas 20 personas en lugar de las 19 que realmente hay. Esta situación es lo que se llama una **Condición de Carrera adicional**, la cual no ocurriría si cada programa consulta el valor de personas y lo actualiza sin que otros procesos interfieran y sin ser interrumpido. En el momento que logremos diseñar un mecanismo para que esto no ocurra, diremos que estamos sincronizando ambos procesos. Esta situación se da porque la salida y la entrada están accediendo a un contador que es común a ambas estaciones y lo hacen de una forma que no es la más adecuada.

Figura 2. 18. Diagrama del local de fiestas con una entrada y una salida. Fuente: elaboración propia.

Esta situación que describimos aquí, o alguna similar, también puede ocurrir con mucha frecuencia dentro de los ordenadores, ya que normalmente dentro de ellos pueden existir una o más CPUs que están ejecutando programas, y estos programas pueden ver detenida su ejecución según lo determine el planificador de procesador. En esta situación, si la forma o el orden en el que se ejecutan estos programas pueden variar los resultados de su ejecución, decimos que estamos en presencia de una Condición de Carrera o (*Race Condition*).

Veamos algunos conceptos básicos asociados a la ocurrencia de problemas de concurrencia:

Operación atómica: es aquella operación o conjunto de operaciones que ocurren sin que puedan ser interrumpidas de alguna forma en el ordenador. Esto puede ocurrir porque durante la ejecución de un programa, de alguna manera se inhibe al planificador o al mecanismo de interrupciones del ordenador para que no interrumpan la ejecución del programa, de forma que no pueda ser detenido o interrumpido en lo que está haciendo.

Sección (o región) crítica: es aquella parte de un programa en la que se comparten recursos con otros programas o procesos y en la cual, si no se establece ningún tipo de control, pueden ocurrir condiciones de carrera cuando hay varios procesos concurrentes en el ordenador.

Recurso compartido: son aquellos recursos del computador como áreas de memoria, estructuras de datos, variables y dispositivos, que son utilizados por diferentes programas o procesos que se ejecutan concurrentemente en el ordenador y que pueden ser el origen de una condición de carrera.

2.4.1. Sincronización de procesos

Como se describió en la sección de concurrencia y sincronización, específicamente en el ejemplo del local de eventos que se muestra en la figura 2.18 se hace necesario contar con un conjunto de funciones para implementar el control de las personas que entran en este local. En este caso, podemos tener una rutina o función para contar la cantidad de personas que entran y otra para contar la cantidad de personas que salen, además de una variable común que almacene cuantas personas hay dentro del local. La figura 2.19 muestra un posible ejemplo de implementación de estas rutinas. Pero si estas rutinas son invocadas por procesos diferentes, uno en la entrada y otro en la salida del local, al trabajar con un recurso crítico, en este caso en la variable común numPersonas, se pueden presentar condiciones de carrera como las que se describieron en la sección de concurrencia y sincronización.

```
int numPersonas = 0; // Variable común

int agregarPersona()   {          int restarPersona()   {
    numPersonas = nupersonas + 1;     numPersonas = nupersonas - 1;
}                                 }
```

Figura 2. 19. Funciones que implementan el conteo de personas que entran en el local. Fuente: elaboración propia.

Para evitar que una condición de carrera pueda presentase, existen una serie de mecanismos, presentes en los sistemas operativos, que pueden ser utilizados para poner de acuerdo a los programas y procesos, de forma tal que no existan problemas con la ejecución de regiones críticas en procesos que comparten algunos recursos. Entre ellos, se encuentran los semáforos, las regiones críticas condicionales y los monitores.

Imaginemos que pudiéramos contar con dos primitivas que lleven el nombre de PedMutex para pedir exclusión mutua y LibMutex para liberar un área de exclusión mutua; de ser así, si tuviéramos un caso como el del local de eventos, podríamos garantizar que solo uno de los programas accede a la variable numPersonas usando las rutinas PedMutex y LibMutex y una variable adicional np de la forma en la que se muestra en la figura 2.20.

```
  int numPersonas = 0; // Variable común
  int np = 0;          // Variable para garantizar exclusión mutua

int agregarPersona()   {          int restarPersona()   {
     PedMutex(np);                     PedMutex(np);
     numPersonas = nupersonas + 1;     numPersonas = nupersonas - 1;
     LibMutex(np);                     LibMutex(np);
}                                 }                                 }
```

Figura 2. 20. Funciones que implementan el conteo de personas con excusión mutua. Fuente: elaboración propia.

El objetivo de estos mecanismos es que las regiones críticas de los procesos no se puedan ejecutar al mismo tiempo, logrando lo que se llama exclusión mutua, tal como se aprecia en la figura 2.21.

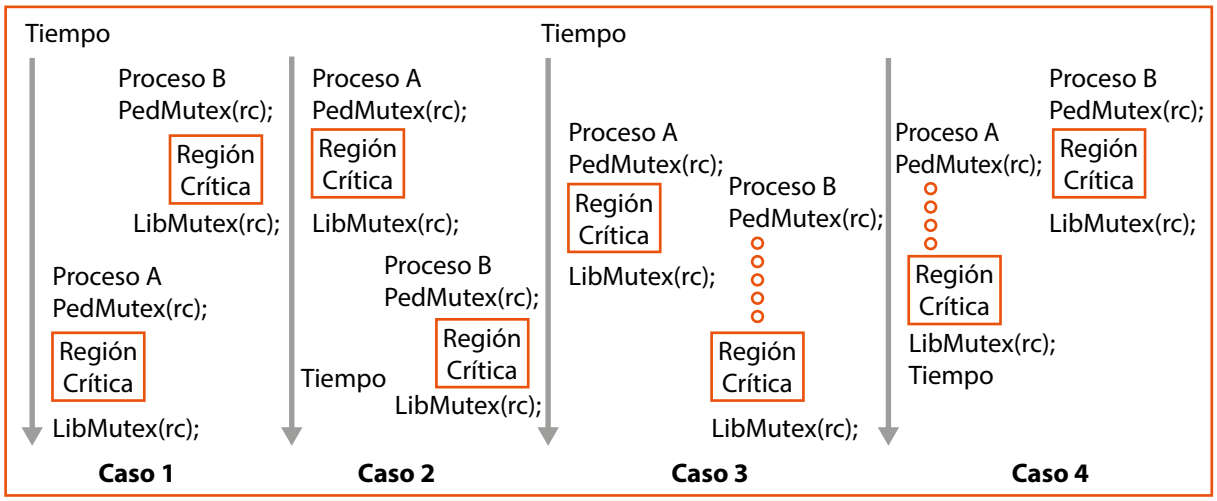


Figura 2. 21. Exclusión mutua de las regiones críticas de dos procesos. Fuente: elaboración propia.

Recordemos que, si dos procesos son concurrentes, es porque se pueden ejecutar de forma independiente uno del otro, pero esto puede ocasionar en algunos casos condiciones de carrera. En los casos 1 y 2 de la figura 2.21, la forma en la que se ejecutaron los procesos A y B permitió que cada uno pudiera ejecutar su sección crítica sin que el otro lo hiciera, por lo que no se generó ninguna condición de carrera. Sin embargo, en los casos 3 y 4, los procesos A y B se han debido ejecutar de forma simultánea, paralela o concurrente, lo que hubiera generado una condición de carrera, pero al usar la primitiva PedMutex(rc) sobre un mismo recurso rc, al primer programa que invocará a la rutina PedMutex(rc) se le permite el acceso a su región critica pero, al segundo proceso que invoca la misma rutina PedMutex(rc), se le suspende su ejecución hasta que el primero de los procesos termine su región critica, lo que es notificado al ejecutar la rutina LibMutex(rc). Este mecanismo permite evitar que ocurran condiciones de carrera entre los procesos A y B que puedan ocasionar problemas y datos inconsistentes entre ellos. Con este mecanismo, si es correctamente aplicado, es posible poner de acuerdo diferentes procesos para que se ejecuten de la forma adecuada y sin efectos colaterales, a esto es a lo que se le llama **Sincronizar la ejecución de procesos**.

Los nombres que han recibido primitivas de sincronización como las que hemos descrito aquí de PedMutex(x) y LibMutex(x) en diferentes diseños, lenguajes e implementaciones han sido numerosos; por ejemplo, Dijkstra plantea un diseño basado en el mecanismo de semáforos con operaciones P(s) y V(s); algunas versiones del lenguaje C emplean un mecanismo similar al usar rutinas wait y signal. Por su parte, en el lenguaje Java se emplea el mecanismo de monitores y la definición de métodos *Synchronized*.

A pesar de que los mecanismos de sincronización resolvieron una buena cantidad de los inconvenientes que ocurren al trabajar con proceso concurrentes, hay una serie de otros inconvenientes que requieren de un tratamiento adicional; aquí haremos referencia dos de ellos: los **Bloqueos Mutuos**, también conocidos como abrazos mortales o *Dead Lock* y la condición de Inanición o *Starvation*.

2.5. Bloqueos Mutuos o Abrazo Mortal

La situación de bloqueo mutuo, también conocida como abrazo mortal, es una condición que puede ocurrir en un sistema cuando los procesos requieren de varios recursos para finalizar su ejecución, pero la forma en la que el sistema operativo ha asignado los recursos ha sido de tal forma, que varios procesos solicitan recursos que no pueden obtener porque han sido asignados a otros procesos que, a su vez, no pueden continuar su ejecución para liberarlos, debido a que esperan por otros recursos que los primeros procesos no van a liberar. Veamos un ejemplo para ilustrar esta situación.

Consideremos una situación en la que en un sistema hay dos procesos Pa y Pb y dos recursos R1 y R2 que pueden ser una estructura de datos compartida o un dispositivo como una unidad de disco. En esta situación, primero se ejecuta el proceso Pa y, en el instante de tiempo 5, solicita el recurso R1, el cual le es asignado por el sistema operativo en el tiempo 6. Luego, el sistema operativo, en el instante de tiempo 10, detiene la ejecución del proceso Pa y pasa a ejecutar el proceso Pb que, a su vez, solicita la asignación del recurso R2 y el sistema operativo se lo asigna en el instante 11. A continuación, el proceso Pa retoma su ejecución y, en el instante 15, solicita el recurso R2, recurso que ya fue asignado al proceso Pb, pero que aún lo mantiene por lo que no se le puede asignar ahora al proceso Pa y, por ello, en el instante 16, el sistema operativo detiene su ejecución, a la espera de que el recurso R2 sea liberado por el proceso Pb. Algunos instantes después, en el tiempo 20, el proceso Pb solicita el recurso R1 al sistema operativo; este recurso ya fue asignado al proceso Pa, que aún lo mantiene y solo lo liberará cuando haya terminado de usarlo, pero esto no va a ocurrir, porque para que Pa termine debe tener el recurso R2 que Pb posee; por ello, el proceso Pb también es detenido en el instante 21. En ese momento, los procesos Pa y Pb están los dos detenidos, cada uno esperando que el otro termine para que libere el recurso que retiene, pero esto no va a ocurrir, dado que el proceso que debe terminar para liberar el recurso también está bloqueado. Para ilustrar mejor esta situación veamos cómo esta se representa en la figura 2.22.

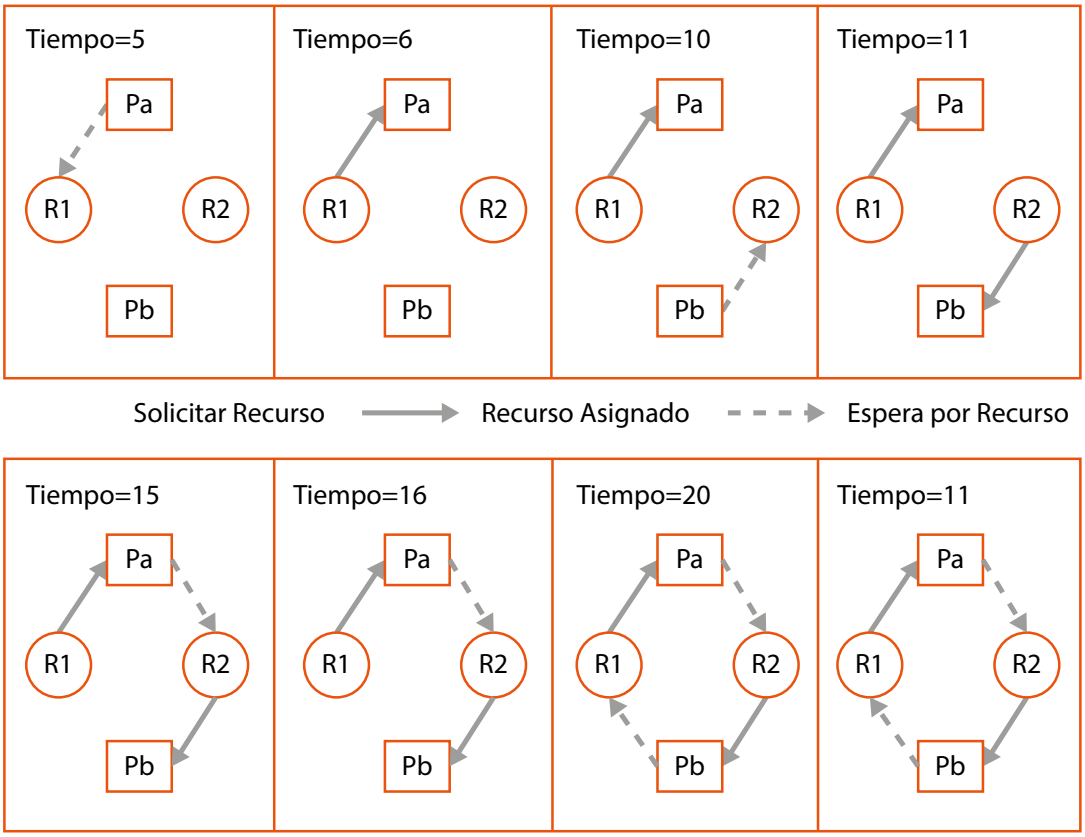


Figura 2. 22. Creación de una situación de interbloqueo o abrazo mortal cuando dos procesos solicitan recursos al sistema operativo para su ejecución. Fuente: adaptado de G.Wolf et al. (2015).

Como podemos observar en el instante 21, el proceso Pa conserva el recurso R1 para su uso y está solicitando el recurso R2. Como no puede obtener el recurso R2 para continuar su ejecución, el proceso Pa se bloquea. Pero, ¿podrá en algún momento terminar el proceso Pb, que es el proceso que mantiene al recurso R2 para que, al ser liberado, el proceso Pa pueda continuar y terminar su ejecución? La respuesta es no, porque, a su vez, el proceso Pb está bloqueado esperando que el proceso Pa libere el recurso que él necesita.

En resumen, Pa espera por algo que Pb tiene y Pb espera por algo que Pa tiene, por la forma en la que ambos esperan, ninguno podrá avanzar y los procesos se quedarán estancados en esta situación de forma indefinida, a menos que se tome alguna acción correctiva. La misma situación puede ocurrir con tres o más procesos siempre que entre ellos ocurra una situación de espera circular, como la que se ha descrito en el último cuadro de la figura 2.22.

2.6. Estrategias para manejar situaciones de Bloqueos Mutuos o Abrazos Mortales

Para atacar y solventar una situación de bloqueo mutuo o abrazo mortal hay varias posibles estrategias: Prevenir, Evitar, Detectar y solucionar y, finalmente, la Estrategia del Avestruz.

2.6.1. Prevenir

La estrategia de prevenir implica ejecutar una serie de acciones de forma que la condición de interbloqueo no pueda ocurrir nunca. Para ello, se pueden seguir diferentes esquemas de operación, los cuales describiremos a continuación.

El primero de estos esquemas consiste en serializar la ejecución de los procesos; si un proceso inicia y no se interrumpe, entonces puede solicitar todos los recursos que necesita y obtenerlos, por lo que no llega a bloquearse y, al terminar, puede liberar todos los recursos que usó para que estén disponibles para otros procesos. Este esquema de operación tiene el problema de que, al serializar los procesos, algunos componentes del ordenador pueden quedar ociosos por ciertos periodos de tiempo y, debido a esto, los grupos de procesos pueden tardar un poco más en ejecutarse al limitarse del todo la concurrencia de procesos en el ordenador.

El segundo esquema se basa en el hecho de que cada proceso solicite, al comenzar su procesamiento, todos los recursos que requiera de una sola vez; de esta forma, cada uno de los procesos, justo al comenzar su ejecución, o toma todos los recursos que requiere e inicia su ejecución, por lo cual se garantiza que terminará ya que contará con todos los recursos que pueda necesitar o, por el contrario, si no cuenta con alguno de los recursos, no se iniciará y, por tanto, al no recibir y mantener recursos reservados, no podrá bloquear la ejecución de otros procesos.

Un tercer esquema para prevenir la ocurrencia de bloqueos mutuos es la organización y solicitud jerárquica de recursos. En este esquema, cada proceso puede solicitar un recurso que tiene un nivel de jerarquía k; si este recurso le es asignado, entonces solo le será posible solicitar recursos de mayor jerarquía y nunca de menor jerarquía; al seguir un orden lineal en la jerarquía de solicitudes de recursos, no es posible que ocurra que un proceso Px se bloquee esperando la liberación de un recurso que mantiene otro proceso Py y, a su vez, el proceso Py esté bloqueado por un recurso que mantiene el proceso Px, porque si el proceso ya tiene un recurso de mayor jerarquía, no puede ahora solicitar uno de menor jerarquía que ya el proceso Px tiene asignado.

¿Cómo se sabe que es de menor jerarquía? Porque el proceso Px lo solicitó antes que el nuevo recurso que está ahora solicitando. Este esquema también garantiza que no pueda existir una espera circular de procesos. Pero, aunque elimina la posibilidad de que lleguen a existir interbloqueos, igual que el esquema anterior, limita la cantidad de procesos concurrentes que pueden existir en el sistema.

2.6.2. Evitar

La segunda gran estrategia es un poco menos limitante y deja que cualquier proceso pueda solicitar los recursos que desee en cualquier momento, pero cada vez que un proceso desee solicitar un recurso, el sistema operativo debe evaluar si la asignación de recursos que le están solicitando puede llegar a ser peligrosa, es decir, si la asignación puede conducir a una situación de bloqueo mutuo; de ser así, entonces la asignación no se realiza o se pospone para otro momento.

Si la asignación no representa ningún problema, se considera segura y se realiza. Para ello, el sistema operativo debe contar con una serie de estructuras de datos que le permitan registrar cuáles son los recursos que le han sido asignados a cada uno de los procesos que están en el sistema y, antes de

hacer una nueva asignación, se debe hacer un recorrido por estas estructuras para identificar si la nueva asignación implica la creación de una espera circular.

2.6.3. Detectar y corregir

En esta estrategia se permite que los procesos soliciten y se les asigne los recursos que requieran sin ningún tipo de control; sin embargo, cada vez que se hace una asignación de recursos a un proceso, se registran los datos de dicha asignación. En este caso, es posible que ocurra una situación de bloqueo mutuo entre dos o más procesos que pudieran quedar bloqueados, pero en esta estrategia el sistema operativo periódicamente puede recorrer las estructuras de datos para identificar si hay una espera circular; esto es fácilmente detectable al ubicar ciclos en el grafo de asignación de recursos a procesos.

Cuando esta situación se detecte, se pueden seguir tres caminos diferentes. El primero, consiste en terminar todos los procesos involucrados en el bloqueo mutuo, esto liberará los recursos retenidos y los procesos podrán intentar reiniciar su ejecución; sin embargo, esto no impide que la situación de bloqueo mutuo pueda volver a repetirse. Un segundo camino, consiste en retroceder la ejecución de los procesos hasta el último punto seguro que se haya registrado en el que los procesos no estén en un bloqueo mutuo, pero esto implica que el sistema debe estar almacenando constantemente los estados de los procesos para permitir hacer un proceso de recuperación como el que aquí se describe; esta estrategia puede ser costosa en el consumo de recursos de un ordenador y requiere también de mucha atención por parte del sistema operativo. El último de los caminos, consiste en ir eliminando uno a uno los procesos que están en el bloqueo mutuo hasta que este efectivamente se rompe. Así, al menos algunos procesos pueden continuar su ejecución. La desventaja es que los procesos eliminados deben iniciar su ejecución desde cero, sin importar cuánto hubieran avanzado.

Todos estos caminos contemplan oportunidades al permitir que los procesos puedan volver a ejecutarse tras haber estado suspendidos, pero también riesgos ya que, al terminar un proceso, los pasos intermedios que este proceso haya realizado pueden hacer que el sistema llegue a quedar en estados inconsistentes si no se tratan de la forma adecuada.

2.6.4. La Estrategia del Avestruz

Esta estrategia se basa en la falsa creencia de que un avestruz, cuando se asusta, esconde la cabeza en un agujero en la tierra para que no lo vean. Esta es una creencia difundida por algunos dibujos animados y por observaciones que no han sido cuidadosas, pero que no son reales. La estrategia del avestruz consiste básicamente en no hacer nada, lo que puede llegar a funcionar. Si ocurre un bloqueo mutuo de procesos, los usuarios que iniciaron estos procesos, en algún momento se darán cuenta de que sus procesos no han terminado y, al ver esto, podrán terminarlos manualmente e iniciarlos nuevamente, lo cual puede solucionar el problema. De no ser así, la cantidad de procesos que quedarán bloqueados se irá incrementando hasta que el sistema se ponga extremadamente lento o incluso llegue a congelarse, en este caso, el usuario podrá apagar el computador o reiniciarlo nuevamente, con lo cual la situación podrá quedar resuelta.

2.7. Inanición o Starvation

La palabra inanición hace referencia a una debilidad por falta de alimentos o algún otro recurso. En el contexto de los sistemas operativos, este término se aplica a una situación en la que existe un proceso Pk que está en el sistema y que requiere de un recurso R1 particular para ejecutarse, pero que el sistema operativo no se lo asigna por lo que no puede continuar con su ejecución. Esto, por lo general, es debido a que existe un conjunto de otros procesos que constantemente se están ejecutando o que están llegando y que tienen una mayor prioridad sobre el uso del recurso R1. En esta situación, al proceso Pk se le mantienen en el sistema, pero no se le suministran los recursos que requiere, por lo que se dice que se muere por la falta de recursos, que no se le asignan debido a la carga de otros trabajos que está llegando al sistema.

La solución a este tipo de situaciones se ha planteado al incluir el concepto de envejecimiento, en el que cada vez que un recurso es negado a un proceso Pk, se le puede incrementar la prioridad un poco al proceso Pk, de forma tal, que en un tiempo razonable, el proceso Pk pueda tener una prioridad similar o incluso mayor a la de los procesos que se han estado adueñando de los recursos que él necesita. Cuando el proceso Pk llega a tener la prioridad suficiente, se le asigna el recurso que ha solicitado, aún por encima de los otros procesos, y entonces puede continuar su ejecución.

2.8. Procesos e Hilos

Como se ha descrito al comienzo de este capítulo, un proceso es un programa que está en algún estado de ejecución. Inicialmente, todo programa tenía asociado una entidad de proceso, lo que significaba que cada proceso tenía un segmento de datos, un segmento de código donde está el programa, la pila de ejecución donde se almacenan las llamadas a procedimientos y las variables locales, el *heap* para el manejo de variables y estructuras dinámicas y una serie de estructuras de datos como la PCB que almacena la información de la ejecución del proceso en el instante actual. Cuando un proceso debe dejar de ejecutarse y pasar al estado de *Listo*, debe ocurrir un cambio de contexto en el cual todos los datos que están en los registros de la CPU son almacenados en la PCB y en otras estructuras de datos, de forma tal, que el proceso, cuando vuelva a ejecutarse, pueda nuevamente cargar estos valores en los registros de la CPU para poder continuar justo en el mismo punto donde se quedó el proceso en el instante que este fue suspendido. Adicionalmente, hay estructuras como la tabla de archivos abiertos, espacios de memoria, entre otros, que están asociados a cada proceso y que también deben ser almacenados, salvaguardados o cambiados cada vez que ocurre un cambio de contexto.

Si se desea que dos procesos diferentes intercambien información, existe una serie de mecanismos como la comunicación a través de archivos, la definición de áreas de memoria compartida, así como el uso de *pipes* y *sockets*. Sin embargo, si los procesos pudieran compartir algunos elementos de su espacio de memoria, les sería más fácil interactuar entre sí para colaborar adecuadamente, pero durante la creación de un proceso se hace una copia de todos los espacios de memoria y estructuras de datos del proceso padre y esta copia es la que se asigna al proceso hijo para su ejecución, por lo que aunque en el proceso padre y el hijo comparten los mismos valores inicialmente, al estar cada uno trabajando en espacios de memoria diferentes, lo que uno de los procesos cambie no podrá ser visto por el otro, por esta razón, es que en los sistemas operativos surgió el concepto de Hilo de

Ejecución o *Threads*. Los *thread* o hilos son unidades internas a un proceso que comparten las áreas de texto, datos y *heap*, pero que poseen un ambiente de ejecución propio, es decir, que siguen una ruta de ejecución que puede ser diferente.

Los hilos también reciben el nombre de **procesos ligeros** ya que, al cambiar de un hilo a otro hilo en el mismo proceso, el cambio de contexto es mucho más ligero que al hacer un cambio de contexto entre dos procesos tradicionales. Bajo este enfoque, a los procesos tradicionales se les puede calificar de procesos pesados.

Un hilo o hebra, como algunos autores también los llaman, es una unidad de ejecución que comparte algunos elementos con otros hilos de ejecución; en particular, el segmento de texto, la tabla de archivos abiertos y algunos componentes de memoria. Sin embargo, cada hilo tiene la posibilidad de ejecutarse de forma independiente del resto de los hilos que conforman un mismo proceso gracias a que cada uno puede manejar su propio grupo de registros y su respectiva área de salvaguarda. En la figura 2.23 podemos apreciar un diagrama en el que se ilustran las principales características y diferencias entre los procesos y los hilos.

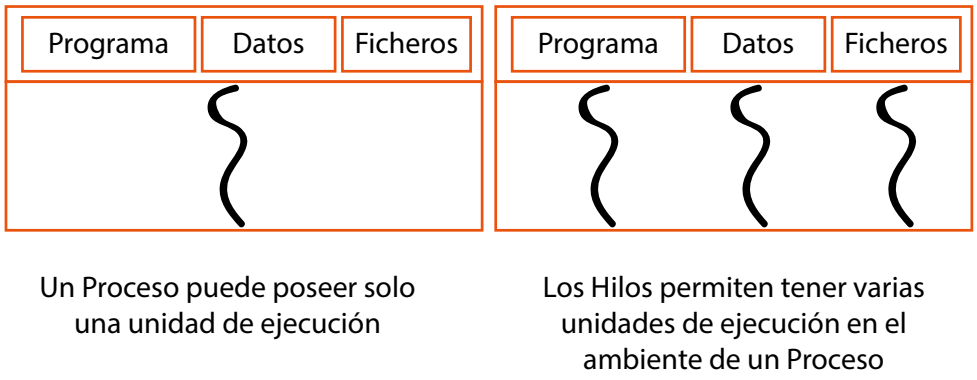


Figura 2. 23. Comparación de los elementos que componen un Procesos y un grupo de Hilos. Fuente: Adaptado de A. Tanenbaum (2009).

Los hilos son un mecanismo muy utilizado hoy en día, en especial cuando se desea hacer computación paralela, porque permiten una mejor utilización de las máquinas multiprocesadoras o con varios núcleos ya que brindan la posibilidad de ejecutar varios flujos de ejecución de un mismo programa de forma concurrente; esto permite que mientras un hilo se bloquea por alguna razón, otro de los hilos pueda continuar ejecutándose para así ir avanzando en la ejecución de toda la aplicación, aunque algunos hilos deban parar. Además, los hilos pueden intercambiar información a través de espacios de memoria compartida gracias a que comparten los recursos de un proceso; esto mismo permite reducir el tiempo que se emplea en realizar cambios de contexto entre hilos de un mismo proceso. Gracias a esto, se obtienen soluciones más rápidas y eficientes con la ejecución de varios hilos, en comparación con lo que se puede lograr haciendo una implementación similar, pero empleando solo el paradigma de procesos pesados.

2.8.1. Primitivas para la creación y manejo de Proceso e Hilos

Existe un conjunto de primitivas o llamadas al sistema para la creación, destrucción y asignación de trabajo a los procesos. En cada lenguaje el conjunto de llamadas disponible puede variar ligeramente, pero entre las que forman parte del lenguaje C, algunas de las que están disponibles son:

- **Fork**: permite la creación de un nuevo proceso hijo a partir de un proceso padre, que es quien originalmente invoca a la llamada fork. En este caso, ambos, padre e hijo, comparten áreas de memoria y ejecutan el mismo segmento de código.
- **Exec**: permite que un proceso pueda iniciar la ejecución de un nuevo segmento de código, diferente del que estaba ejecutando hasta el momento de ocurrir la llamada.
- **Wait**: permite que un proceso padre detenga su ejecución a la espera de que un hijo, que él ha creado mediante una llamada *Fork*, culmine su ejecución.
- **Exit**: permite que un programa hijo retorne un valor que puede ser tomado por su proceso padre para identificar cómo fue la ejecución que realizó el proceso hijo.
- **Kill**: permite enviar una señal a un proceso y, si esta señal no es atendida adecuadamente, puede ocasionar la terminación del proceso que recibe la señal. La llamada *kill* es una derivación de la llamada *signal*.
- **Signal**: es una llamada que permite definir un manejador para un tipo de señal específica, esto permite sobreescribir el tratamiento estándar que el sistema pueda dar a ciertas señales en caso de que esto sea necesario en algún proceso particular.

También existe un conjunto de llamadas funcionalmente similares, pero para los hilos, entre las que se encuentra:

- **pthread_create**: permite la creación de un hilo a partir del hilo que se está ejecutando. A diferencia de la llamada *fork*, la llamada *pthread_create* permite especificar un programa diferente al actual, como un parámetro de la llamada misma.
- **pthread_exit**: permite la terminación del hilo que la invoca, pero sin que esto genere que se termine el proceso que engloba a los hilos de la aplicación.
- **pthread_join**: permite que un hilo espere a que otro hilo termine su ejecución y le hace posible identificar cuál fue el status con el que concluyó el hilo finalizado.

Tema 3.
El administrador de Memoria

La memoria es el componente del ordenador que almacena y registra todos los valores y eventos que ocurren en el sistema en tiempo de ejecución. Además, todo proceso o hilo, antes de ejecutarse en la CPU, debe estar en memoria, ya que es de la memoria donde la CPU toma la información: instrucciones, variables, constante y estructuras de datos que usa para ejecutar los programas.

El componente del sistema operativo que se encarga de manejar la memoria del ordenador es el administrador de memoria o **gestor de memoria**, su tarea es la de suministrar espacios de memoria a los diferentes procesos e hilos que se estén ejecutando en el ordenador y, también, proteger el espacio que está siendo usado por un proceso para que este no sea invadido por otro de los procesos, ya que esto puede generar comportamientos inesperados en la ejecución de los procesos.

Para entender un poco algunas de las actividades que debe realizar el gestor de memoria, veamos primero cuál es el proceso de un programa desde que es escrito por un programador hasta que está listo para ejecutarse en la CPU. El programador escribe y arma un programa uniendo un programa principal escrito en un lenguaje de alto nivel con un conjunto de funciones que también ha escrito el programador o sus compañeros de trabajo, con algunas rutinas y librerías del sistema. Cada uno de

estos elementos es compilado, primero por separado y, luego, enlazado para generar un programa en lenguaje de máquina; una vez que tenemos esto, un elemento del sistema que recibe el nombre de **cargador** se encarga de solicitar espacio de memoria para almacenar allí el programa, junto con espacio para los datos que va a procesar. Este proceso se ilustra en la figura 3.1.

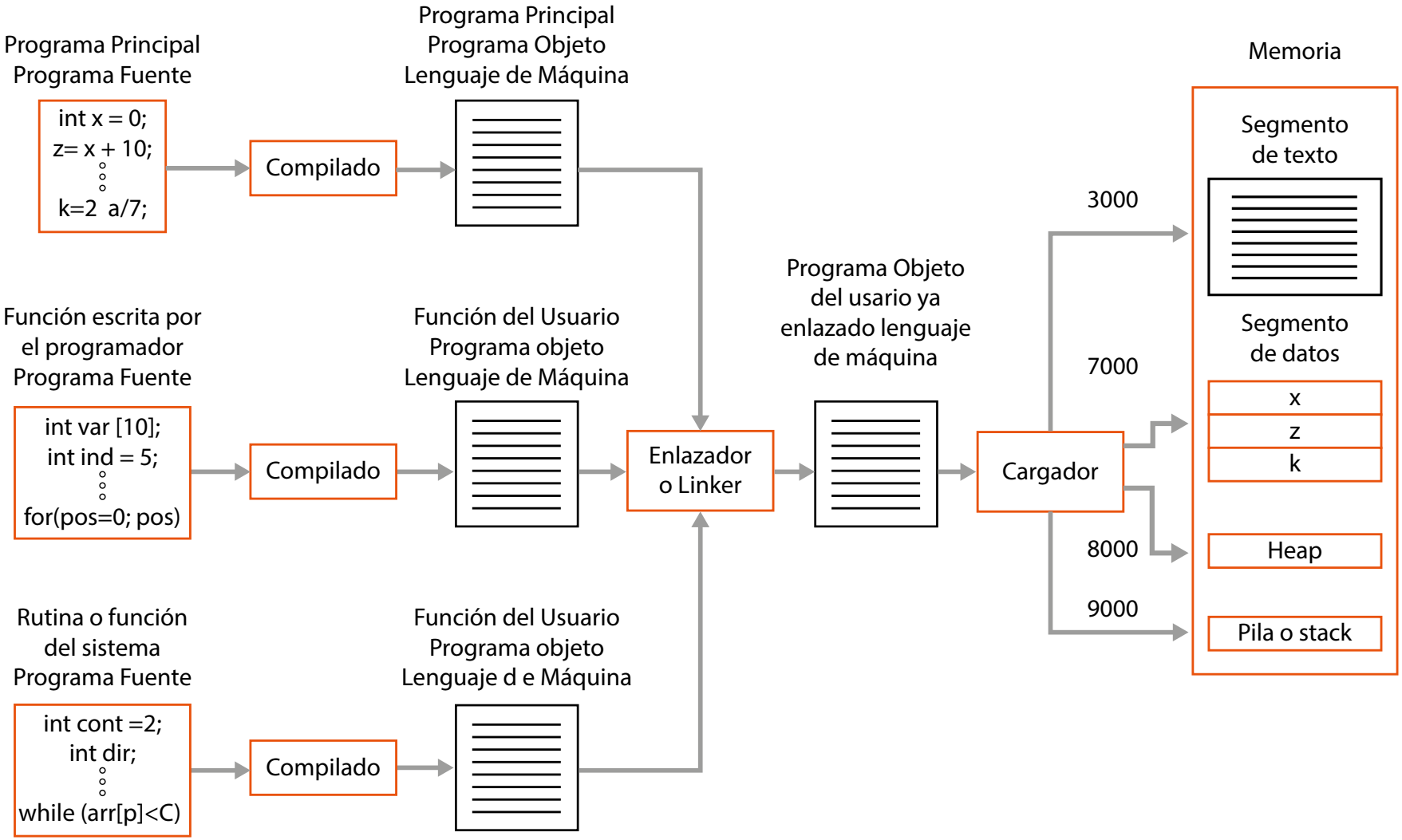


Figura 3. 1. Proceso desde que un programa es escrito por el programador hasta que está listo para ejecutarse. Fuente: elaboración propia.

La forma más sencilla de hacer esto es cuando, al compilar y enlazar el programa, se decide en qué posiciones podrá ser cargado el programa y estas permanecen fijas; entonces el cargador simplemente colocará el programa en esas posiciones de memoria y le indicará al despachador que está listo para ejecutarse. Luego, el despachador comenzará su ejecución según sea la política que se esté usando en el ordenador.

En el caso de que el cargador siempre cargue el programa en la misma posición de la memoria, se dice que el cargador es un cargador absoluto. Sin embargo, en un sistema operativo multiprogramado, al existir varios programas, no hay ninguna garantía de en qué orden pueden ser invocados y cargados los programas y esto puede hacer difícil garantizar que el mismo conjunto de direcciones de memoria, que deba ocupar un programa, no esté ya ocupado por otro programa que haya sido cargado antes.

Para resolver este inconveniente, los cargadores, desde ya hace algún tiempo, tienen la capacidad de colocar los programas del usuario en diferentes posiciones de memoria, a esto se le llama un **cargador relocalizable**. La forma de lograr esto puede ser relativamente sencilla; el cargador identifica la primera instrucción de máquina del programa y la etiqueta con la dirección 0, las siguientes posiciones de memoria las identifica con direcciones relativas a este comienzo del programa usando un desplazamiento, por ejemplo, la instrucción 10 del programa estará 10 palabras de memoria más abajo que el comienzo del programa, entonces estará en la posición relativa 10. Luego, cuando el cargador solicita un espacio al administrador de memoria para almacenar el programa, toma la dirección de comienzo de esa área y la carga en uno de los registros de la CPU. De allí en adelante todas las direcciones en el programa son calculadas sumando el contenido de un campo de desplazamiento, que indica lo lejos que está una instrucción o un dato a partir del comienzo del programa y el contenido de un registro que contiene justo la dirección de memoria donde comienza el programa, tal como se puede apreciar en la figura 3.2. Lo mismo puede ocurrir con las direcciones de datos, dentro del área de datos.

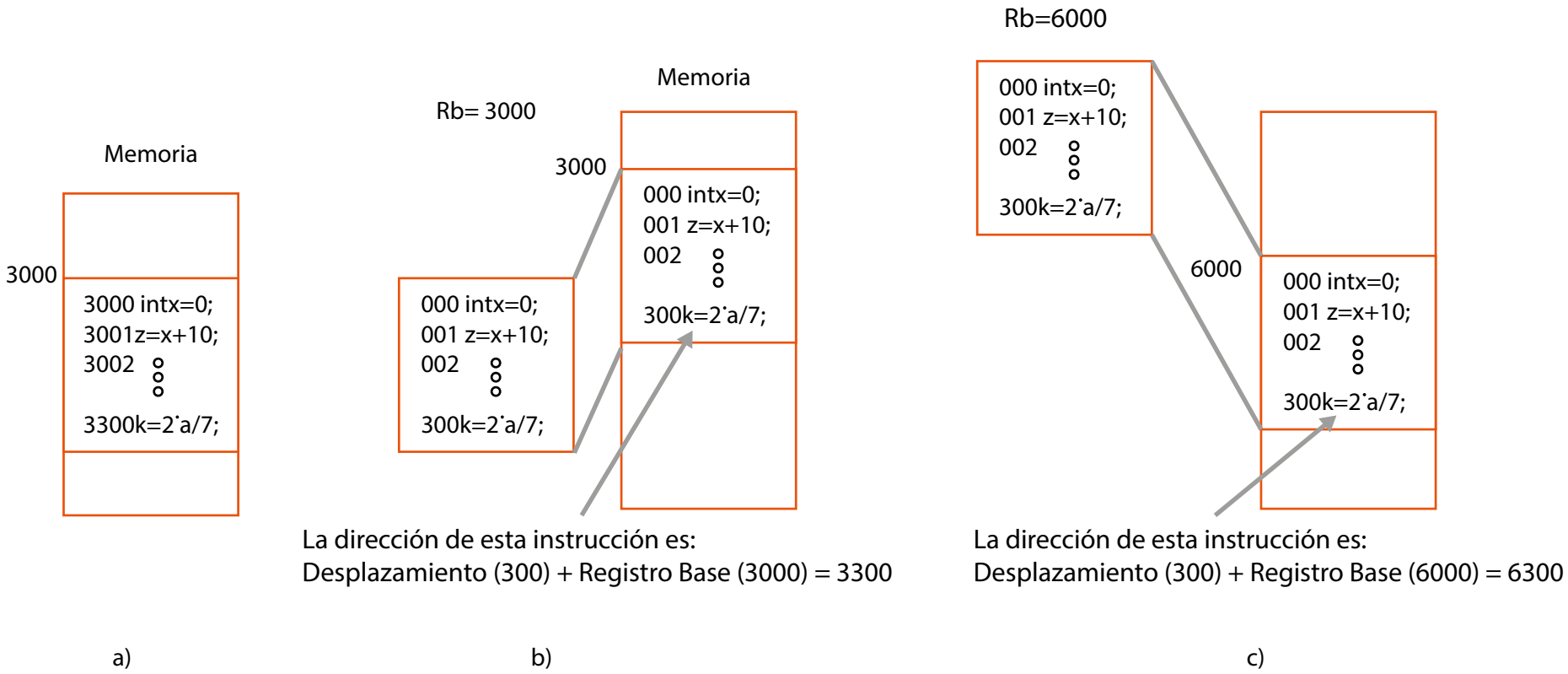


Figura 3. 2. Funcionamiento de un cargador absoluto (a) y de un cargador relocalizable (b) y (c) al cargar un programa en dos lugares diferentes de la memoria del ordenador. Fuente: elaboración propia.

3.1. Asignación de Memoria de forma Contigua

La forma en la que un programa solicita memoria y le es asignada, ha ido variando con el tiempo. El primer enfoque consistía en dividir la memoria en espacios iguales de memoria donde a cada proceso se le asigna uno de estos espacios; este es un método simple de implementar y mantener, pero con el inconveniente de que los procesos no ocupan siempre la misma cantidad de espacio en la memoria, por lo que un espacio fijo puede hacer que algunos procesos muy grandes no quepan en el espacio asignado. Para solventar esto se pueden usar bloques grandes de memoria, pero si los bloques son grandes, la memoria quedará dividida en solo algunos pocos bloques grandes; al hacer eso estaremos limitando la cantidad de procesos concurrentes en un sistema a solo unos pocos. Al usar bloques grandes, lo otro que puede pasar es que, si hay procesos pequeños, se desperdiciará una gran cantidad de espacio, tal como se puede apreciar en la figura 3.3.

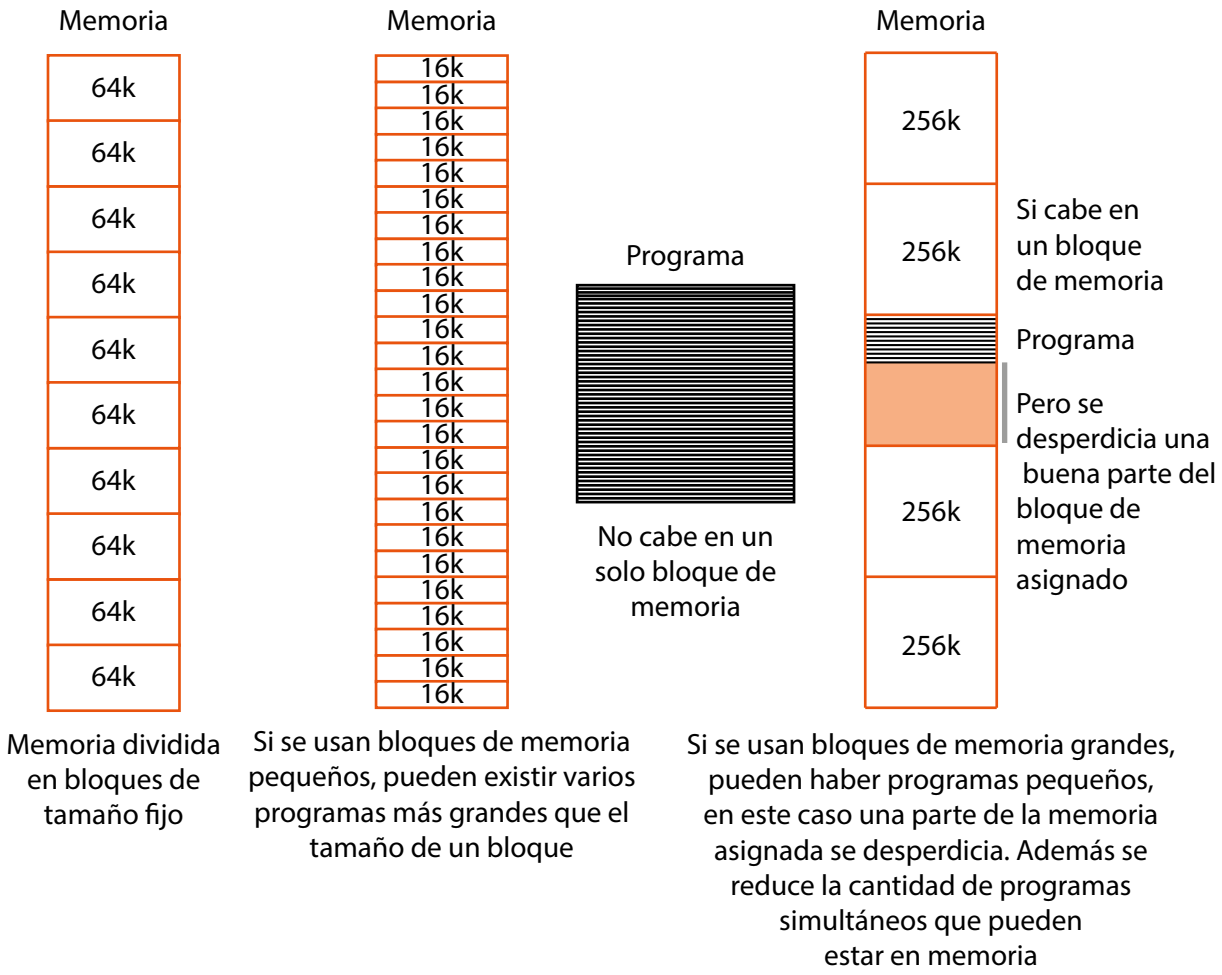


Figura 3. 3. Asignación estática de un bloque de memoria de tamaño uniforme para cada proceso del sistema. Fuente: elaboración propia.

La siguiente opción es asignar la memoria en espacios no uniformes. En este caso, cada proceso solicita el espacio que va a requerir para ejecutarse de un todo y como una sola unidad. Este es un enfoque relativamente simple, pero también menos flexible, ya que si hay varios procesos o hilos que comparten el mismo programa, no será posible compartir el programa completo sin compartir los datos y otros espacios, ya que estos son manejados como un solo bloque en la memoria.

3.1.1. Segmentación

Para permitir compartir algunos espacios, pero sin compartirlo todo, se creó la asignación de memoria por segmentos en la que se asignan espacios de memoria no uniformes para el segmento de texto que contiene solo las instrucciones del programa, otro segmento para los datos y otros segmentos adicionales para la pila de ejecuciones y otras estructuras que registren la evolución en la ejecución del proceso.

Al manejar cada segmento por separado, es posible que dos procesos o hilos compartan el segmento de texto, pero que cada uno tenga su propio segmento de ejecución para que realmente puedan trabajar de forma independiente uno del otro, como se puede apreciar en la figura 3.4. Al compartir un mismo segmento, no solo es posible ahorrar espacio de la memoria, sino que también se facilita el intercambio de información entre los procesos, ya que un proceso A puede dejarle datos o resultados a otro proceso B en una de estas áreas compartidas.

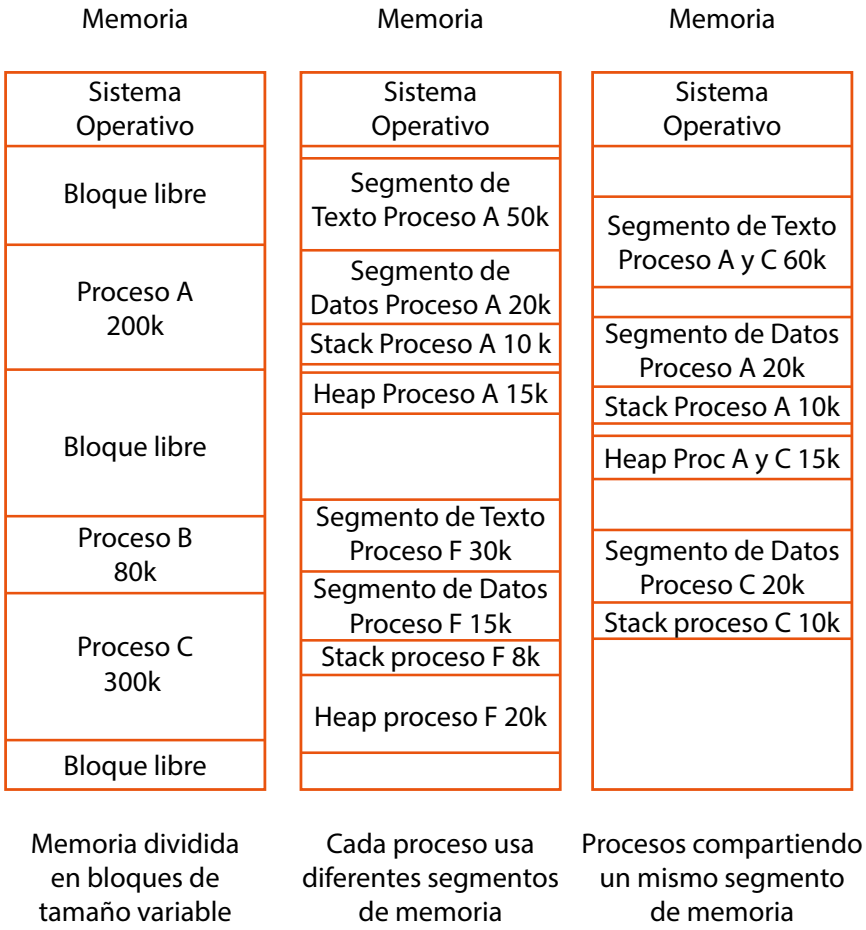


Figura 3. 4. Uso de segmentos de tamaño variable en la asignación de memoria a los procesos en el ordenador. Fuente: elaboración propia.

En los esquemas que hemos visto de asignación contigua de memoria, se pueden presentar algunos inconvenientes. A medida que procesos se van ejecutando, se van reservando espacios en la memoria, pero si algunos de los procesos terminan, los espacios de memoria que tenían esos

procesos reservados ya no van a ser usados y, por tanto, se les puede liberar. Esto puede ocasionar que, aunque se vayan asignando los espacios de forma secuencial, termine ocurriendo que los procesos que van terminando vayan generando huecos o espacios disponibles en el medio de la memoria. Si estos espacios son pequeños, puede ser difícil que vuelvan a ser utilizados debido a que los nuevos procesos que van llegando pueden ser más grandes. A este fenómeno se le llama **fragmentación externa**, ya que los espacios libres que se van generando son externos a las unidades de memoria que están siendo reservadas. Esta situación es la que se pretende describir en la figura 3.5.

Cuando se solicita un espacio de memoria en el ordenador, se pueden seguir tres técnicas diferentes para asignar el espacio de memoria. La técnica del primer ajuste lo que hace es buscar, en la lista de espacios disponibles, el primer espacio disponible donde el nuevo elemento pueda caber y allí hace la asignación. La siguiente técnica del peor ajuste, lo que hace es buscar el espacio más grande y, allí mismo, hace la asignación. Al buscar un espacio más grande, lo que se está buscando es que, aun después de hacer la nueva asignación, el espacio restante que quede disponible en ese mismo bloque sea lo suficientemente grande como para que quepa algún otro proceso que pueda estar luego buscando un espacio de memoria. Finalmente, la técnica del mejor ajuste lo que hace es buscar un espacio donde quepa la nueva solicitud de espacio, pero donde el espacio que se desperdicie después de hacer la asignación sea el menor posible; en otras palabras, se busca el espacio disponible que sea lo más parecido al espacio que se está buscando para que así el desperdicio de memoria debido al fenómeno de fragmentación externa sea lo más pequeño que se pueda.

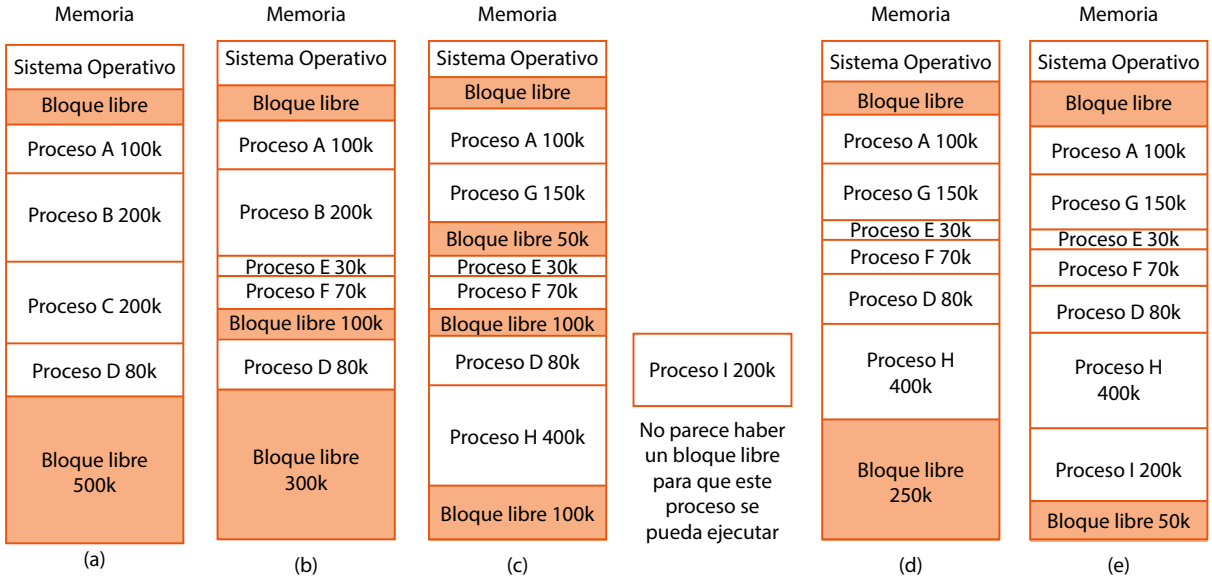


Figura 3. 5. Proceso de asignación de espacios de memoria en el que ocurre fragmentación externa. Fuente: elaboración propia.

En esta figura podemos ver cómo en el paso (a) han sido asignados espacios para los procesos A, B, C y D. Luego, en el instante (b), ha finalizado el proceso C y han llegado los procesos E y F que son ubicados en los primeros espacios disponibles que se encuentran justo donde estaba antes el proceso C, que ya ha terminado. En el instante (c), el proceso B ha terminado y liberado el espacio de memoria que ocupaba, pero también llegaron los proceso G y luego el H. En este punto, llega el proceso I,

pero no hay ningún espacio contiguo de memoria que pueda alojarlo. Sin embargo, si sumamos los espacios libres que hay de 50k 100k y 100k, habría 250k libres donde sí podría tener espacio el proceso I; esto es lo que se hace en el instante (d) en el que los procesos son relocalizados para que los espacios libres disponibles puedan ser juntados y, de esta forma, se tenga lugar para el proceso I y aún resten 50k de espacio libre en la memoria. Al proceso que permite reubicar los espacios usados de memoria para que los espacios libres queden juntos y poder mejorar así la asignación de memoria, se le llama **compactación**.

3.2. Asignación de Memoria no Contigua

Para aprovechar los espacios grandes o pequeños que puedan presentarse en la memoria, se buscó una técnica que permitiera aprovechar tanto los espacios grandes y continuos como también los espacios pequeños, no contiguos, producto de la fragmentación externa.

3.2.1. Paginación

Una de las opciones para reducir los problemas inherentes a la fragmentación interna y externa se basa en usar bloques pequeños, de tamaño fijo, para los procesos, pero de forma tal que un proceso pueda usar un espacio suficiente al contar varios bloques pequeños de tamaño fijo, no necesariamente consecutivos, para almacenar su contenido.

En la figura 3.6 se puede apreciar cómo puede ser el uso de bloques de tamaño fijo para almacenar la información de un proceso, empleando bloques de memoria del mismo tamaño en el ordenador. A cada uno de estos bloques pequeños se le ha dado el nombre de **Página**.

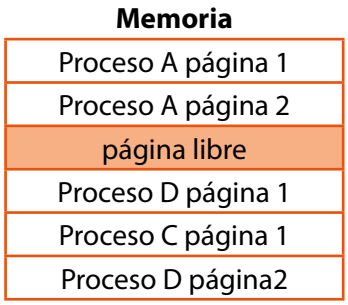


Figura 3. 6. Asignación de bloques de memoria mediante Páginas. Fuente: elaboración propia.

Ahora, si un proceso posee su espacio de direcciones contenido en un conjunto de páginas no contiguas, ¿cómo un programa puede ejecutarse y pedir la siguiente instrucción si está en una página varios espacios más adelante? La solución a este inconveniente se basa en emplear un esquema de traducción de direcciones, donde se emplean dos conjuntos de direcciones que son manejados a través de una o varias tablas. En estas tablas se manejan direcciones lógicas en las que las direcciones de las instrucciones y datos se manejan como si estuvieran en direcciones contiguas de memoria, y también se maneja otro conjunto de direcciones que llamaremos físicas, en las que se manejan los lugares donde realmente están estas direcciones en la memoria, esto se hace para las páginas que almacenan los programas y los datos.

El tamaño de las páginas en un ordenador puede oscilar entre los 4k y los 64k bytes de memoria. Si suponemos que un sistema posee páginas de 4k bytes, es decir, 4096 bytes de longitud, en este caso, los 12 bits menos significativos de la dirección se usan para generar un desplazamiento sobre la página, mientras que los 20 bits más significativos se usan para diferenciar un marco de página dentro de todos los marcos que están en la memoria. En la figura 3.7 se ilustra cómo puede ser la traducción de las direcciones lógicas a las direcciones físicas para conseguir adecuadamente la información que se está buscando en la memoria física del ordenador.

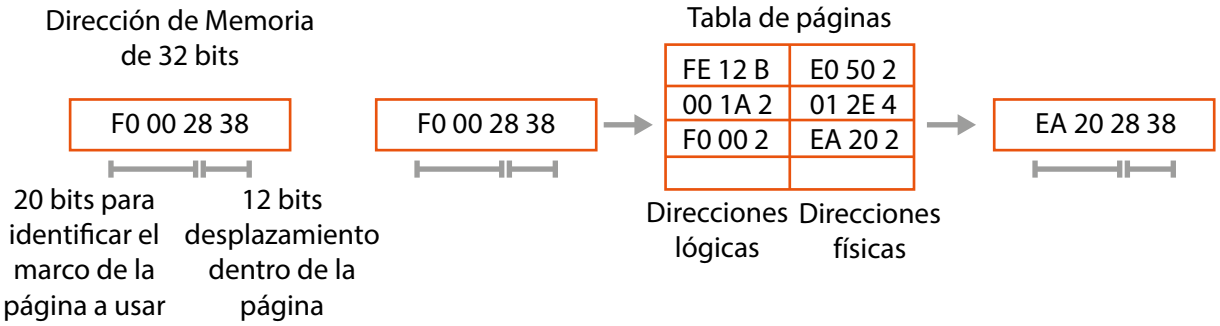


Figura 3. 7. Esquema de traducción de memoria lógica a memoria física para ubicar la información que se desea en la página correcta de la memoria RAM del ordenador asumiendo que las páginas son de 4k. Fuente: elaboración propia.

Dado que la traducción de direcciones lógicas a físicas debe ser una traducción muy rápida y eficiente ya que se realiza prácticamente en cada instrucción, en los ordenadores actuales se utiliza una tabla especial que, con ayuda del hardware y empleando memorias asociativas, permite agilizar la búsqueda y comparación de información. Esta tabla se identifica por las siglas *TLB* que corresponden a *Translation Look-Aside Buffer*.

En una memoria asociativa se ingresa qué patrón se está buscando y esta entrada se compara de forma simultánea con todas las entradas de la tabla; solo aquella entrada que coincida con el patrón de bits que está siendo buscado, es la que va a responder el proceso de traducción a la memoria real del ordenador donde se encuentra almacenado el dato o la instrucción que estamos buscando. Esta es la situación que se ilustra en la figura 3.8. De esta forma, es posible acelerar de forma importante la búsqueda de direcciones dentro del ordenador.

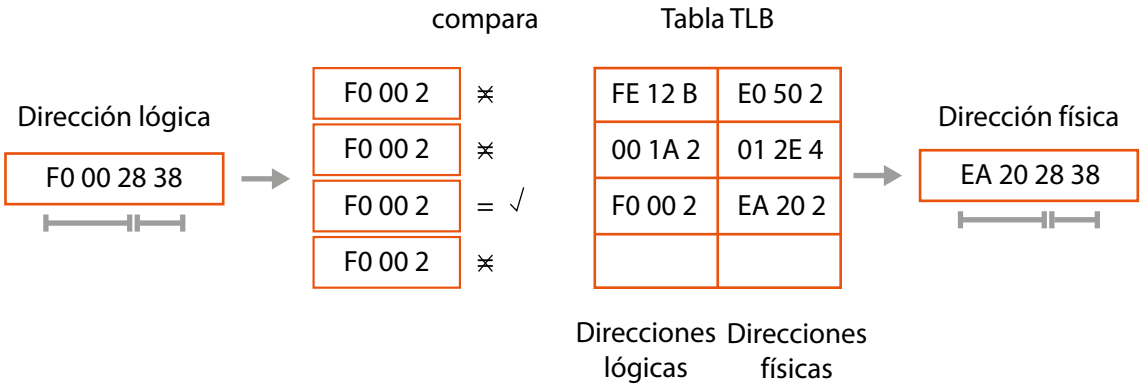


Figura 3. 8. Traducción de una dirección lógica a una dirección física en el ordenador. Fuente: elaboración propia.

Cada proceso tiene su propia tabla de páginas y también puede haber una tabla global. Dado que esta tabla es fundamental para la operación de los programas, a la hora de hacer un cambio de contexto también se debe salvaguardar el estado de la tabla de páginas del proceso actual.

Para una mejor administración de las páginas, las entradas en la tabla de página pueden contener una serie de bits con significados especiales, además de las direcciones lógicas y físicas de las páginas, tal como se puede apreciar en la figura 3.9.

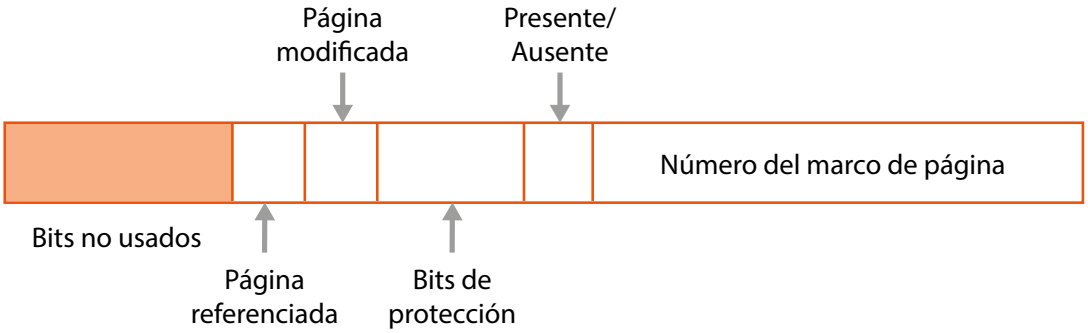


Figura 3. 9. Información que se puede almacenar en la tabla de páginas. Fuente: elaboración propia.

Entre los campos que se muestran allí están:

- **Bits de página referenciada**: este valor se coloca en 1 cada vez que se hace referencia a cualquier dirección de la página. Es utilizado por el sistema operativo para saber si la página ha sido usada ya que una página que ha sido colocada en memoria, pero que no ha sido referenciada aún, de ser sacada, volverá a generar un fallo de página. Esta información puede ser usada por los algoritmos de sustitución de páginas para elegir una página que deba ser sustituida.

- **Bits de página modificada**: si el programa ha realizado algún cambio a algún contenido de la página, saberlo es muy útil porque, si la página no ha sufrido cambios, simplemente se puede descartar, pero si el programa ha cambiado algo en ella, la página debe ser almacenada en la memoria virtual antes de ser descartada.

- **Bits de protección**: permiten especificar los permisos que tendrá el proceso respecto a la información que contiene la página. Los permisos pueden identificar que la pagina es de solo lectura, de escritura o de lectura y escritura.

- **Bit de Presencia o ausencia**: indica si la página está presente en memoria principal (RAM) o si, por el contrario, se encuentra solo en la memoria virtual.

- **Número del marco de página**: si la página se encuentra en memoria principal (RAM), este campo indica la dirección física en donde está ese marco de página almacenado.

3.2.2. Segmentación Paginada

Para hacer un uso más eficiente de la memoria se han unido las ventajas de las técnicas de **segmentación** y **paginación** en un mismo enfoque que ha recibido el nombre de **segmentación**

paginada. En este enfoque, el espacio de un proceso es dividido en un conjunto de espacios no uniformes que reciben el nombre de segmentos, en cada uno de los cuales está una porción de los datos que requiere el proceso para ejecutarse.

El proceso maneja entonces una tabla de segmentos en la que se registra donde está ubicado cada uno de los segmentos que usa el proceso. Esta tabla permite que un mismo proceso pueda tener sus segmentos de texto, datos y ejecución en posiciones de memoria diferentes que no tienen por qué estar consecutivas para que el proceso pueda ejecutarse. Pero, luego, cada segmento es, a su vez dividido, en un conjunto de páginas uniformes. Con esto, cada proceso tendrá una tabla de segmentos y, por cada entrada en la tabla de segmentos, existirá una tabla de páginas que registrará dónde están cada una de las páginas de ese segmento en especial. Incluso es posible tener un esquema aun más complejo en el que exista un esquema jerárquico, con varias tablas de páginas, para poder referenciar un área mucho más extensa o con más páginas en la memoria.

3.3. Memoria Virtual

La memoria virtual es una técnica de manejo de la memoria que permite que un computador pueda usar más memoria principal de la que posee realmente. La estrategia que se usa es bastante simple; el espacio completo de direcciones de la memoria, el cual es llamado memoria virtual, es almacenado en un dispositivo de almacenamiento secundario como puede ser un disco duro. Esa imagen de la memoria es dividida al menos en páginas, aunque también se puede usar el esquema de segmentación paginada y lo que se hace es colocar en la memoria RAM solo un conjunto, que puede ser no muy grande, de las páginas que hay en la memoria virtual a medida que los programas se van ejecutado. Si algunas páginas dejan de usarse, entonces estas pueden ir siendo sustituidas por nuevas páginas del mismo proceso o de otros procesos. Esta situación que describimos aquí se ilustra en la figura 3.10.

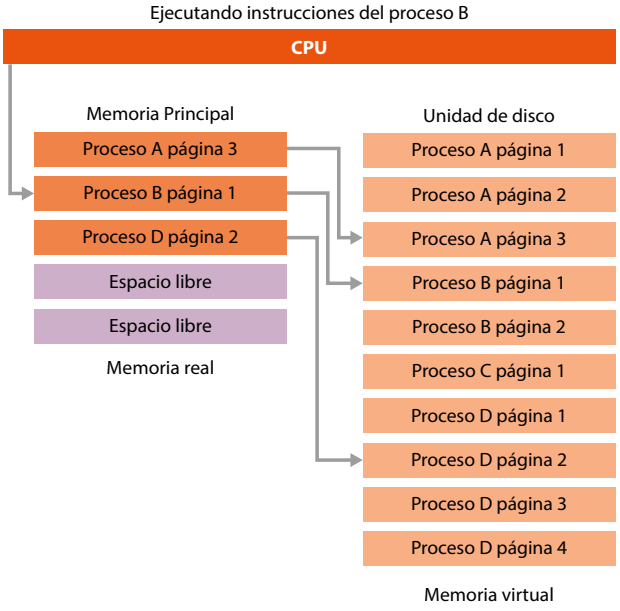


Figura 3. 10. Esquema general de operación de la memoria Virtual de un ordenador. Fuente: elaboración propia.

En este esquema podemos apreciar que, en la memoria principal del ordenador, están cargadas solo 3 páginas, una del proceso A, otra del B y una del Proceso D. Estos procesos realmente poseen 3, 2 y 4 páginas respectivamente, pero realmente no hace falta que estén todas las páginas de un proceso todo el tiempo en memoria para que puedan ejecutarse, con solo algunas de ellas puede ser suficiente.

3.3.1. Fallos de Página

Cuando un proceso requiere de información que está en la memoria virtual, es decir, en el disco duro, pero esta no está en la memoria real, ocurre lo que se llama un fallo de página. En este proceso el sistema operativo se da cuenta de que la dirección de los datos o programas que está buscando no está en la memoria real y pasa a buscar dónde está la información en la memoria virtual; para ello, mira la tabla de páginas y verifica si el bit de presente está en 0, indicando que la página no está en la memoria principal, pero sí en la memoria virtual, se toma la dirección en la memoria virtual y se busca el contenido de la página. Una vez que se ha conseguido la página en la memoria virtual, se verifica si hay espacio disponible para esa página en la memoria real; si la respuesta es positiva, entonces la página pasa de la memoria virtual a la memoria real, donde puede ser tomada por la CPU para usar su información durante la ejecución del proceso correspondiente y actualiza entonces la tabla de páginas para indicar que la página ahora si está presente en la memoria principal. Si en la memoria principal no hay espacio disponible para cargar la nueva página, entonces se debe elegir una víctima para sacar al menos una de sus páginas, se selecciona entonces una página de esa víctima, se descarta dicha página y, en su lugar, se coloca la página nueva.

3.3.2. Algoritmos de Selección de Páginas

Para seleccionar cuál es la página que debe salir de la memoria principal, se toma en consideración el principio de localidad temporal, en el que se considera que, si una posición de memoria ha sido usada recientemente, hay una alta probabilidad de que la misma posición de memoria sea usada en el futuro cercano. Esta estrategia se basa en lo común que es el uso de ciclos en la programación y, si una instrucción o un dato es usado en un ciclo, cuando se vuelva a ejecutar el ciclo se volverá a acceder a la misma información. En función de este y otros principios, ha sido diseñada una serie de mecanismos que han sido implementados mediante algoritmos para sustituir páginas de la memoria principal, que a continuación pasamos a describir:

- **PEPS o FCFS**: en este mecanismo se busca la página que tenga más tiempo en la memoria, y esta es la página que se toma como candidata a ser sustituida.

- **MRU o LRU**: menos recientemente usada, o *Least Recently Used*, es un mecanismo en el que se elige como página víctima a ser sustituida, aquella que desde hace más tiempo no ha sido tocada o accedida en el ordenador, asumiendo que, si hace tiempo que no se ha usado, es poco probable que vuelva a ser usada en el futuro cercano.

- **Segunda oportunidad**: en este mecanismo, se toma la página que tenga más tiempo en la memoria y se verifica si ha sido accedida recientemente, si no ha sido así, se sustituye esa página, pero si su bit de referencia está en 1; en ese caso, se coloca el bit de referencia en 0 y

se coloca como última en la lista de páginas a ser sustituidas; la próxima vez que sea seleccionada, entonces sí saldrá de la memoria principal. Esto le da una segunda oportunidad si la página ha sido empleada recientemente.

- **Envejecimiento**: en este mecanismo se hace una modificación del mecanismo de *MRU* o *LRU*, en el cual se usa serie de bits como un esquema de prioridad para saber si la página debe salir de la memoria principal. En este esquema, si una página es referenciada, entonces se coloca en uno el bit de más a la izquierda, es decir, el más representativo; por ejemplo, 10000000, y después de un periodo de tiempo, se divide todo el número entre dos o se hace un corrimiento o *shift* a la derecha, de forma que la prioridad queda como 01000000. Pero si vuele a ser accedida entonces, la prioridad queda en 11000000, de esta forma, si en un lapso de tiempo el patrón de accesos de una página es 01010110, entonces la prioridad para esa página queda con el valor 001101010, después de un lapso de tiempo. Como se puede apreciar, una página que ha sido referenciada recientemente tiene un valor de prioridad alto y, a medida que va pasando el tiempo, ese nivel de prioridad va bajando al ser dividida la prioridad entre 2. Por esto, si se ha de sustituir la página para que no permanezca en la memoria principal, se puede elegir la página que tenga una menor prioridad, ya que esto significará que las otras páginas han sido accedidas más recientemente o con más frecuencia y, por ello, tienen una mayor probabilidad de ser referenciadas en el futuro.

Estos y otros posibles mecanismos asociados a la memoria virtual, lo que hacen es que parezca para el usuario que todas las páginas están en la memoria cuando realmente solo hay un subconjunto de ellas en un instante de tiempo dado, pero cuando nuevas páginas son solicitadas, si estas no caben en la memoria principal, entonces se sacan algunas de las páginas que están actualmente para dar cabida a las nuevas.

En esto consiste la Memoria Virtual en los ordenadores. Este es un mecanismo muy poderoso porque permite trabajar con más recursos de los que realmente se disponen en la memoria principal del ordenador, gracias al hecho de poder almacenar en un espacio en el disco duro toda la imagen o todas las páginas de la memoria virtual. Esto también permite trabajar con más procesos de forma concurrente en el sistema, hacer un mejor uso de la CPU y permitir que se ejecute una mayor cantidad de trabajo por unidad de tiempo.

Al área del disco donde son almacenadas las páginas de la memoria virtual se le suele llamar **área de intercambio** o **área de *Swap*** y en algunos sistemas operativos se suele colocar en una partición aparte para garantizar que esta misma área no pueda ser accedida por otros programas sin los permisos adecuados.

3.3.3. Problemas que pueden ocurrir en un ambiente de Memoria Virtual

Aunque los beneficios de contar con un ambiente de memoria virtual en el que se puedan emplear más recursos físicos de los que están realmente disponibles son bastantes, los mecanismos de memoria virtual también pueden ocasionar que el comportamiento del ordenador no sea el esperado en algunos casos.

El primero de los problemas es que en un esquema de memoria virtual, en cualquier momento un programa invocará a una instrucción o a un dato que no esté en la memoria principal, en este caso ocurre lo que se llama una falla de página; recordemos que todas la paginas están en la memoria virtual que suele estar en un disco duro, entonces el sistema, al darse cuenta que la información requerida no está en la memoria principal, debe iniciar una operación de Entrada/Salida para recuperar la página que contiene la información solicitada.

Una operación de entrada salida toma muchísimo más tiempo que acceder a un dato que está en la memoria principal, con lo cual el programa que se está ejecutando debe pararse y esperar a que la página nueva sea cargada en la memoria. Esto hace que la ejecución de ese programa en particular tome mucho más tiempo del que le tomaría si pudiera estar totalmente cargado en la memoria principal del ordenador. Aunque esto claramente perjudica la velocidad de ejecución de ese programa en particular, no debemos olvidar que, al tener más programas en el sistema, gracias a la memoria virtual, también podremos tener a la CPU ocupada con otros procesos, aunque algunos programas interrumpan su ejecución.

Sin embargo, si la cantidad de programas es grande y cada uno de ellos usa muchas páginas durante su ejecución, la cantidad de fallas de páginas que puede ocurrir en el sistema se puede incrementar de forma importante y llevarnos a una situación que recibe el nombre de **Hiperpaginación** o *thrashing*. Esta es una condición en la que hay muchos procesos en el sistema y, para traer nuevas páginas, hace falta seleccionar algunas páginas actuales en la memoria principal, para que estas sean sustituidas por nuevas páginas. Si las paginas elegidas están aún en uso, el programa volverá a solicitarlas y como fueron sacadas, esto podrá generar nuevos fallos de páginas; si esta situación se mantiene, la cantidad de fallas de página en el sistema se puede incrementar de forma importante haciendo que el sistema se torne lento e incluso llegue a dejar de responder algunos instantes por estar continuamente seleccionando páginas a ser sustituidas y generando operaciones de entrada y salida para traerlas de vuelta a la memoria principal.

Tema 4.
Gestión de la Entrada/Salida y Ficheros

Los principales componentes de hardware de un ordenador son la **Memoria**, que es el elemento en el que son almacenados los datos y programas para ser procesados, la **CPU o unidad central de proceso**, que es el componente que hace el procesamiento de los datos que están en la memoria y, finalmente, los **dispositivos de Entrada/Salida**.

Los procesos de entrada y salida hacen referencia a cualquier transferencia de información que ocurre desde o hacia la memoria o el procesador, incluyendo la transferencia, tanto de datos como de programas, y que involucra el uso de dispositivos periféricos. Es por ello que estos procesos pueden ser empleados, tanto para introducir programas y datos en la memoria, como para mostrar los resultados del procesamiento de los programas y datos a los usuarios para que estos puedan entenderlos y usarlos adecuadamente.

4.1. Dispositivos de Entrada/Salida

A los dispositivos de Entrada/Salida también se les conoce como dispositivos periféricos, ya que están en la periferia del ordenador y permiten comunicar el centro de ordenador que está conformado por la CPU y la memoria con el exterior del equipo, donde pueden existir otros componentes y donde también residen los usuarios.

Cada uno de los dispositivos de Entrada/Salida tienen formas y tecnologías específicas que pueden ser muy diferentes, desde un teclado que usa componentes mecánicos y electrónicos cada vez que presionamos una tecla, hasta dispositivos mecánicos y ópticos en los *CDROM*, *DVD* y *Blue Ray* que con un láser leen los surcos que hay en el disco, pasando por los componentes electrónicos y magnéticos de los discos duros, donde se guardan bloques de datos de forma no secuencial y se cuenta con un índice para conseguir de forma más rápida y eficiente los datos buscados.

Poder trabajar con dispositivos tan diferentes puede implicar que los usuarios deban aprender diferentes formas de comunicarse con cada uno de los dispositivos que puedan existir, dependiendo de las tecnologías que estos empleen. Sin embargo, desde el punto de vista del usuario, lo que él desea hacer es colocar o extraer información hacia y desde el ordenador; si esto se puede hacer de una manera uniforme, independiente, sencilla, fácil de utilizar y empleando siempre más o menos los mismos comandos o llamadas, será mucho más fácil para el usuario y el programador trabajar con todos estos periféricos diferentes en el ordenador.

Es por esto que servir de interfaz entre los usuarios y los dispositivos periféricos es una parte de las tareas que hace el gestor o manejador de Entrada/Salida en el sistema operativo, ofreciéndole a los usuarios un conjunto de llamadas comunes que pueda invocar para manejar las diferentes tecnologías y formas de operación que pueda tener cada uno de los dispositivos periféricos existentes.

Adicionalmente, para hacer más fácil el trabajo de los programadores, es preferible ocultar algunos de los detalles de operación y manejo de estos dispositivos, de manera que no tengan que estar en los programas de los usuarios todos los detalles de operación de cada tipo de dispositivo. Imagine que deba recordar cada vez que requiera un dato del disco duro, a qué velocidad debe rotar el plato del disco y cuánto tiempo se debe esperar desde el inicio del disco para que la información que usted requiere pase debajo de la cabeza lectora/escritora, para que allí usted pueda hacer la lectura del dato que busca. De tener que recordar todos estos detalles, nos lo pensaríamos cada vez que necesitáramos hacer una operación de lectura del disco, por lo complicado que sería. Debido a esto, será entonces el software del sistema operativo el encargado de lograr y ofrecer al usuario esta independencia de la tecnología de operación del dispositivo a través de un conjunto de llamadas al sistema relacionadas a la E/S, que utilizarán los programas y que serán independientes del tipo, modelo y tecnología del periférico. Con esto, colaboran los fabricantes de los equipos ofreciendo controladores o *drivers* que le permitan al sistema operativo identificar y gestionar las particularidades de cada dispositivo y operar con él de forma adecuada.

Los dispositivos de Entrada/Salida se pueden clasificar en tres grandes grupos: aquellos que permiten solo la entrada de datos, como es el caso del teclado y el ratón; dispositivos de solo salida, como puede ser el caso de la impresora y quizás la pantalla del ordenador, aunque actualmente las pantallas táctiles pueden ser consideradas como un dispositivo de entrada y de salida y; finalmente, los dispositivos de almacenamiento secundario que pueden ser considerados como dispositivos de salida cuando almacenan datos y, de entrada, cuando vuelven a leer datos que previamente han almacenado.

Aunque el trabajo de la memoria principal es el de almacenar los datos que debe usar el ordenador, por el tipo de tecnología que se usa en la construcción de la memoria principal, esta es volátil, es decir, que puede operar perfectamente mientras el ordenador esté encendido pero, una vez que el ordenador

se apaga, la memoria principal deja de operar y se borra la información que esté contenida en ella. De esta forma, la función principal de la memoria RAM que es el almacenamiento de datos y programas, se deja de cumplir ya que, para su funcionamiento, la memoria RAM requiere de estar alimentada eléctricamente todo el tiempo, pero la realidad es que cuando apagamos un ordenador, los datos que este contiene no se pierden. Esto ocurre porque muchos de los datos son almacenados en medios de almacenamiento secundario para evitar que se pierdan y que, de esta forma, estén disponibles la próxima vez que encendamos el ordenador. Los principales dispositivos de almacenamiento secundario de hoy en día son: los discos duros, los discos de estado sólido y las memorias externas.

4.2. Gestión de la Entrada/Salida

El componente del sistema operativo que se encarga de manejar los procesos de Entrada/Salida debe brindar a los programadores y procesos los siguientes elementos:

- **Comandos que faciliten el uso de los periféricos**: estos deben ser un conjunto de comandos y llamadas al sistema que los usuarios pueden invocar y que faciliten el uso de los dispositivos periféricos, haciendo más uniforme el uso de estos dispositivos, independientemente de las tecnologías que estos empleen para realizar sus funciones.

- **Llamadas al sistema operativo para tareas de E/S**: para que los programadores puedan, desde sus programas, solicitar los servicios del sistema operativo para que este, a través de un conjunto de llamas al sistema, pueda realizar y gestionar todas las operaciones que requieran los dispositivos para publicar, almacenar o extraer información de una forma más o menos uniforme, al menos desde el punto de vista del usuario.

- **Gestión de interrupciones**: la mayoría de los dispositivos de entrada/salida son relativamente lentos en comparación con la CPU, es por esto que cuando se desea solicitar una información, en lugar de dejar que la CPU la realice directamente, se le pide al dispositivo que la inicie y se pasa a la CPU a hacer otra actividad, con el fin de poder aprovechar este tiempo en la ejecución de otros procesos. Cuando la operación está lista, el dispositivo debe avisar a la CPU que los datos solicitados ya están disponibles, de esta forma, cada dispositivo puede ir trabajando asíncronamente de forma separada de la CPU, mientras tanto la CPU puede ir avanzando con la ejecución de otros procesos. De esta manera, se pueden mantener operativos tanto la CPU, como los dispositivos de forma independiente, sin que uno deba esperar al otro.

- **Gestión de errores**: el gestor de dispositivos debe poder manejar cualquier tipo de situación en la que los dispositivos no estén en la capacidad de suministrar los servicios que el usuario les requiere y traducir estas situaciones a un lenguaje que los usuarios puedan manejar para poder atender adecuadamente estas situaciones. Las condiciones de error deben contemplar desde esquemas de operación no adecuados del dispositivo, hasta el hecho de que la información que esté siendo solicitada no esté realmente disponible en el dispositivo que está siendo consultado.

4.2.1. Estrategias para la gestión de la Entrada/Salida

El sistema operativo puede gestionar la operación de los dispositivos de Entrada/Salida de tres formas diferentes:

- **Gestión de E/S por muestreo**: este tipo de gestión es similar al mecanismo de *Polling* que hemos descrito antes. En él, el gestor de Entrada/Salida periódicamente verifica el estado de cada uno de los dispositivos del sistema. Para hacerlo, comprueba los contenidos de los registros de estado, de los puertos de Entrada/Salida asociados a los dispositivos; si hay datos provenientes del dispositivo, estos deben ser tratados; si no los hay, se pasa a verificar el siguiente dispositivo. Al hacerlo, la lista de dispositivos del sistema se puede recorrer en base a dos criterios diferentes:

  - **Gestión por muestreo con prioridad uniforme**: en este esquema, todos los dispositivos reciben las mismas oportunidades de ser atendidos; se pregunta al primer dispositivo si requiere atención y se le atiende, luego, se pasa al siguiente, y así sucesivamente hasta dar una vuelta completa.

  - **Gestión por muestreo con prioridad escalonada**: tras atender a cualquier dispositivo, se vuelve a comenzar el sondeo por el primer dispositivo. El orden es determinante, pues se establece una relación de prioridad entre dispositivos determinada por el orden de atención que esté siendo empleado.

  Estos mecanismos pueden llegar a no ser muy eficaces en sistemas interactivos debido a que requiere realizar una espera ocupada, por lo que se hace más lento atender a varios dispositivos de forma simultánea.

- **Gestión de E/S por interrupciones**: una interrupción es una señal o alarma que puede ser originada por un dispositivo de Entrada/Salida para notificar a la CPU que un evento ha ocurrido. Por lo general, esta es una señal de que una operación de Entada/Salida ha culminado o un error ha ocurrido y esta situación debe ser tratada. Por ejemplo, imagine que un proceso solicita un dato que está almacenado en un disco duro mediante la llamada al sistema *read*, en este caso, una vez que la información es obtenida del disco y está disponible para que la CPU la tome, el dispositivo levanta o genera una interrupción que es empleada para que se ejecute la rutina de tratamiento de interrupciones y está de paso al despachador para que cambie el estado del proceso, que esperaba la operación de entrada salida, de bloqueado a listo, para que al volver a ejecutarse pueda obtener los datos que ya estan disponibles y continuar su ejecuación. Las interrupciones son un mecanismo que ofrece la arquitectura de ordenador para comunicar los dispositivos de Entrada/Salida con la CPU, aunque también son empleadas para atender varios tipos de eventos asíncronos y excepciones que puede ocurrir en el sistema.

  El dispositivo de E/S emplea un canal de interrupción que conecta al dispositivo con la CPU. Cada uno de estos canales o líneas corresponde a un cierto tipo de dispositivo o a una familia de dispositivos de características similares. La asignación de canales y dispositivo es estática y sucede al momento de iniciar el ordenador, por tanto, no cambia a lo largo del tiempo. En caso de que haya datos a tratarse en el dispositivo, se notifica al procesador mediante la línea

de interrupción adecuada. Ante esto, la rutina de tratamiento de interupciones debe tomar el control, para dar paso al despachador para que tome el proceso que esté ejecutandose, lo detenega, pase el estado del proceso que habia sido bloqueado, a la espera de la finalización de la opeación de entrada/salida, de bloqueado a listo, para que pueda ser un candidato a volver a ejecutarse y en ese momento pueda obtener los datos que ya estan disponibles, para así continuar su ejecuación.

Las interrupciones no son programas reentrantes, esto implica que cuando se está realizando el tratamiento de una interrupción, se debe desactivar temporalmente la notificación de interrupciones similares o menos importantes. Por tanto, una interrupción se ejecuta hasta el fin de su tratamiento y solo en algunos casos específicos, una interrupción puede ser interrumpida por otra interrupción que esté ocurriendo también en ese momento, pero que sea más importante.

- **Gestión de Entrada/Salida Híbrida**: esta es una de las aproximaciones que es más usada en sistemas operativos modernos y que consiste en combinar las dos soluciones anteriores. En este tipo de gestión, primero se realiza una gestión por interrupciones, pero de ocurrir situaciones de estrés en las que exista una carga de trabajo muy alta que pueda generar una sobrecarga de conmutaciones, se limita en el tiempo la consulta de datos. En otro ejemplo, imagine que la tarjeta de red esté saturando al procesador con interrupciones, en este caso, se puede optar por atender a todas las interrupciones juntas de forma periódica, es decir, en lotes.

  Según la forma en la que son tratadas las solicitudes de información asociadas a una operación de Entrada/Salida, estas se pueden clasificar en:

  - **Entrada/Salida síncrona**: esta forma de operación implica que el proceso que inicia la operación de Entrada/Salida espera pacientemente hasta que esta finaliza e, inmediatamente tras la finalización de las operaciones de Entrada y Salida, el control es retornado al proceso que generó esa Entrada/Salida.

  - **Entrada/Salida Asíncrona**: en esta forma de operación se retorna al programa usuario sin esperar que la operación de Entrada/Salida finalice. Se necesita una llamada al sistema adicional que le permita al usuario esperar por la finalización de la operación de entrada y salida, si esto es necesario para el usuario.

  - **Acceso Directo de la Memoria (DMA o Direct Memory Access):** el acceso directo a memoria o *Direct Memory Access* (*DMA*) es un modo de operación en el que los dispositivos de Entrada/Salida tienen la habilidad de transferir directamente la información de los dispositivos de Entrada/Salida a la memoria del ordenador de forma autónoma e independiente del uso de la CPU, para ello, se emplea un sistema de buses que pueda trabajar de forma independiente a la CPU para comunicarse con la memoria. El proceso de *DMA* acelera el proceso de transferencia de los datos, ya que no hace falta que la CPU esté controlando el proceso de Entrada/Salida, por lo que el envío de información se puede hacer de forma concurrente y con alta probabilidad en un menor tiempo de ejecución.

4.3. Dispositivos

Los diferentes dispositivos que pueden pertenecer a un ordenador se pueden clasificar en dos grandes grupos:

- **Los dispositivos de procesamiento de caracteres**: que son dispositivos que pueden procesar un carácter a la vez, como son el caso del teclado, el ratón y los puertos seriales, entre otros.
- **Los dispositivos de procesamiento de bloques**: que son dispositivos que manejan bloques completos de información que, generalmente, son de tamaño fijo. Entre los dispositivos de bloque más conocidos están los discos duros.

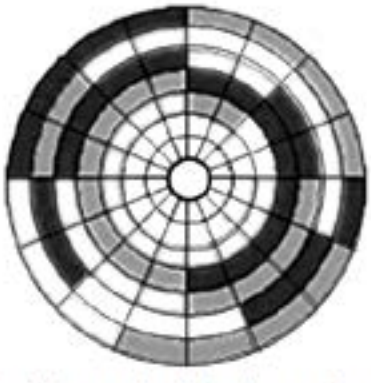
Ambos tipos de dispositivos requieren de buffers en memoria para ir almacenando los datos a enviar o los resultados parciales que vayan siendo calculados.

Los dispositivos de procesamiento de caracteres suelen ser más sencillos porque van enviando o recibiendo información en la medida que esta es generada, por ejemplo, cuando el usuario va presionando las teclas en el teclado o por programas que van enviando caracteres a ciertos elementos como las impresoras.

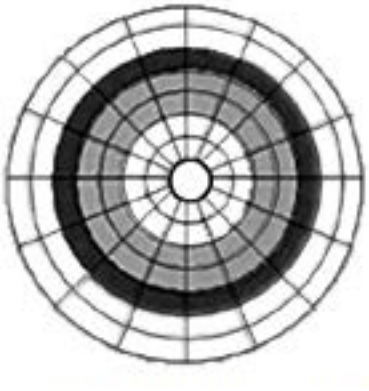
Los dispositivos de procesamiento por bloque, por su parte, deben ir armando lotes de información, muchas veces del mismo tamaño, antes de poder enviarlos. Al recibir un lote, los dispositivos deben buscar dentro de la unidad recibida la información que se requiere, lo que usualmente implica descartar algunos otros datos que están en el mismo lote para llegar a los datos que fueron efectivamente solicitados. Este tipo de dispositivo que usualmente puede tener una gran capacidad de almacenamiento es dividido en bloques de información, cada uno de los cuales contiene información específica de algún tema o proceso.

En el caso de que se requiera almacenar un grupo de datos y estos por su tamaño no caben en un solo bloque, entonces se asignan varios bloques a la tarea y estos bloques se enlazan formando listas enlazadas en las que cada bloque va apuntando al siguiente bloque, donde continua la información. Los bloques ocupados son organizados empleando un conjunto de índices o tablas para poder acceder de forma relativamente rápida a los elementos que se requieran. El conjunto de bloques libres o disponibles también es almacenado en alguna estructura de datos para poder ubicarlos rápidamente cuando se necesite almacenar un grupo de nuevos datos.

A medida que un dispositivo de bloques va siendo usado, se van ocupando y liberando secuencias de bloques de una forma que puede no ser la más ordenada; en estos casos el espacio del dispositivo puede llegar a quedar bastante fragmentado. En algunos dispositivos es posible reacomodar estos bloques de forma que los bloques con información relacionada queden relativamente cerca, lo que puede mejorar los tiempos de acceso a la información. Este mismo proceso también puede reunir los espacios libres de forma que se facilite su administración; a este proceso se le da el nombre de **desfragmentar el disco**.



Los bloques de datos de un mismo fichero están distribuidos a lo largo del disco



Los bloques de datos de un mismo fichero están juntos

Figura 4. 1. Compactación de bloques del dispositivo, en este caso un disco duro, con el fin de hacer que los bloques puedan ser leidos en menos operaciones de lectura reduciéndose el tiempo de estas operaciones en el sistema. Fuente: elaboración propia

En el caso de que una información no alcance para llenar uno de los bloques, el resto del bloque que no esté ocupado simplemente se desperdicia, ya que intentar ocupar ese pequeño espacio disponible puede complicar de forma importante la labor de administración de los bloques, por lo que muchos administradores prefieren que se desperdicie algo de espacio, pero logrando un esquema más simple de administración.

Algunos dispositivos de almacenamiento masivo como es el caso de los discos duros, suelen manejar una cantidad de peticiones de información provenientes de diferentes programas y diferentes usuarios; es labor del gestor de dispositivos determinar de qué forma y en qué orden son atendidos estos requerimientos, de manera que se puedan procesar la mayor cantidad de requerimientos en el menor tiempo posible.

4.3.1.Discos Duros

Los discos duros son los dispositivos por excelencia para almacenar de forma permanente los datos en el ordenador; están conformados por un sistema electromecánico en el que hay una superficie magnética que se encuentra en un disco o plato, este disco es el que contiene almacenados los datos de los usuarios. Un esquema de cómo es una unidad de disco duro lo podemos apreciar en la figura 4.2.

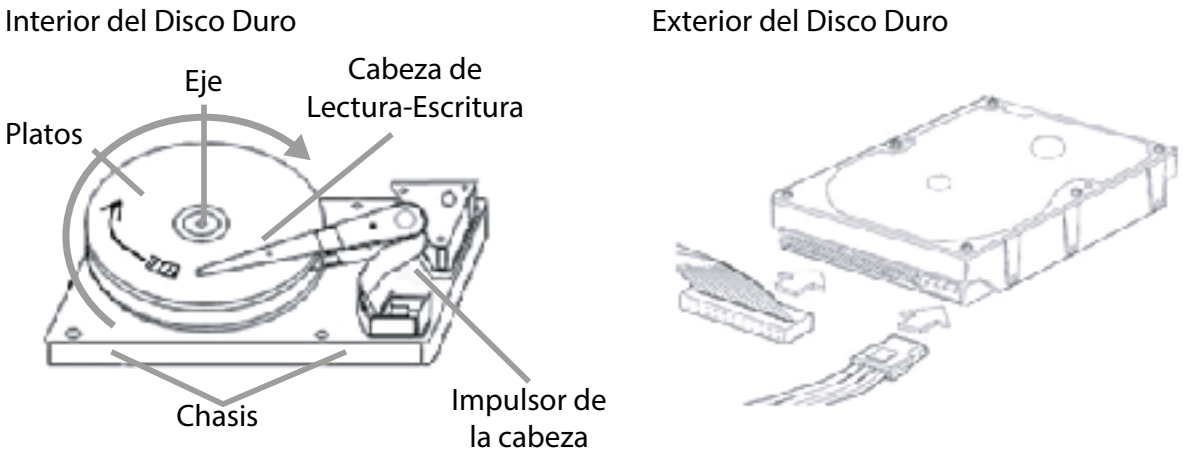


Figura 4. 2. Esquema interno y externo de una unidad de disco duro. Fuente: adaptado a partir de N. Carrodeguas (2016).

El disco recibe bloques de información y los va almacenando en la superficie magnética del plato, la cual se encuentra organizada en pistas y sectores como se puede apreciar en la figura 4.3. Durante su operación, el disco va girando y la cabeza lectora escritora se va moviendo para posicionarse en la pista adecuada y allí debe esperar hasta que el bloque de datos pase por debajo de la cabeza de Lectura Escritura para, efectivamente, leer el bloque de datos.

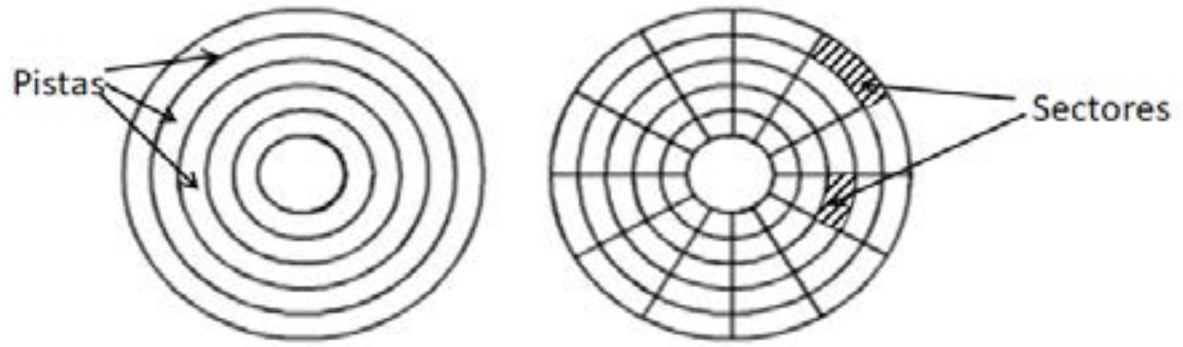


Figura 4. 3. Organización interna de la información en el disco duro. Fuente: elaboración propia.

Dado que la cantidad de bloques en el disco puede llegar a ser bastante grande, es importante llevar alguna forma de organización de la información que esta allí almacenada. Para ello, la pista más externa del disco usualmente es empleada para almacenar una especie de índice, tal como se presenta en la figura 4.4, de forma tal que sea mucho más sencillo ubicar cualquier bloque de información que esté presente en el disco e identificar a qué fichero corresponde y qué cantidad de información hay allí almacenada.

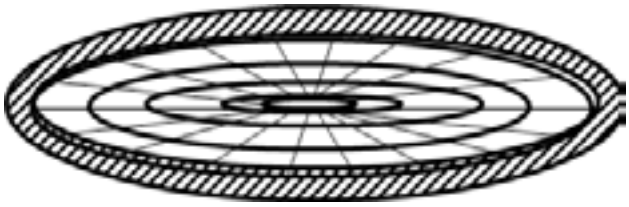


Figura 4. 4. Índice del disco duro en la pista más externa para facilitar la búsqueda de la información almacenada y la gestión de espacio en el disco. Fuente: elaboración propia.

A medida que se va trabajando con la unidad del disco duro, esta se van llenado de informaciones en diferentes bloques, pero también los usuarios van borrando datos, por lo que se producen algunos procesos de fragmentación en los que los bloques de información de un archivo pueden quedar dispersos a lo largo de toda la superficie del disco o inclusive en superficies diferentes, esto hace que la lectura de estos bloques de información se tome más tiempo, ya que es posible que después de leer un bloque, se deba mover la cabeza de lectura/escritura antes de poder volver a leer el siguiente bloque. Es común que los sistemas operativos cuenten con herramientas para desfragmentar los bloques del disco ya que esto tiende a mejorar el desempeño de los accesos a disco para la búsqueda de información.

4.4. Ficheros

La información que debe permanecer almacenada dentro del ordenador, aun cuando este se apague o desconecte, debe ser almacenada y gestionada de alguna manera. A la unidad mínima de información con la que el ordenador puede trabajar para almacenar datos de forma permanente se le da el nombre de fichero o archivo.

Un fichero es una unidad de información clasificada que es almacenada en un dispositivo para su conservación y consulta posterior en cualquier momento. Los ficheros contienen información en forma de secuencias de bits que pueden ser interpretadas de formas diferentes, como datos, en cuyo caso está conteniendo información que puede ser registrada y consultada por los usuarios o aplicaciones, y como programas, que son elementos que contienen un conjunto de instrucciones mediante las cuales se pueden ejecutar algoritmos y procedimientos que produzcan el procesamiento de ciertos datos con el fin de satisfacer alguna necesidad de información de los usuarios.

Debido a que la cantidad de ficheros que puede contener un ordenador es bastante grande, los ficheros son organizados en lo que se llama un sistema de ficheros o *File System*. Esto es una estructura de datos en la que hay principalmente dos tipos de componente; los ficheros que son repositorios de datos o programas; y los directorios o carpetas que son una abstracción que permite organizar los ficheros en grupos que faciliten su manejo por parte de los usuarios o por parte del mismo sistema. En los sistemas de archivos modernos también se suelen conseguir ficheros especiales que permiten el acceso a ciertos elementos, como es el caso de los enlaces o *links*.

Debido al gran volumen de información que puede contener un ordenador, esta se debe organizar internamente de alguna manera, para ello, se ha elegido trabajar con un esquema jerárquico en el que la información va siendo distribuida en diferentes niveles para que, en cada nivel, el volumen de información presente sea menor y pueda ser manejado por los usuarios de forma relativamente cómoda. Esto es parte de lo que hace el Sistema de Archivos.

4.5. Sistemas de Ficheros o Sistemas de Archivos (File System)

La organización de la información en ficheros y directorios es una abstracción que ha funcionado en los ordenadores desde hace mucho tiempo, está basada en una analogía donde los ficheros son representados como documentos y, los directorios, como carpetas en alusión a los elementos que se usan en un ambiente de oficina tradicional. Esta analogía está presente en todas las interfaces

gráficas de los sistemas de archivos, en las figuras 4.5 y 4.6 podemos apreciar la representación gráfica de estos elementos.

Figura 4. 5. Representación gráfica de los ficheros como documentos y los directorios como carpetas, en alusión a la metáfora del manejo de documentos de trabajo en una oficina. Fuente: elaboración propia.

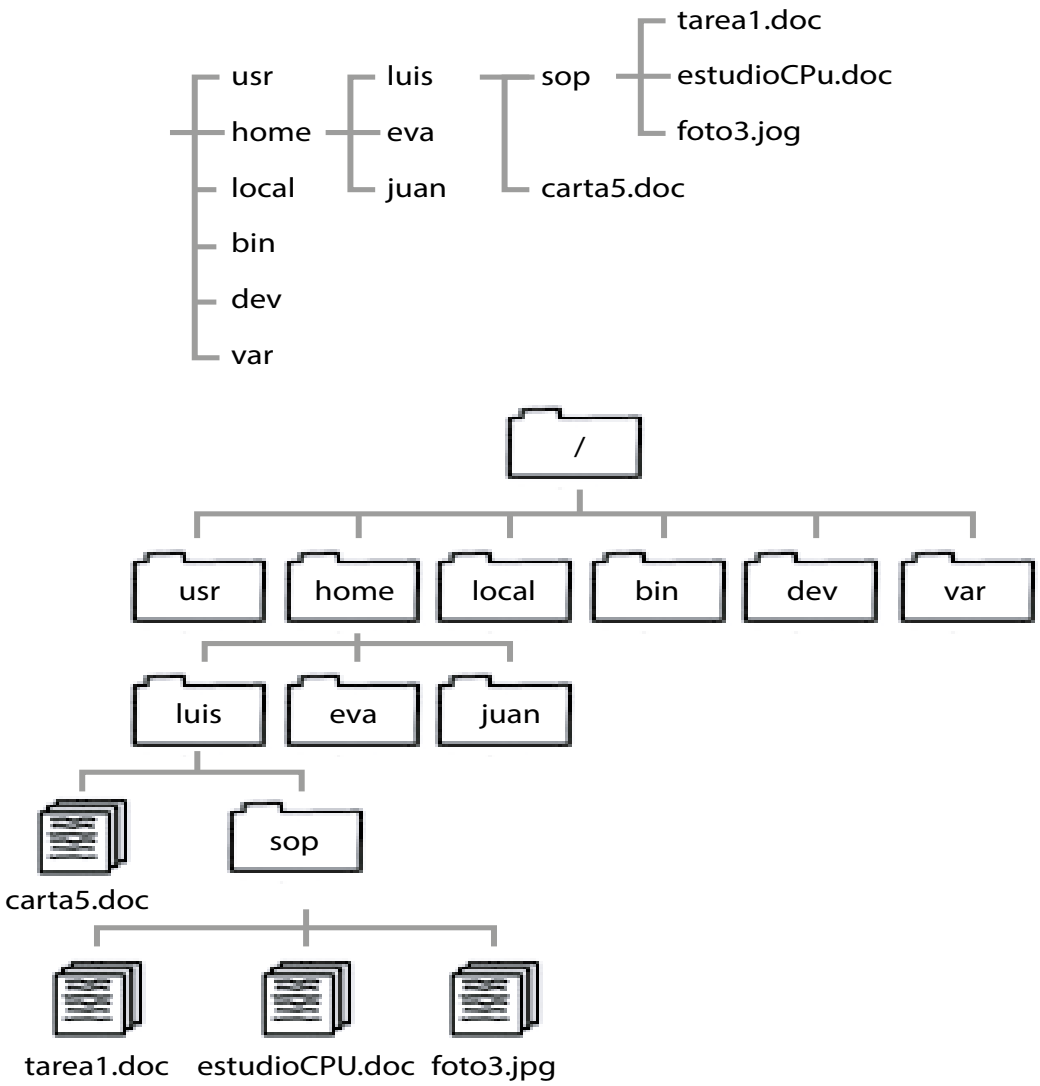


Figura 4. 6. Representación de un segmento del sistema de archivos de un ordenador como elementos gráficos de carpetas y documentos, este tipo de interfaz está presente en prácticamente todas las interfaces gráficas de los sistemas de ficheros. Fuente: elaboración propia.

Esta analogía es tan sencilla y natural para los usuarios que, en algunos sistemas operativos, se ha usado la misma analogía e incluso la misma interfaz del sistema de archivo para manejar componentes diferentes del sistema, como los dispositivos y los procesos que están en ejecución en el sistema. Por ejemplo, en Windows, cuando tengo un dispositivo, este es como una gran carpeta que contiene en su interior documentos o ficheros y otras carpetas. En el caso de Linux, el directorio /proc mantiene en su interior lo que parecen nombres de ficheros o directorios, pero que, en realidad, son la representación de los procesos que el sistema operativo está ejecutando en ese procesador, así como la representación de algunos de los elementos del hardware del equipo.

Los usuarios pueden usar los ficheros de diferentes formas: para extraer de ellos información mediante operaciones de lectura; para depositar en ellos información mediante operaciones de escritura o para mantener información actualizada, lo cual se puede lograr leyendo datos mediante operaciones de lectura y actualizando o eliminando aquellos que deben ser modificados mediante nuevas operaciones de lectura, escritura o eliminación de información.

4.5.1. Características de los ficheros en el sistema

En un sistema de ficheros, cada fichero tiene asociado una serie de estructuras de datos y campos que pueden almacenar una cantidad de datos, características e informaciones, de tamaño variable, acerca de su contenido y su uso. Entre las características primordiales que un fichero muestra, se encuentran:

- **Nombre**: que permite identificar unívocamente, es decir, de forma única al fichero o archivo dentro del sistema de ficheros del ordenador.
- **Descriptor de fichero**: que suele ser un número entero que se usa internamente en el ordenador para identificar el fichero. Para el ordenador, es más simple un número que un nombre para identificar al fichero, además el número suele ser más corto.
- **Extensión**: que es una parte del nombre del archivo, es opcional y puede identificar si los datos contenidos tienen un formato específico que permita que sean usados por algún programa en particular; por ejemplo, los programas con extensión .doc o .docx son archivos de texto que han sido elaborados por el programa Microsoft Word; los archivos con extensión .txt son archivos de texto y los programas con extensión .sh suelen ser programas en Linux que contienen Scripts o comandos que pueden ser ejecutados para implementar alguna función o requerimiento de información de los usuarios.
- **Ubicación**: del archivo dentro del árbol de directorios y archivos que conforman el *file system*.
- **Longitud**: que establece qué cantidad de información está almacenada o contenida en el fichero.
- **Permisos o permisología**: que son una serie de datos que establecen qué usuarios pueden acceder o no a la información de los archivos y qué pueden hacer con ella, si solo la pueden leer o si se les permite modificarla o incrementarla.

4.5.2. Estableciendo los permisos de acceso de un fichero

Para brindar un cierto nivel de seguridad sobre el uso que se hace de la información en un ordenador, el sistema operativo ha definido una serie de estructuras de datos que permiten establecer un conjunto de permisos para proteger la información, que contienen los ficheros contra intentos de accesos no autorizados por parte de los usuarios.

En este sentido, se puede dar permiso a los usuarios para leer, escribir o ejecutar los archivos, en el caso de que los archivos contengan programas e instrucciones. Esto se hace, por ejemplo, en los sistemas Linux estableciendo grupos de tres bits que reciben el nombre de rwx; si a un usuario se le asignan los permisos para un archivo con el valor 5 en decimal, o el valor 101 en binario, esto quiere decir que se le está activando el permiso para leer información del fichero (r, 100 o 4) y para ejecutar el fichero (x, 001 o 1) en caso de que sea un programa, pero no se le permite escribir información nueva dentro del fichero, ni modificar la información ya almacenada. Esto último se podria hacer, si se da el permiso de escritura correspondiente (w, 010, 2).

Usualmente, en Linux se le asigna a cada archivo un grupo de tres elementos para el dueño o creador del archivo (rwx), otro grupo de tres bits para los usuarios que trabajan o se relacionan con el dueño del archivo (rwx) y, un tercer grupo de tres bits, para cualquier otro usuario del sistema (rwx). De forma tal, que la representación de estos valores queda como: rwx r-x r-- o 111 101 100 para indicar que el dueño del archivo tiene permiso para leer, escribir o ejecutar el archivo; las personas que estén en el mismo grupo tendrán permisos para leer o ejecutar el archivo y, finalmente, el resto de los usuarios del sistema solo podrán leer el archivo.

4.5.3. Ficheros locales y Ficheros remotos

Los ficheros locales son aquellos que están contenidos en los dispositivos de almacenamiento secundario del ordenador y que pueden ser recuperados en cualquier momento. Los ficheros remotos son aquellos que están almacenados en un equipo diferente, pero que pueden ser accedidos al hacer una solicitud a un ordenador en una Red de computadores. En muchos casos, el sistema operativo puede gestionar archivos locales y remotos de una manera uniforme para el usuario, al menos a nivel de interfaz, es decir, que el usuario pueda acceder de la misma forma y con prácticamente las mismas llamadas a un archivo local o remoto. Es el sistema operativo el que se encarga de manejar las diferencias que deban manejarse, en caso de que la información no esté en el mismo ordenador en el que se genera a consulta.

4.5.4. Almacenamiento de ficheros en el ordenador

Los ficheros son almacenados dentro de los dispositivos de almacenamiento secundario, para esto, a cada fichero se le asigna un conjunto de Bloques de disco que dependerán de la cantidad de información que almacene el fichero. Si el archivo tiene la suficiente longitud, entonces se le pueden asignar varios bloques de disco a un mismo fichero y, de ser así, los bloques deben enlazarse de forma tal que un usuario que esté leyendo uno de los bloques se encuentre en disposición de conseguir, con relativa facilidad, donde está el siguiente bloque de información de ese mismo archivo en el *file system* del ordenador, tal como se muestra en la figura 4.7.

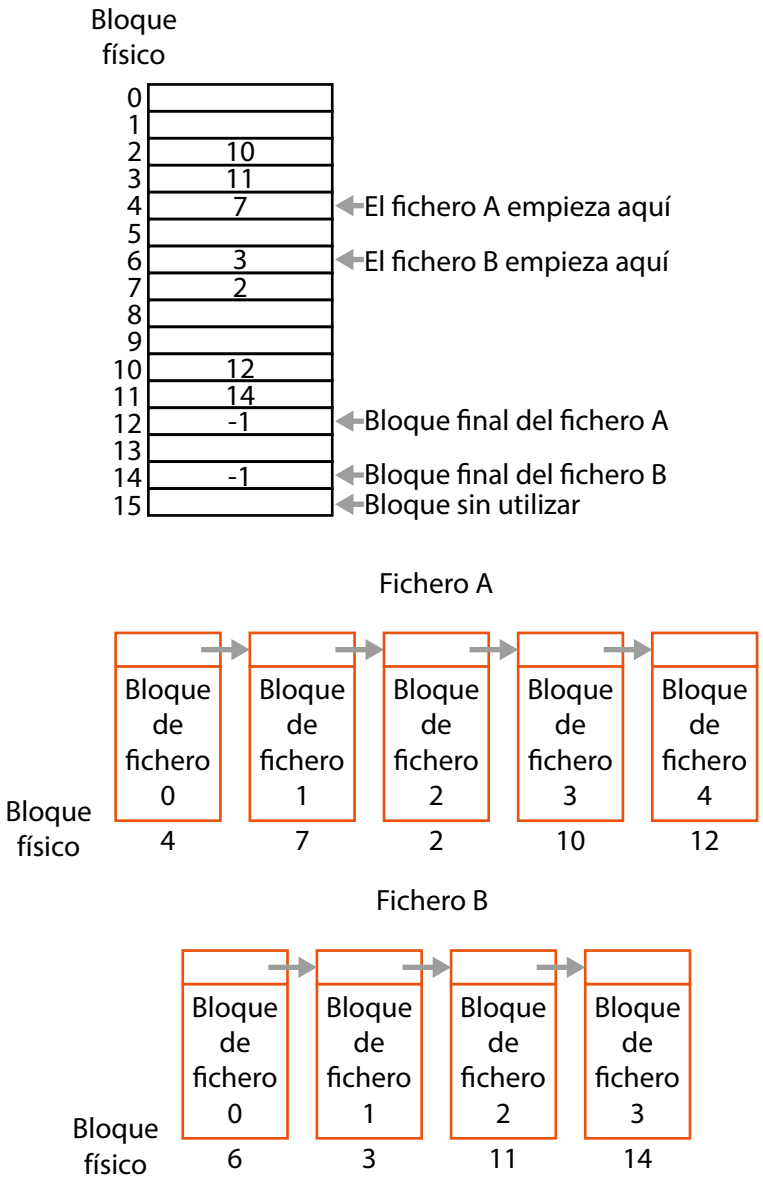


Figura 4. 7.Almacenamiento de bloques de información de un fichero en un dispositivo, donde cada nuevo bloque apunta al siguente bloque del mismo fichero en el sistema de ficheros, salvo el último bloque, que indica con una marca, que no hay más bloques asignados a ese fichero. Fuente: adaptado a partir de L. Castellanos (2014).

Tema 5.
Tendencias pasadas, presentes y futuras en los Sistemas Operativos

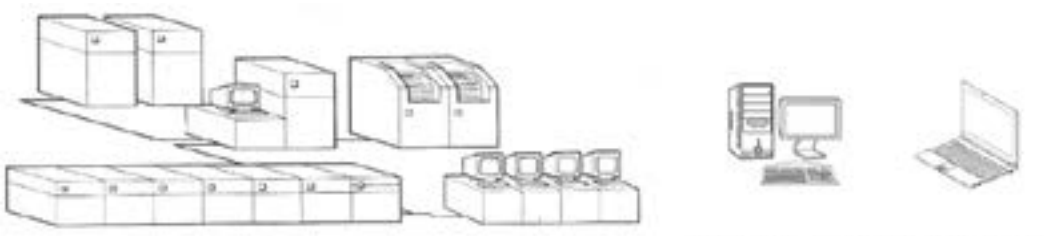
La evolución de los ordenadores ha producido una evolución también en los sistemas operativos que los administran y controlan.

5.1 Del Mainframe al computador Personal

En sus inicios, los computadores eran equipos grandes y costosos, por lo que se debían compartir entre un número de usuarios; estos primeros equipos recibieron el nombre de *Mainframe*. Para brindar una idea del tamaño de estos equipos, en la figura 5.1 mostramos un esquema de uno de ellos. A los de menos tamaño, se les llegó a llamar minicomputadores, aunque seguían ocupando el tamaño de un armario y eran significativamente más grandes que unas cuatro torres de servidores actuales.

Los bancos y las grandes empresas empleaban *Mainframes* para atender los programas de todos los usuarios de la organización. Los equipos de este estilo eran usados para aplicaciones de misión crítica, por lo que se esperaba que fueran muy confiables y robustos. En estos ambientes, nacieron los conceptos de multiprogramación, segmentación y paginación de la memoria, espacio de swap,

memoria virtual, sistemas de manejo de interrupciones, concurrencia y sincronización; es decir, la mayoría de los conceptos que hemos visto a lo largo del curso.



Mainframe para múltiples usuarios

Computadores Personales
PC un solo usuario

Figura 5. 1. Ordenador del tipo Mainframe y computadores personales. Fuente: elaboración propia.

El mundo de los ordenadores cambió significativamente con la aparición del Computador Personal o PC que permitió que cada persona tuviera acceso a un equipo propio que no necesitaba compartir con otros usuarios. Hacia finales de los 80 y en los 90, se popularizaron las computadoras personales y se desarrollaron sistemas operativos específicos para este tipo de equipos, entre ellos tenemos a: CPM, MS DOS, Windows, Mac OS, Unix y su descendiente GNU/Linux.

En un PC, normalmente hay un único usuario, por lo que la cantidad de tareas que debe realizar el sistema operativo no es tan grande; además, gracias a no tener que compartir el ordenador con otros usuarios, se pueden simplificar las actividades que necesita hacer un sistema operativo. Esto hace que los sistemas operativos personales pueden ser menos elaborados, ya que no hay necesidad de proteger los espacios de un usuario de lo que puedan hacer otros usuarios.

Por otra parte, la masificación de los PC permitió que cualquier tipo de usuario pudiera tener un ordenador, incluyendo aquellos que no tenían muchos conocimientos técnicos del uso y operación de esos dispositivos, por lo que las interfaces de los sistemas sufrieron una tremenda evolución de forma que hasta usuarios menos expertos pudieran emplear los recursos de un ordenador personal de manera más o menos sencilla. En este sentido, las contribuciones de los sistemas Apple inicialmente con el equipo Macintosh y el MacOS, junto con las diferentes versiones de Windows y las múltiples interfaces gráficas de Linux, han jugado un rol muy importante que es permitir que cualquier usuario pueda usar los recursos de un ordenador de forma simple, pero poderosa.

5.2 Los cambios que han introducido las Redes de computadores e Internet

El surgimiento de Internet y las redes de computadores cambiaron el panorama tecnológico al permitir que los equipos personales puedan acceder a información de otros equipos; esto provocó grades beneficios al permitir al usuario acceder no solo a la información que cada usuario procesa, sino también la que otros usuarios pueden llegar a generar, procesar y almacenar. Esto incrementó de forma importante la información a la que un usuario puede acceder, esto se multiplicó, de forma exponencial, con el uso de Internet, que es la gran red de redes a la que están conectados prácticamente la totalidad de los ordenadores del mundo.

Gracias a Internet, podemos consultar información que puede estar o haber sido generada en Nueva Zelanda, El Cairo, París, Chicago o en nuestra misma ciudad, sin importar la distancia y de forma prácticamente instantánea. Pero con este gran poder, también llegaron importantes problemas y calamidades. Los correos electrónicos basura o *Spam* con información que nunca solicitamos, los Virus y los *Malware*.

Ahora, los computadores no están solo expuestos a sus usuarios directos, sino a cualquier otro usuario que pueda estar conectado a Internet. Por ello, las amenazas se han multiplicado, un virus creado en Corea puede infectar ordenadores en Argentina y Luxemburgo en cuestión de minutos. Esto también ha cambiado el uso que damos a los sistemas operativos, ya que estos se han tenido que volver más seguros, controlando de una mejor forma los espacios de memoria y disco, evitando que algún proceso use o consuma los recursos de otro, porque estos son algunos de los mecanismos que emplean los programas malicioso y virus para propagarse. Aunque formalmente aún la protección contra virus y programas maliciosos no forma parte de la arquitectura estándar de los sistemas operativos, es difícil contar con una instalación que no contemple algún tipo de protección contra estas amenazas. La realidad es que los desarrolladores de sistema deben poner mucha atención a los errores y vulnerabilidades que sus sistemas puedan presentar.

5.3. Servidores

Las redes de computadores y, en especial Internet, al permitir que los usuarios se conectaran a otros equipos para solicitar e intercambiar información, generaron un modelo operación que lleva el nombre de **Cliente/Servidor.** En este modelo, los equipos de los usuarios normales son clientes que requieren de algún tipo de recurso y lo solicitan a los servidores.

5.3.1. Los Servidores, aplicaciones o equipos físicos

Los servidores son aplicaciones o equipos que poseen los recursos y los ponen a disponibilidad de los clientes a través de una serie de mecanismos. En informática, el término servidor está asociado un tipo de "Software" o programa que realiza ciertas tareas en nombre de los usuarios. Aunque el término también se utiliza para referirse al "computador físico", en el cual funciona ese software, una máquina cuyo propósito es proveer datos e información, de modo que otras máquinas y sus usuarios puedan utilizarlos. Esta dualidad en el uso de la palabra Servidor puede ocasionar algunos inconvenientes.

Hay muchos servidores en Internet y muchos tipos de servidores, pero todos comparten la función común de proporcionar el acceso a ciertos recursos, tales como: Ficheros o Archivos, Espacio de Almacenamiento, Video y otros elementos Multimedia, Servicios de Colaboración, Aplicaciones, Servicios de Cómputo, Páginas Web, Listas, Correos, Noticias, Proxys, Chats, entre muchos otros recursos.

5.3.2. Sistemas Operativos para Servidores

En general, los Sistemas Operativos hacen la distinción de si son S.O. para clientes (estaciones de trabajo), o bien para Servidores (equipo de cómputo más robusto), pero, en qué se diferencia

realmente un sistema operativo de un servidor de un sistema operativo para un cliente, pues el sistema operativo para un Servidor debe:

- Estar en la capacidad de soporta configuraciones complejas.
- Permitir configuraciones de varios procesadores de última tecnología, muchos Gigas o incluso Terabytes de memoria y almacenamiento para poder procesar rápidamente los requerimientos de los usuarios.
- Poder almacenar grandes cantidades de datos en dispositivos de almacenamiento secundario.
- Permitir sustituir componentes averiados sin la necesidad de apagar el sistema para llevar a cabo labores de mantenimiento (*Hot Swap*) ya que, en el caso de los sistemas de misión crítica, puede no ser posible detener un servicio para realizar alguna actividad de mantenimiento o reemplazar un componente dañado.
- Puede tener componentes redundantes que le permitan exhibir características de tolerancia a fallos.
- El software debe poder utilizar al máximo las funcionalidades y capacidades del equipo.
- Debe estar en la capacidad de controlar máquinas potentes.
- Debe estar enfocado en ofrecer una serie de servicios a clientes, tanto locales como remotos.

Un servidor dedicado debe estar diseñado para manejar prácticamente cualquier tarea que se le asigne, desde aquellas que requieren alto tráfico hasta el almacenamiento de archivos y descargas extremas, por esta situación el sistema operativo que use debe manejar los recursos de forma mucho más eficiente y confiable, al compararlo con lo que hace un SO para un computador personal.

Adicionalmente, un sistema operativo para un servidor debe contar con:

- Un monitor que permita identificar cuáles son los eventos más resaltantes que van ocurriendo en el computador y que llame la atención del operador cuando se presente algún riesgo para la operación adecuada del equipo y sus programas.
- Un mecanismo de *Logs* que registre lo que va ocurriendo durante la operación del equipo para saber qué ha ocurrido en todo momento, ya que, al llevar una traza de los eventos que ocurren en el sistema, se podrá analizar en caso de fallo para detectar posibles situaciones de riesgo y poder hacer análisis "post morten" de ciertas situaciones.
- Un mecanismo para realizar copias de seguridad, en caso de que ocurra una falla catastrófica. En este caso, uno de los posibles planes de acción es reiniciar el computador y llevarlo a algún estado previo que sea consistente, por eso es importante contar con un mecanismo que permita realizar copias de seguridad periódicas que puedan ser usadas para restaurar el sistema en caso de fallas.

- Ofrecer mecanismos de seguridad, autentificación y de control de accesos. Un mecanismo de autenticación permite verificar que el usuario realmente es un usuario válido del sistema, pero esto no es suficiente, se debe contar con mecanismos para que los usuarios que entren al sistema estén realmente autorizados para hacerlo y para acceder a los recursos que solicitan.

Actualmente, los sistemas operativos más conocidos cuentan con versiones específicas para servidores, tal es el caso de las varias distribuciones de Linux; existen servidores Windows (Windows Server) que poseen una interfaz gráfica que facilita su uso y también Servidores Mac OS y OSX.

5.4 Dispositivos Móviles

Desde el surgimiento del iPhone, el mundo de los dispositivos móviles sufrió un cambio importante y aunque existían asistentes personales, agendas digitales y poderosas calculadoras portátiles, no es hasta el surgimiento de los teléfonos inteligentes, y luego de las tabletas, que se difundió el uso de dispositivos pequeños con pantallas táctiles para que el usuario pudiera interactuar de forma más sencilla con un equipo pequeño y con limitaciones de recursos. Sin embargo, los equipos móviles de hoy en día están mucho más cerca de un computador de propósito general que de un teléfono; además de esto, el éxito comercial de los dispositivos móviles ha hecho que, a partir del 2015, la cantidad de dispositivos móviles en el mundo haya sobrepasado a la cantidad de computadores de escritorio existentes.

Pero los dispositivos móviles y, por ende los sistemas operativos que estos dispositivos emplean, deben adaptarse a una nueva serie de realidades, diferentes a las de un sistema operativo de un computador personal de escritorio. En este sentido, los sistemas trabajan inicialmente para un único usuario, lo que simplifica algunas actividades, aunque después de las primeras versiones incluyeron también la posibilidad de multiprogramación para que pudiéramos usar diferentes aplicaciones más o menos al mismo tiempo, limitados solo por el tamaño de la pantalla del dispositivo y ya no por las características del sistema operativo que lo controla y administra.

Un segundo aspecto tiene que ver con el manejo de energía, al ser dispositivos portátiles, es difícil que los dispositivos móviles estén conectados de forma permanente a un sistema de potencia eléctrica, por lo que deben operar mediante baterías, pero mientras más operaciones realice el dispositivo, más funciones y mayor sea la pantalla, mayor será el consumo de potencia y menor la duración de la batería. En este sentido, las interfaces y los sistemas operativos manejan formas de operación de bajo consumo en los que varios de los componentes del equipo pueden ser puestos a dormir para reducir el consumo de potencia y después ser despertados nuevamente cuando se requiera su uso regular.

Un tercer aspecto, es que los dispositivos móviles poseen una cantidad importante de sensores que les permiten recolectar información de su entorno, entre los sensores se encuentran GPS, acelerómetros, micrófonos, cámaras, sensores de humedad y temperatura, ente otros; esto permite nuevos modelos de uso que van desde recepción de información geolocalizada, gracias a la ayuda del GPS, hasta el uso de alarmas ante la detección de eventos sísmicos gracias al uso de acelerómetros y la medición de recorridos con la combinación del uso de GPSs y acelerómetros. Pero esto requiere que el sistema operativo posea importantes mecanismos para la detección y atención de estos eventos.

Aunque ha existido una serie importante de sistemas operativos para dispositivos móviles como Symbian, Blackberry OS, Windows Mobile, Firefox OS, en la actualidad, el mercado de dispositivos móviles es dominado prácticamente por los sistemas iOS de Apple y Android de Google.

5.5 IoT y los sistemas embebidos

Internet de las **Cosas o IoT**, que es un término que se ha popularizado recientemente y que se basa en la construcción de sistemas en los que todas las cosas puedan estar conectadas a Internet para suministrar, recolectar o consumir información en todo momento.

El término internet de las cosas hace referencia a "las cosas"; está intentando abarcar una muy amplia gama de diferentes dispositivos físicos que incluyen: vehículos, edificaciones, negocios, electrodomésticos y un amplio rango de elementos integrados con componentes electrónicos, software, sensores, actuadores y conectividad que permite que estos objetos se conecten e intercambien datos e información a través de la Internet actual.

El desarrollo de sistemas operativos para este tipo de sistema es un reto enorme que contiene algunos de los aspectos que ya han enfrentado los dispositivos móviles como el uso de dispositivos con recursos limitados de procesamiento y almacenamiento de datos, el manejo del consumo de potencia eléctrica para aumentar el tiempo de vida de las baterías con las que operan muchos de los dispositivos de este tipo; pero, dado que se desea desplegar grandes desarrollos con muchos dispositivos para reducir el costo de estos, se emplean dispositivos aún más pequeños y con menos recursos que los teléfonos inteligentes, donde incluso puede ser difícil que pueda estar almacenado un sistema operativo de propósito general. Una de las soluciones que se ha usado en este caso en que el sistema operativo se funda con la aplicación que va a ser desplegada y, por tanto, se instalen solo los módulos que la solución que se va a implementar requiera, esto puede reducir de forma importante el espacio que el sistema operativo requiera.

El uso de sensores y la conexión de un dispositivo con otros, con características y prestaciones diferentes a las de otros de los dispositivos que estén en su entorno , es otra de las características más importantes de los sistemas para IoT y que los sistemas operativos para estos equipos deben manejar.

5.6 Sistemas en la Nube

La computación en la nube es un modelo de prestación de servicios de negocio y tecnología en el que se ofrecen o se reciben servicios de computación a través de Internet. Esto implica que a los usuarios se les permite, mediante un acceso de red, el acceso bajo demanda a un grupo de recursos configurables que pueden incluir: servidores, almacenamiento, aplicaciones, instrumentos y servicios que pueden aprovisionarse y lanzarse rápidamente con un mínimo esfuerzo de gestión o interacción con el proveedor de estos servicios. El modelo de computación en la nube puede brindar a los usuarios el acceso a un catálogo de servicios estandarizados que le permitan responder a las necesidades de un negocio de forma rápida, flexible y adaptativa. Esto es especialmente importante en caso de que se presenten demandas inesperadas y no previsibles de trabajo. Para poder usar nuevos equipos, el proceso es muy rápido y eficiente, no hay que esperar a que los equipos sean adquiridos, simplemente se solicitan los recursos que se requieran y estos son colocados a su disposición por el proveedor de

los servicios. En este modelo, el usuario puede pagar específicamente por el consumo de recursos que realmente haga, incluso el servicio puede llegar a ser gratuito, esto ocurre en algunos casos muy especiales en los que los proveedores se financian mediante otros modelos de negocio como la publicidad, o en el caso de que los servicios sean brindados por organizaciones sin ánimo de lucro.

Los riesgos que representan usar este modelo de operación, implican que la disponibilidad de los servicios de la nube será tan robusta como la conexión a Internet con la que se cuente, ya que los servicios se brindan es desde Internet, por lo que la interrupción de la conexión implica que el servicio no estará disponible por ese lapso de tiempo.

La computación en la nube puede hacer que la organización dependa de la confiabilidad de su conexión a Internet. Si el servicio de Internet sufre interrupciones o velocidades bajas, la computación en la nube puede no ser la mejor opción para esa organización. Otro problema con el que se pueden topar sistemas basados en la nube es la compatibilidad con todos los sistemas de TI de una empresa. Hoy en día, se reconoce universalmente que la computación en la nube funciona como la opción más rentable para las empresas. Sin embargo, el problema surge del hecho de que la organización puede tener que reemplazar muchas de sus infraestructuras de TI existentes para hacer que el sistema sea cien por cien compatible con los servicios en la nube que emplee.

En este modelo de operación, los sistemas operativos pueden tener participación en cuatro lugares diferentes. En primer lugar, en el equipo que es usado para acceder a la nube; este es el equipo que emplea el usuario para conectarse. En principio, lo que se requiere es que el usuario esté conectado a Internet y que emplee un navegador web para interactuar con el sistema en la nube, por lo que con cualquier sistema operativo tradicional o incluso ligero, será suficiente para conectarse a la red y de allí a los servicios de la nube. Ahora los recursos que el usuario ve en la nube deben ser administrados y se les debe brindar una interfaz para que el usuario pueda interactuar y solicitar servicios; aquí hace falta un sistema operativo para manejar los recursos que el usuario emplea y solicita de la nube. En el caso de que la nube sea empleada para obtener servicios de plataformas de hardware, estas plataformas tendrán que contar con un sistema operativo que el usuario pueda manejar para administrar los equipos, posiblemente virtuales, que le sean asignados.

Finalmente, los equipos de los centros de cómputo que se usan para ofertar los servicios de nubes deben contar con un sistema operativo que les permita brindar una cantidad importante de servicios remotos a sus usuarios, lo que implica no solo altos niveles de confiabilidad, sino también la posibilidad de manejar esquemas de virtualización que permitan optimizar la adquisición y el uso de los recursos que el proveedor de servicios coloca a disposición de sus usuarios.

Glosario

Aplicaciones

Conjunto de uno o más programas que, en conjunto, realizan la tarea general que un usuario le ha encomendado.

Archivo

El archivo o fichero es una unidad de información compuesta por una secuencia de bits que permite el almacenamiento de información, datos o programas, de forma permanente en un dispositivo de almacenamiento secundario dentro del ordenador.

Arquitectura de Kernel

Es un tipo de arquitectura de Sistema Operativo en la que las rutinas específicas de manejo de procesos o recursos, que deben ser realizadas solo por el sistema operativos y todas las llamas asociadas al uso de recursos comunes a varios procesos, son concentrados en una parte especial del Sistema Operativos que lleva el nombre de kernel o núcleo; aquí es el sistema operativo el que ejecuta en un modo de operación especial estas rutinas, lo que le permite al usuario el acceso a recursos considerados críticos, que deban ser compartidos adecuadamente para el correcto funcionamiento del ordenador.

Arquitectura monolítica

Es una arquitectura donde el sistema operativo se construye en una sola pieza que contiene todos los comportamientos que debe contener para poder realizar sus funciones.

Bloqueos Mutuos

Situación en la que dos o más procesos están bloqueados y no pueden continuar porque cada uno de ellos mantiene una serie de recursos y solicita otros que no puede obtener porque otro de los procesos del sistema los tiene asignados y estos procesos, a su vez, no liberarán los recursos que tienen asignados, esto ocurre porque, a su vez, están bloqueados a la espera de recursos que nuestro primer proceso tiene reservados. Aquí se produce una situación de bloqueo mutuo o de espera circular que los procesos no pueden romper sin ayuda del sistema operativo o del usuario. Otros nombres que son usados para este mismo concepto son: abrazo mortal, *dead lock* e interbloqueo.

Contador de Programa

Registro de la CPU que almacena la dirección de la siguiente instrucción a ejecutar en el proceso actual. Este es uno de los datos que se almacena en la PCB de un proceso.

CPU

Esta abreviatura corresponde a las siglas del término en inglés *Central Processing Unit* o Unidad Central de Proceso, que es algo así como el cerebro del ordenador, es decir, es el componente que se encarga de la ejecución de los programas en el ordenador.

Despachador

Componente del sistema operativo cuyo trabajo es tomar los procesos e irlos cambiando de Cola, estado o de estructura de datos a medida que se van ejecutando o que realizan operaciones que hacen que su estado de ejecución deba cambiar.

Espera Ocupada o Polling

Es un mecanismo para la ejecución de diferentes tareas en el que un componente va preguntando al resto de los elementos del sistema si tienen algo que aportar o un requerimiento por hacer y, de tenerlo, es atendido para luego pasar a preguntar al siguiente componente hasta llegar al final y, después, volver a iniciar el proceso de preguntas desde el primero de los componentes o elementos.

Interfaz de Comandos o Shell

Es un programa del sistema operativo cuyo trabajo es recibir comandos del usuario y ejecutar estos comandos, para que el usuario pueda correr sus programas y administrar los recursos del computador a través de estos comandos. Es una interfaz para que los usuarios puedan emplear los servicios del sistema operativo para usar el ordenador y sus recursos.

Job o Trabajo

Unidad de procesamiento que el ordenador puede ejecutar de forma autónoma, independiente y sin la intervención del usuario, salvo al momento de entregar el trabajo al sistema operativo o al buscar sus resultados.

Kernel

Núcleo del sistema operativo; es aquel componente que engloba las funciones fundamentales del sistema operativo que tienen que ver con la gestión de recursos compartidos y escasos que debe realizar el ordenador y que deben ser realizados por un componente imparcial que permita dar el tratamiento adecuado a cada programa que se desee ejecutar.

Llamadas al sistema

El sistema operativo ofrece a sus usuarios una interfaz a través de la invocación de un conjunto de llamadas a rutinas y funciones, mediante las cuales los usuarios pueden solicitar al sistema operativo que realice alguna actividad en su nombre. Como la actividad la realiza el sistema operativo, este

puede mantener un grado importante de control sobre recursos y estructuras que pueden ser de uso delicado; las cuales, si fueran usadas directamente por los usuarios, se podrían llegar a generar problemas que afecten el comportamiento de varios de los procesos en los que esté trabajando el sistema operativo.

Micro Kernel

Es un tipo de arquitectura en la que el kernel del sistema operativo ha sido optimizado para que sea bastante pequeño y altamente eficiente.

Modelo Cliente/Servidor

Es un modelo de programación en el que se identifican claramente dos programas independientes; uno de los cuales posee una información o recurso y que se llama Servidor y, otro programa, que requiere la información que el servidor posee y administra, recibiendo el nombre de Cliente. El cliente realiza una solicitud de información al servidor siguiendo una serie de pasos estructurados que reciben el nombre de protocolo, y que debe ser acordado y aceptado por todas las partes involucradas, para que pueda ser empleado efectivamente para lograr la comunicación de los procesos clientes con el servidor.

Modo Kernel

Modo de operación de un programa en el que posee permisos especiales que le permiten acceder a todos los recursos del ordenador.

Modo Supervisor

Ver modo Kernel.

Modo Usuario

Es un modo de operación de los programas en el que el programa posee permisos restringidos para emplear algunos de los recursos del ordenador. Cuando un programa requiere acceder a otros servicios que no funcionen en modo usuario y que deban ser compartidos o que puedan ser accedidos por varios procesos de forma simultánea, se debe pedir tener acceso en modo kernel para poder realizar alguna de estas operaciones.

Movimiento del Software Libre

Es un movimiento social cuyo objetivo es obtener y garantizar una serie de libertades a los usuarios de aplicaciones que les permitan ejecutar, estudiar, modificar y redistribuir copias de la misma aplicación o programa, con o sin cambios, bajo unas ciertas condiciones.

Página

Unidad de información en las que se puede dividir la memoria para hacer un uso más rápido y eficiente de los espacios disponibles en la memoria principal del ordenador.

Paginación

Proceso en el que se hace el manejo y gestión de los espacios de una memoria principal que está dividida en páginas.

Planificador de Procesos

Programa que se encarga de identificar cuál debe ser el próximo proceso que debe ejecutarse en el ordenador, en función de los programas que estén en la cola de *Listo* y de las políticas de asignación del procesador que el sistema posea.

Polling

Ver el concepto de Espera Ocupada.

Procesamiento Interactivo

Procesamiento de programas en los que estos realizan interacciones constantes con sus usuarios a través de dispositivos de Entrada/Salida.

Procesamiento por Lotes o Batch

Tipo de procesamiento en el que un conjunto de actividades es entregada al sistema operativo para que este gestione de forma autónoma la ejecución de las tareas y actividades involucradas, sin que estas tengan que interactuar con sus usuarios. Una vez finalizadas las actividades asociadas al Lote de trabajos o Jobs, el usuario podrá pasar luego a ver qué resultados se obtuvieron durante su ejecución.

Programa

Es la implementación de un algoritmo en un lenguaje que pueda ser entendido de alguna manera por un ordenador.

Programa Ejecutable

Es un programa escrito en un lenguaje que permite su ejecución directamente en el ordenador.

Programas Concurrentes

Un conjunto de programas que están en el ordenador se pueden ejecutar de forma concurrente si estos pueden ser iniciados en el ordenador y estar en diferentes estos de su ejecución en un instante de tiempo dado.

Programas de Sistema

Conjunto de programas que, al ejecutarse, no resuelven los requerimientos directos de información de los usuarios, sino que sirven para apoyar la ejecución de otros programas o actividades en el ordenador.

Proceso

Es la representación de un programa que se está ejecutando en el ordenador.

Quantum

Unidad de tiempo en el ordenador que es empleada para asignar una misma cantidad de tiempo a cada proceso que se está ejecutando; este es un término muy usado en la política de *Round Robin*. Cuando un proceso se ejecuta por un periodo de tiempo igual a su quantum, entonces el sistema operativo suspende ese proceso para pasar a ejecutar otro de los procesos disponibles en la cola de *Listos*.

Segmentación

Técnica de asignación de espacio de memoria en la que se le asigna a cada proceso un conjunto de espacios no uniformes de memoria a los que se les da el nombre de segmento de memoria.

Segmento

Es un área de memoria no uniforme, es decir, que no todos los segmentos de la memoria deben tener el mismo tamaño, que se le asigna a un proceso para que almacene su programa (segmento de texto) o las variables internas (segmento de datos).

Shell

Interfaz de comandos que ofrece el sistema operativo para permitir la interacción directa del usuario con los recursos y servicios que posee el ordenador.

Sistema Manejador de Interrupciones y Excepciones

Serie de rutinas, mecanismos y elemento de hardware que permiten la atención de eventos asíncronos en el ordenador; el sistema manejador de interrupciones es invocado cuando una señal, alarma o interrupción, proveniente de algún dispositivo de almacenamiento, indica que la operación de Entada/Salida que había sido solicitada ya ha sido realizada y los datos requeridos ya están disponibles en la memoria. También es un mecanismo que permite la ejecución de ciertos procesos cuando ocurra una interrupción de reloj o *timer*, lo que permite la ejecución de procesos cada cierto tiempo o la posibilidad de interrumpir algún proceso para dar paso a otro que aún no haya podido ejecutarse.

Sistema Operativo

Programa que se encarga de la gestión de los recursos del ordenador y que facilita la ejecución de los programas de los usuarios.

Tiempo de ejecución

Es el lapso de tiempo que transcurre desde que un proceso ingresa al sistema hasta que este finaliza su ejecución, ya sea que esto ocurra de forma natural o provocada.

Tiempo de Espera

Es el tiempo que pasa un proceso en alguna de las colas del sistema operativo, ya sea la espera de la CPU para ejecutarse, o a la espera de un dato que vengan de algún dispositivo. Aquí, se contabiliza todo el tiempo que transcurre desde que el proceso llega al sistema hasta que termina, pero restando el tiempo en que el proceso ha estado ejecutándose en la CPU.

Trabajo o Job

Unidad de trabajo que, para su ejecución, no requiere de la interacción con su usuario.

Bibliografía

Referencias bibliográficas

Ampudia, M. (2012). Arquitectura de la Torre. Recuperado de: http://reiminx.blogspot.com.co/2012/02/disco-duro.html

Anasagasti, P., Pérez, F. (2016). Sistemas Operativos. Universidad Politécnica de Madrid. Recuperado de: http://www.elai.upm.es/moodle/pluginfile.php/3574/mod_resource/content/1/sistemasoperativosupm.pdf

Castellanos, L. (2014). Sistemas Operativos. Recuperado de: https://lcsistemasoperativos.wordpress.com/tag/contigua/

Carrodeguas, N. (2016). Cómo conectar y agregar otro disco duro interno a la PC, tutorial. Recuperado de: https://norfipc.com/articulos/como-conectar-instalar-otro-disco-duro-interno-pc.html

Hang, T. D. (2017). Operating system: from 0 to 1.

Hansen, B. (1972). Operating system principles.

Hormechea, M. G., Hernández, C. M., Lizcano, R. N., Botero L. F., Chavarro A. Y., Lozada, S. (s.f.). Sistemas operativos para Servidores. Recuperado de: https://senaintro.blackboard.com/bbcswebdav/institution/semillas/217219_1_VIRTUAL/OAAPs/OAAP1/aa1/dcto_so_server/sistemas_servidores.pdf

Olivares, J. C. (s.f.). Network Operating Systems. Recuperado de: http://dsc.itmorelia.edu.mx/~jcolivares/courses/nm091q/nm_u3.ppt

Ramírez, J. (2014). Descripción del esquema del disco duro. Recuperado de: http://zaraitarangoperlaalanis.blogspot.com.co/2014/05/descripcion-esquema-del-disco-duro.html

Tanenbaum, A. (2009). Sistemas Operativos Modernos (3ª Ed.). Pearson Education. México.

Vanover, R. (2017). Servidores físicos vs. Máquinas virtuales. Veeam Blog. Recuperado de: https://www.veeam.com/blog/es-lat/why-virtual-machine-backups-different.html

Wikipedia. Computación Grid. Recuperado de: https://es.wikipedia.org/wiki/Computaci%C3%B3n_grid

Wolf, G., Ruiz, E., Bergero, F., Meza, E. (2015). Fundamentos de Sistemas Operativos. (1ª Ed.). Universidad Nacional Autónoma de México. Recuperado de: https://sistop.org/pdf/sistemas_operativos.pdf

1&1. Los sistemas operativos para servidores: historia y situación actual. Recuperado de: https://www.1and1.es/digitalguide/servidores/know-how/los-sistemas-operativos-para-servidor-a-traves-del-tiempo/

Bibliografía recomendada

Anasagasti, P., Pérez, F. (2016). Sistemas Operativos. Universidad Politécnica de Madrid. Recuperado de: http://www.elai.upm.es/moodle/pluginfile.php/3574/mod_resource/content/1/sistemasoperativosupm.pdf

Tanenbaum, A. (2009) Sistemas Operativos Modernos (3ª Ed.) Pearson Education. México.

Wolf, G., Ruiz, E., Bergero, F., Meza, E. (2015). Fundamentos de Sistemas Operativos. (1ª Ed). Universidad Nacional Autónoma de México. Recuperado de: https://sistop.org/pdf/sistemas_operativos.pdf

Agradecimientos

Autor
D. Ricardo González García

Departamento de Recursos para el Aprendizaje
D.ª Carmina Gabarda López
D.ª Cristina Ruiz Jiménez
D.ª Sara Segovia Martínez