Tema 4

Sistemas operativos. Características y elementos constitutivos. Sistemas Windows. Sistemas Unix y Linux. Sistemas operativos para dispositivos móviles.

Guion-resumen

1. **Introducción**
2. **Concepto y funciones**
3. **Clasificaciones**
4. **Trabajos, procesos e hilos**
5. **Gestión de recursos y planificación**
6. **Memoria**
7. **Entrada/salida**
8. **Sistemas Windows**
9. **Sistemas UNIX y LINUX**
10. **Sistemas operativos para dispositivos móviles**

1. Introducción

1.1. Cuestiones generales

A finales de los años 40, el uso de computadoras estaba restringido a aquellas empresas o instituciones que podían pagar su alto precio y no existían los sistemas operativos. En su lugar, el programador debía tener un conocimiento profundo del **hardware**, y en el desgraciado caso de que su programa fallara, debía examinar los valores de los registros y paneles de luces indicadoras del estado de la computadora para determinar la causa del fallo y poder corregir su programa, además de enfrentarse nuevamente a los procedimientos de apartar tiempo del sistema y poner a punto los compiladores, ligadores, etc; es decir, para volver a correr su programa, enfrentaba el problema del procesamiento en serie.

La importancia de los sistemas operativos nace históricamente en los años 50, cuando se hizo evidente que trabajar en una computadora por medio de tableros enchufables en la primera generación y por medio del procesamiento por lotes en la segunda generación podía mejorar notoriamente, pues el operador realizaba siempre una secuencia de pasos repetitivos, lo cual es una de las características contempladas en la definición de lo que es un programa. Por tanto, se comenzó a ver que las tareas mismas del operador podían plasmarse en un programa, el cual a través del tiempo y por su enorme complejidad se le llamó **sistema operativo**. Así, tenemos entre los primeros sistemas operativos al ***Fortran Monitor System* (FMS) e IBSYS.**

Posteriormente, en la tercera generación de computadoras nace uno de los primeros sistemas operativos con la filosofía de administrar una familia de computadoras: el OS/360 de IBM. Fue este un proyecto tan novedoso y ambicioso que supuso por primera vez una serie de problemas conflictivos debido a que anteriormente las computadoras eran creadas para dos propósitos en general: comercial y científico. Así, al tratar de crear un solo sistema operativo para computadoras que podían dedicarse a un propósito, al otro o a ambos, puso en evidencia la problemática del trabajo en equipos de análisis, diseño e implantación de sistemas grandes.

Surgió también en la tercera generación de computadoras el concepto de la **multiprogramación**, porque debido al alto coste de las computadoras era necesario idear un esquema de trabajo que mantuviese a la unidad central de procesamiento más tiempo ocupada, así como la recogida *(spooling)* de trabajos para su lectura hacia los lugares libres de memoria o la escritura de resultados. Sin embargo, se puede afirmar que los sistemas durante la tercera generación siguieron siendo básicamente sistemas por lotes.

En la cuarta generación la electrónica avanzó hacia la integración a gran escala, pudiendo crear circuitos con miles de transistores en un centímetro cuadrado de sílice y ya fue posible hablar de las computadoras personales y las estaciones de trabajo. Surgen los conceptos de **interfaces amigables** intentando así atraer al público en general al uso de las computadoras como herramientas cotidianas. Se hacen populares el **MS-DOS** y **UNIX** en estas máquinas. También es común encontrar clones de computadoras personales y una multitud de empresas pequeñas ensamblándolas por todo el mundo.

A mediados de los años 80, comienza el auge de las redes de computadoras y la necesidad de sistemas operativos en red y sistemas operativos distribuidos. La red mundial **Internet** se va haciendo accesible a toda clase de instituciones y se comienzan a dar muchas soluciones (y problemas) al querer hacer convivir recursos residentes en computadoras con sistemas operativos diferentes. En los años 90, el paradigma de la programación orientada a objetos cobra auge, así como el manejo de objetos desde los sistemas operativos. Las aplicaciones intentan crearse para ser ejecutadas en una plataforma específica y poder ver sus resultados en la pantalla o monitor de otra diferente (por ejemplo, ejecutar una simulación en una máquina con UNIX y ver los resultados en otra con DOS). Los niveles de interacción se van haciendo cada vez más profundos.

1.2. Evolución de los sistemas operativos

La evolución de los sistemas operativos se puede resumir en los siguientes hitos:

— **Sistemas de procesamiento en serie.** Todavía no existían los sistemas operativos, por lo que el programador tenía que estar interactuando directamente con el hardware. La entrada se realizaba mediante tarjetas perforadas y la salida por impresora.

— **Sistemas de procesamiento por lotes.** Estos sistemas permitieron que se preparase con antelación una serie de tareas o programas que el hardware ejecutaría después. El programa monitor era el encargado de ejecutar estas tareas de forma secuencial, desde la primera a la última.

— **Sistemas de procesamiento por lotes con multiprogramación.** El programa monitor del apartado anterior tenía el inconveniente de no aprovechar de forma óptima los recursos de la máquina. Para solucionarlo, se le incluyó al programa monitor la posibilidad de ejecutar varias tareas de forma simultánea, para que la CPU estuviese ocupada el mayor tiempo posible. La multiprogramación hizo que el programa monitor aumentase en complejidad, incluyendo, entre otras, características de planificación para elegir qué proceso tenía que ejecutarse en cada momento. El programa monitor ya se había convertido en un sistema operativo.

— **Sistemas de tiempo compartido.** Además de incluir multiprogramación y posibilidad de programar tareas, permitieron que el usuario pudiera interactuar con el sistema operativo: escribir una orden y obtener al momento la respuesta. CTSS (*Compatible Time-Sharing System*, sistema de tiempo compartido compatible) se considera el primer sistema de este tipo.

2. Concepto y funciones

2.1. Concepto

Los sistemas operativos existen porque son una manera razonable de solucionar el problema de crear un sistema de computación utilizable. Mirándolo desde diversas perspectivas un sistema operativo debe ser:

— Un programa que realiza funciones de intérprete entre el usuario de un ordenador y el hardware del mismo, ofreciendo un entorno más o menos amigable para que el usuario pueda ejecutar programas. Sus objetivos son facilitar el uso del sistema informático de forma eficiente.

— También podemos hablar del sistema operativo como un asignador de recursos. Un sistema informático tiene muchos recursos (subsistemas físico y lógico). El sistema operativo actúa gestionando estos recursos y los asigna a los programas en ejecución y, por tanto, a los usuarios, según las necesidades de sus respectivas tareas: decide a qué solicitudes se asignan los recursos para que el sistema informático opere de forma adecuada.

— Controla todos los recursos de la computadora y proporciona la base sobre la cual pueden ejecutarse los programas de aplicación. Logra que el sistema de computación se use de manera cómoda, y que el hardware del computador se emplee de manera eficiente.

— Sirve de base para que se puedan ejecutar los distintos programas de aplicación. Si no hay sistema operativo, no se puede ejecutar ningún programa de aplicación (y no puede funcionar el ordenador).

— Sirve de intérprete entre el lenguaje de la máquina y el humano. Gobierna internamente la máquina. Se encarga de brindar al usuario una forma amigable y sencilla de operar, interpretar, codificar y emitir las órdenes al procesador central para que Este realice las tareas necesarias y específicas para una orden.

— Permite a un grupo de usuarios compartir una instalación de computadora eficazmente.

— Amplía el potencial y la utilidad global del sistema, completando el hardware disponible con ciertas funciones nuevas o más potentes, como por ejemplo la carga y descarga automática de programas en función del espacio de memoria disponible, la gestión de los distintos periféricos, el control de la ejecución automática de los programas, con detección automática de determinados tipos de errores, el análisis de los recursos utilizados por los distintos programas, no solo por motivos contables, sino también para facilitar su acceso bajo condiciones controladas, etc. El sistema operativo también mantiene la comunicación con el operador del sistema, tanto para tenerlo informado de los trabajos en curso como para pedir, si es necesario, su intervención.

Los sistemas operativos han evolucionado con el tiempo al igual que lo ha hecho el hardware. Si los sistemas operativos se diseñaron para facilitar el manejo del hardware, algunas mejoras introducidas en este permitieron el desarrollo de mejores sistemas operativos. Este es el caso de las interfaces con las que el usuario puede trabajar delante de un ordenador de las que existen dos tipos:

— **Interfaz de línea de comandos.** La forma de interfaz entre el sistema operativo y el usuario en la que este escribe los comandos utilizando un lenguaje de comandos especial. Los sistemas con interfaces de

líneas de comandos se consideran más difíciles de aprender y utilizar que los de las interfaces gráficas. Sin embargo, los sistemas basados en comandos son por lo general programables, lo que les otorga una flexibilidad que no tienen los sistemas basados en gráficos carentes de una interfaz de programación.

— **Interfaz Gráfica del Usuario (GUI).** Es el tipo de visualización que permite al usuario elegir comandos, iniciar programas y ver listas de archivos y otras opciones utilizando las representaciones visuales (iconos) y las listas de elementos del menú. Las selecciones pueden activarse a través del teclado o ratón. Para los autores de aplicaciones, las interfaces gráficas de usuario ofrecen un entorno que se encarga de la comunicación con el ordenador.

2.2. Funciones de los sistemas operativos

Aunque existan en el mercado diversos sistemas operativos, todos ellos comparten las siguientes funciones:

— Proporcionar ya sea una interfaz de línea de comando o una interfaz gráfica, para que el usuario se pueda comunicar con la computadora.

— Control de recursos. Administrar los dispositivos de hardware en la computadora. El sistema operativo sirve de intermediario entre los programas y el hardware.

— Administrar y mantener los sistemas de archivo de disco. Los sistemas operativos agrupan la información dentro de compartimentos lógicos (llamados archivos) para almacenarlos en el disco. El sistema operativo mantiene una lista de los archivos en un disco, y nos proporciona las herramientas necesarias para organizar y manipular estos archivos.

— Apoyar a otros programas. Otra de las funciones importantes del sistema operativo es proporcionar servicios a otros programas. Cuando los programadores escriben programas de computadora, incluyen en sus programas instrucciones que solicitan los servicios del sistema operativo. Estas instrucciones son conocidas como “llamadas del sistema”.

— Interpretar los comandos que permiten al usuario comunicarse con el ordenador.

— Servir de base para la creación del software, logrando que equipos de marcas distintas funcionen de manera análoga, salvando las diferencias existentes entre ambos.

— Configurar el entorno para el uso del software y los periféricos. Dependiendo del tipo de máquina que se emplea, debe establecerse en forma lógica la disposición y características del equipo.

— Aceptar todos los trabajos y conservarlos hasta su finalización.

— Manejo de dispositivos de E/S: organiza los archivos en diversos dispositivos de almacenamiento, como discos flexibles, discos duros, discos compactos o cintas magnéticas.

— Manejo de errores: gestiona los errores de hardware y la pérdida de datos.

— Secuencia de tareas: el sistema operativo debe administrar la manera en que se reparten los procesos y definir el orden.

— Protección: evitar que las acciones de un usuario afecten el trabajo que está realizando otro usuario.

— Multiacceso: un usuario se puede conectar a otra máquina sin tener que estar cerca de ella.

— Eficiencia: un sistema operativo permite que los recursos de la computadora se usen de la manera más eficiente posible.

— Habilidad para evolucionar: un sistema operativo deberá construirse de manera que permita el desarrollo, prueba o introducción efectiva de nuevas funciones del sistema sin interferir con el servicio.

— Organizar datos para acceso rápido y seguro.

— Manejar las comunicaciones en red. El Sistema Operativo permite al usuario manejar con facilidad todo lo referente a la instalación y uso de las redes de computadoras.

— Procesamiento por bytes de flujo a través del bus de datos.

3. Clasificaciones

Los sistemas operativos los podremos clasificar por: su **estructura** (visión interna), los **servicios** que ofrecen y, finalmente, por la **forma** en que ofrecen sus servicios (visión externa).

3.1. Sistemas operativos por su estructura

Se deben observar dos tipos de requisitos cuando se construye un sistema operativo:

— **Requisitos de usuario:** sistema fácil de usar y de aprender, seguro, rápido y adecuado al uso al que se le quiere destinar.

— **Requisitos del software:** se engloban aspectos como el mantenimiento, forma de operación, restricciones de uso, eficiencia, tolerancia frente a los errores y flexibilidad.

A continuación se describen las distintas estructuras que presentan los actuales sistemas operativos para satisfacer las necesidades que de ellos se quieren obtener.

3.1.1. Estructura monolítica

Es la estructura de los primeros sistemas operativos constituidos fundamentalmente por un solo programa compuesto de un conjunto de rutinas entrelazadas, de tal forma que cada una puede llamar a cualquier otra. Las características fundamentales de este tipo de estructura son:

— Construcción del programa final a base de módulos compilados separadamente que se unen a través del enlazado.

— Buena definición de parámetros de enlace entre las distintas rutinas existentes, que puede provocar mucho acoplamiento.

— Carencia de protecciones y privilegios al entrar a rutinas que manejan diferentes aspectos de los recursos de la computadora, como memoria, disco, etc.

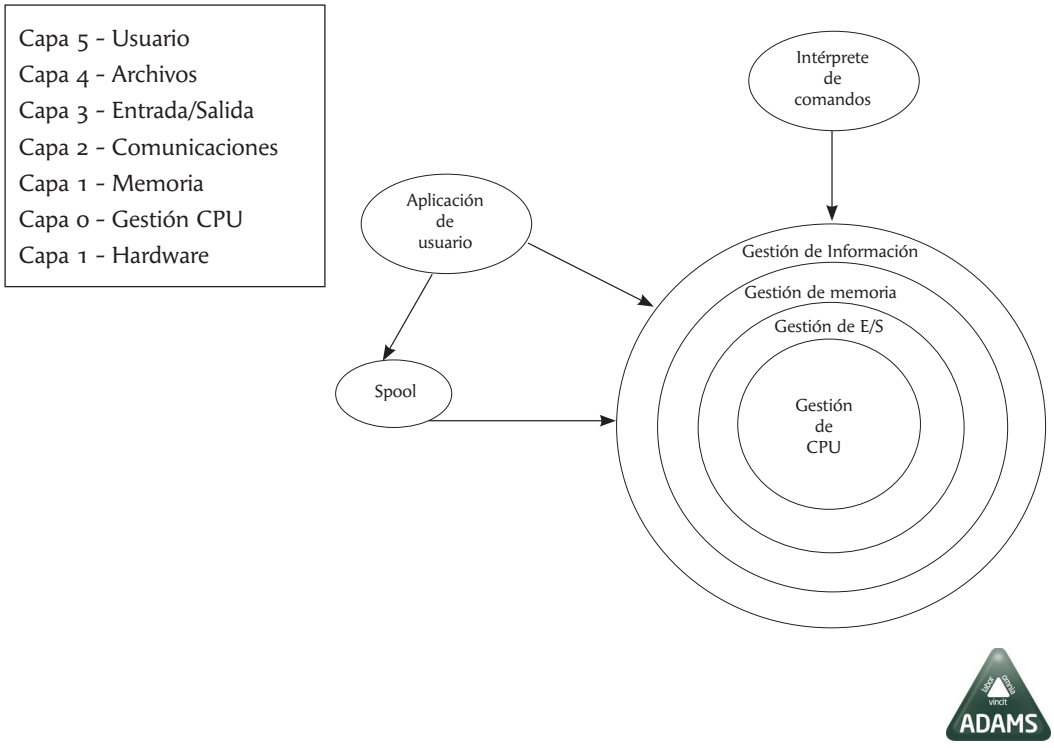
— Generalmente están hechos a medida, por lo que son eficientes y rápidos en su ejecución y gestión, pero por lo mismo carecen de flexibilidad para soportar diferentes ambientes de trabajo o tipos de aplicaciones.

3.1.2. Estructura jerárquica

A medida que fueron creciendo las necesidades de los usuarios y se perfeccionaron los sistemas, se hizo necesaria una mayor organización del software, del sistema operativo, donde una parte del sistema contenía subpartes estando organizado en forma de niveles.

Se dividió el sistema operativo en pequeñas partes, de tal forma que cada una de ellas estuviera perfectamente definida y con un claro interface con el resto de elementos.

Se constituyó una estructura jerárquica o de niveles en los sistemas operativos, el primero de los cuales fue denominado THE (*Technische Hogeschool*, Eindhoven), de Dijkstra, que se utilizó con fines didácticos (Ver Figura derecha de anillos concéntricos). Se puede pensar también en estos sistemas como si fueran "multicapa". Multics y Unix caen en esa categoría (Ver Figura de la izquierda en recuadro).



En la estructura anterior se basan prácticamente la mayoría de los sistemas operativos actuales. Otra forma de ver este tipo de sistema es la denominada de anillos concéntricos o “rings”.

Cada anillo en el sistema tiene una apertura conocida como puerta o trampa (trap), por donde pueden entrar las llamadas de las capas inferiores. De esta forma, las zonas más internas del sistema operativo o núcleo del sistema estarán más protegidas de accesos indeseados desde las capas más externas. Las más internas serán, por tanto, más privilegiadas que las externas.

3.1.3. Máquina virtual

Se trata de un tipo de sistemas operativos que presenta una interfaz a cada proceso, mostrando una máquina que parece idéntica a la máquina real subyacente. Estos sistemas operativos separan dos conceptos que suelen estar unidos en el resto de sistemas: la **multiprogramación** y la **máquina extendida**. El objetivo de los sistemas operativos de máquina virtual es el de integrar distintos sistemas operativos dando la sensación de ser varias máquinas diferentes.

El núcleo de estos sistemas operativos se denomina **monitor virtual** y tiene como misión llevar a cabo la multiprogramación, presentando a los niveles superiores tantas máquinas virtuales como se soliciten. Estas máquinas virtuales no son máquinas extendidas, sino una réplica de la máquina real, de manera que en cada una de ellas se pueda ejecutar un sistema operativo diferente, que será el que ofrezca la máquina extendida al usuario.

3.1.4. Cliente/Servidor *(microkernel)*

El tipo más reciente de sistemas operativos es el denominado cliente/servidor, que puede ser ejecutado en la mayoría de las computadoras, ya sean grandes o pequeñas.

El núcleo tiene como misión establecer la comunicación entre los clientes y los servidores. Los procesos pueden ser tanto servidores como clientes. Por ejemplo, un programa de aplicación normal es un cliente que llama al servidor correspondiente para acceder a un archivo o realizar una operación de entrada/salida sobre un dispositivo concreto. A su vez, un proceso cliente puede actuar como servidor para otro.

Este paradigma ofrece gran flexibilidad en cuanto a los servicios posibles en el sistema final, ya que el núcleo *(microkernel)* provee solamente funciones muy básicas de memoria, entrada/salida, archivos y procesos, dejando a los servidores proveer la mayoría de las funciones que el usuario final o programador puede usar. Estos servidores deben tener mecanismos de seguridad y protección que, a su vez, serán filtrados por el núcleo que controla el hardware.

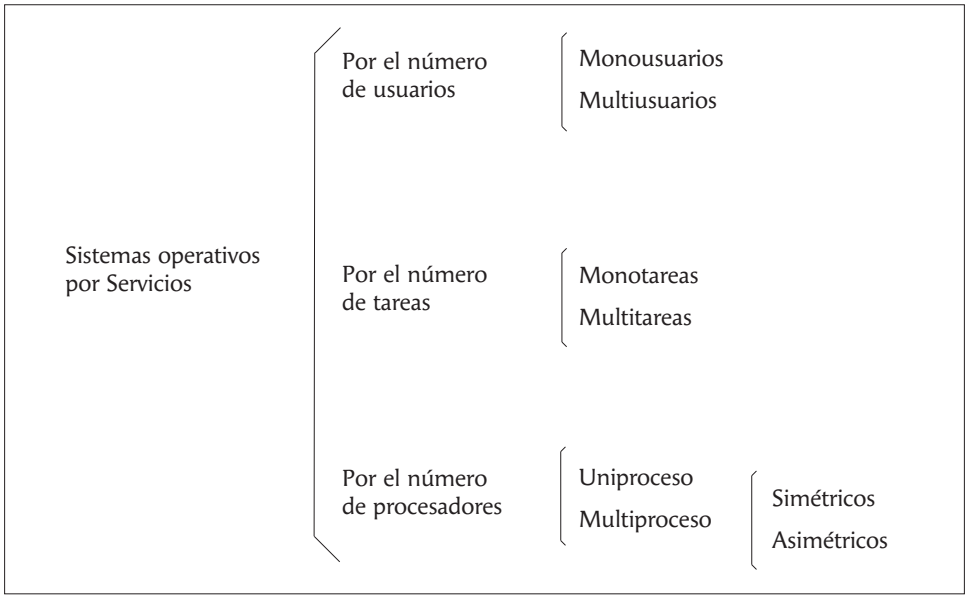
3.1.5. Híbrido

Básicamente es un sistema operativo con un núcleo que tiene una arquitectura que trata de combinar aspectos y beneficios de las arquitecturas de los sistemas operativos microkernel y kernel monolítico.

La idea detrás de un núcleo híbrido es tener una estructura similar a la que tiene un microkernel, pero implementar dicha estructura según un kernel monolítico. A diferencia de un microkernel, en un núcleo híbrido todos los servicios del sistema operativo se encuentran en el espacio del núcleo, así que no cuenta con ningún beneficio del microkernel, que tiene los servicios en espacio de usuario. Sin embargo, cuenta con los beneficios de los núcleos monolíticos, ya que no hay sobrecarga de rendimiento en el intercambio de mensajes y el cambio de contexto entre el núcleo y el modo de usuario, cosa que ocurre siempre en el caso de los microkernel.

3.2. Sistemas operativos por servicios

Esta clasificación es la más comúnmente usada y conocida desde el punto de vista del usuario final. Esta clasificación se comprende fácilmente con el esquema que a continuación se muestra en la figura.



3.2.1. Monousuarios

Los sistemas operativos monousuarios son aquellos que soportan a un usuario a la vez, sin importar el número de procesadores que tenga la computadora o el número de procesos o tareas que el usuario pueda ejecutar en un mismo instante de tiempo.

3.2.2. Multiusuarios

Los sistemas operativos multiusuarios son capaces de dar servicio a más de un usuario a la vez, ya sea por medio de varias terminales conectadas a la

computadora o por medio de sesiones remotas en una red de comunicaciones. No importa el número de procesadores en la máquina ni el número de procesos que cada usuario puede ejecutar simultáneamente.

3.2.3. Monotareas

Los sistemas monotarea son aquellos que solo permiten una tarea a la vez por usuario. Puede darse el caso de un sistema multiusuario y monotarea, en el cual se admiten varios usuarios al mismo tiempo pero cada uno de ellos puede estar haciendo solo una tarea a la vez.

3.2.4. Multitareas

Un sistema operativo multitarea o con multiprogramación es aquel que le permite al usuario estar realizando varias labores al mismo tiempo. Por ejemplo, puede estar editando el código fuente de un programa durante su depuración mientras compila otro programa, a la vez que está recibiendo correo electrónico en un proceso en background. Es común encontrar en ellos interfaces gráficas orientadas al uso de menús y el ratón, lo cual permite un rápido intercambio entre las tareas para el usuario, mejorando su productividad.

3.2.5. Uniproceso

Un sistema operativo uniproceso es aquel que es capaz de manejar solamente un procesador de la computadora, de manera que si la computadora tuviese más de uno le sería inútil.

3.2.6. Multiproceso

Un sistema operativo multiproceso (multiprocesador o con multiprocesamiento) se refiere al número de procesadores del sistema, que es más de uno y este es capaz de usarlos todos para distribuir su carga de trabajo. Generalmente estos sistemas trabajan de dos formas: simétrica o asimétricamente. Cuando se trabaja de manera asimétrica, el sistema operativo selecciona a uno de los procesadores el cual jugará el papel de procesador maestro y servirá como pivote para distribuir la carga a los demás procesadores, que reciben el nombre de esclavos. Cuando se trabaja de manera simétrica, los procesos, o partes de ellos (hilos), son enviados indistintamente a cualquiera de los procesadores disponibles teniendo, teóricamente, una mejor distribución y equilibrio en la carga de trabajo bajo este esquema.

3.3. Sistemas operativos por la forma de ofrecer sus servicios

Esta clasificación también se refiere a una visión externa, que en este caso es la del usuario, es decir, cómo acceder a los servicios. Bajo esta clasificación se pueden detectar dos tipos principales: sistemas operativos de red y sistemas operativos distribuidos.

3.3.1. Sistemas operativos de red

Los sistemas operativos de red se definen como aquellos que tienen la capacidad de interactuar con sistemas operativos en otras computadoras a través de un medio de transmisión con el objeto de intercambiar información, transferir archivos, ejecutar comandos remotos y un sin fin de otras actividades. El punto crucial de estos sistemas es que el usuario debe saber la sintaxis de un conjunto de comandos o llamadas al sistema para ejecutar estas operaciones, además de la ubicación de los recursos a los que se desee acceder.

3.3.2. Sistemas operativos distribuidos

Los sistemas operativos distribuidos abarcan los servicios de red, logrando integrar recursos (impresoras, unidades de respaldo, memoria, procesos, unidades centrales de proceso) en una sola máquina virtual a la que el usuario accede de forma transparente. Es decir, ahora el usuario ya no necesita saber la ubicación de los recursos, sino que los conoce por nombre y simplemente los usa como si todos ellos fuesen locales a su lugar de trabajo habitual.

Todo lo anterior es el marco teórico de lo que se desearía tener como sistema operativo distribuido, pero en la realidad no se ha conseguido crear uno del todo, por la complejidad que supone: distribuir los procesos en varias unidades de procesamiento, reintegrar sub-resultados, resolver problemas de concurrencia y paralelismo, recuperarse de fallas de algunos recursos distribuidos y consolidar la protección y seguridad entre los diferentes componentes del sistema y los usuarios.

Los avances tecnológicos en las redes de área local y la creación de microprocesadores de 32 y 64 bits lograron que computadoras más o menos baratas tuvieran el suficiente poder en forma autónoma para desafiar en cierto grado a los mainframes, y a la vez se dió la posibilidad de intercomunicarlas, surgiendo la oportunidad de partir procesos muy pesados en cálculo en unidades más pequeñas y distribuirlas en varios microprocesadores para luego reunir los sub-resultados, creando así una máquina virtual en la red que excediera en poder a un mainframe.

El sistema integrador de los microprocesadores que hace ver a las memorias, procesadores y todos los demás recursos como una sola entidad en forma transparente, se le llama sistema operativo distribuido. Las dos razones principales para crear o adoptar sistemas distribuidos son: por necesidad (debido a que los problemas a resolver son inherentemente distribuidos) o porque se desea tener más confiabilidad y disponibilidad de recursos. En el primer caso tenemos, por ejemplo, el control de los cajeros automáticos de la red bancaria. Ahí no es posible ni eficiente mantener un control centralizado, es más, no existe capacidad de cómputo y de entrada/salida para dar servicio a los millones de operaciones por minuto. En el segundo caso, supóngase que se tienen en una gran empresa varios grupos de trabajo, cada uno necesita almacenar grandes cantidades de información en disco duro con una alta confiabilidad y disponibilidad. La solución puede ser que para cada grupo de trabajo se asigne una partición de disco duro en servidores diferentes, de manera que si uno de los servidores falla, no se deje de dar el servicio a todos, sino solo a unos cuantos y, más aún, se podría tener un sistema con discos en

espejo (mirror) a través de la red, de manera que si un servidor se cae, el servidor en espejo continúa trabajando y el usuario no llega a advertir estos fallos, es decir, obtiene acceso a recursos en forma transparente.

A) Ventajas de los sistemas distribuidos

En general, los sistemas distribuidos (no solamente los sistemas operativos) tienen algunas ventajas sobre los sistemas centralizados, que se describen a continuación:

a) **Economía**: el cociente precio/desempeño de la suma del poder de los procesadores separados contra el poder de uno solo centralizado es mejor cuando están distribuidos.

b) **Velocidad**: relacionado con el punto anterior, la velocidad sumada es muy superior.

c) **Confiabilidad**: si una sola máquina falla, el sistema total sigue funcionando.

d) **Crecimiento**: el poder total del sistema puede irse incrementando al añadir pequeños sistemas, lo cual es mucho más difícil en un sistema centralizado y caro.

e) **Distribución**: algunas aplicaciones requieren de por sí una distribución física.

Por otro lado, los sistemas distribuidos también muestran algunas ventajas sobre sistemas aislados. Estas ventajas son:

— **Compartir datos:** un sistema distribuido permite compartir datos más fácilmente que los sistemas aislados, que tendrían que duplicarlos en cada nodo para lograrlo.

— **Compartir dispositivos**: un sistema distribuido permite el acceso a dispositivos desde cualquier nodo en forma transparente, lo cual es imposible con los sistemas aislados. El sistema distribuido logra un efecto sinergético.

— **Comunicaciones**: la comunicación persona a persona es factible en los sistemas distribuidos, en los sistemas aislados no.

— **Flexibilidad**: la distribución de las cargas de trabajo es factible en los sistemas distribuidos; se puede incrementar el poder de cómputo.

B) Desventajas de los sistemas distribuidos

Así como los sistemas distribuidos tienen grandes ventajas, también se pueden identificar algunas desventajas, algunas de ellas tan serias que han frenado la producción comercial de sistemas operativos en la actualidad. El problema más importante en la creación de sistemas distribuidos es el software: los problemas de compartición de datos y recursos son tan complejos que los mecanismos de solución generan mucha sobrecarga al sistema haciéndolo ineficiente. El comprobar, por ejemplo, quiénes tienen acceso a algunos recursos y quiénes no, el aplicar los mecanismos de protección y registro de permisos consume demasiados recursos.

Otros problemas de los sistemas operativos distribuidos surgen debido a la **concurrencia** y al **paralelismo**. Tradicionalmente las aplicaciones son creadas para computadoras que se ejecutan secuencialmente, de manera que el identificar secciones de código "paralelizable" es un trabajo arduo, pero necesario para dividir un proceso grande en sub-procesos y enviarlos a diferentes unidades de procesamiento para lograr la distribución. Con la concurrencia se deben implantar mecanismos para evitar las condiciones de competencia, las postergaciones indefinidas, el ocupar un recurso y estar esperando otro, las condiciones de espera circulares y, finalmente, los interbloqueos *(deadlocks)*. Estos problemas de por sí se presentan en los sistemas operativos multiusuarios o multitareas, y su tratamiento en los sistemas distribuidos es aún más complejo, y por tanto, necesitaría de algoritmos más complejos con la inherente sobrecarga esperada.

4. Trabajos, procesos e hilos

4.1. Procesos

Un proceso se define como la imagen de un programa en ejecución. Consta del programa ejecutable, los datos y el contexto de ejecución.

El concepto de proceso es más amplio que el de programa puesto que no solo incluye el código sino también el estado de su ejecución. El sistema operativo ve a los programas que se están ejecutando en un momento dado como procesos. A cada uno de ellos le asignará unos recursos determinados (memoria, CPU y entrada/salida).

Para registrar los distintos procesos que hay en ejecución, el sistema operativo utiliza una tabla de procesos donde registra, entre otra información: el identificador del proceso (PID), su proceso padre (PPID), memoria ocupada, usuario, prioridad, estado, comando que inició el proceso y el tiempo en ejecución.

En los sistemas Unix se puede consultar esta tabla de procesos con la orden **ps**. También se puede utilizar el comando **top** para obtener un listado dinámico ordenado por el uso de CPU en cada instante de los distintos procesos. En Windows, el Administrador de tareas muestra dinámicamente para cada proceso, su nombre, usuario, uso de CPU y uso de memoria.

4.1.1. Jerarquía de Procesos

El primer proceso en cargarse en un sistema Unix es el proceso planificador o swapper cuyo identificador de proceso (PID) es el número 0. A continuación se cargará el proceso conocido como **init**, que tiene como identificador de proceso (PID) el número 1. El proceso init a partir de aquí, será el padre de todos los demás procesos, como es el caso, por ejemplo, de las interfaces de comandos *(shells)* de los usuarios. El proceso **init** se encarga de arrancar los llamados *daemons* (demonios), que son procesos (servidores) que se mantienen siempre en ejecución, salvo que se detengan.

Según vemos, existe una jerarquía de procesos comenzando desde el proceso de inicialización. Para la creación de procesos hijo, Unix proporciona la llamada al sistema **fork**. Se traduce como "tenedor" puesto que la invocación

de esta llamada al sistema por parte de un proceso genera que el proceso se duplique en dos, siendo uno el padre y el otro el hijo, y continuando ambos su ejecución en paralelo. Un proceso hijo, cuando finaliza, le envía una señal a su proceso padre (el hijo debería finalizar antes que el padre), el cual le debe esperar mediante la llamada al sistema **wait**.

Un **proceso zombie** es un proceso que ha finalizado, liberando todos sus recursos, pero que todavía aparece en la tabla de procesos. Esta situación no deseable puede producirse por un error del proceso padre que no ha generado el wait correspondiente para ese hijo.

Un **proceso huérfano,** en cambio, se produce cuando el padre finaliza antes que el hijo. Cuando se produce esta situación, el proceso hijo pasa a ser adoptado por el proceso init (el PPID del proceso huérfano pasa a ser 1).

4.1.2. Estado de los procesos

Los procesos desde su creación hasta su finalización pasan por distintos estados, si bien existen nueve estados distintos, los podemos resumir en tres:

— En ejecución: el proceso está haciendo uso de la CPU avanzando en su ejecución.

— Listo: el proceso está preparado para pasar a ejecutarse en la CPU, pero actualmente se encuentra detenido.

— Bloqueado: el proceso se encuentra a la espera de algún evento, como puede ser que se le asigne permiso de lectura/escritura sobre algún soporte de almacenamiento.

Una vez enumerados estos estados, las transiciones entre ambos que se pueden dar son las siguientes:

— De Ejecución a Listo: esto quiere decir que el proceso ha consumido su tiempo de CPU y tienen que volver a esperar su turno.

— De Listo a Ejecución: el proceso ha esperado su turno y ya le ha llegado.

— De Ejecución a Bloqueado: el proceso se estaba ejecutando pero se ha visto bloqueado por algún evento por el que debe esperar, por tanto, cede la CPU al siguiente proceso a ejecutar quedando en estado de "Bloqueado".

— De Bloqueado a Listo: en este caso el proceso ya ha conseguido ese recurso o permisos por el que se vio bloqueado, de modo que pasa a la lista de procesos que están listos para ser elegidos a ocupar la CPU y ejecutarse.

4.2. Trabajos e hilos *(threads)*

El conjunto de uno o más procesos relacionados se denomina **trabajo**. Un trabajo se corresponde con un comando o programa que el usuario ha invocado.

El comando **jobs** muestra los trabajos (tareas) en ejecución. A cada uno le asigna un número comenzando por %1. Se pueden utilizar los comandos **fg** *(foreground)* y **bg** *(background)* para llevar un trabajo a primer plano o a segundo plano, respectivamente.

Para finalizar un trabajo o un proceso se utiliza la orden **kill**. En realidad, esta orden permite enviar distintos tipos de señales, aunque la más usual es la que provoca la finalización repentina de un proceso: **kill -9 pid_proceso.**

Un **hilo** ***(thread)*** es un trozo o sección de un proceso que tiene sus propios registros, pila y contador de programa, pero que comparte memoria con el resto de hilos de un mismo proceso. Son habituales los procesos de un solo hilo.

Debido a que cada hilo es un trozo de código distinto, observemos que no hay más posibilidad que cada hilo disponga de su contador de programa (contiene la dirección de la próxima instrucción a ejecutarse de ese código). Lo mismo ocurre con los registros de la CPU y el contenido de la pila.

Sin embargo y debido a que los recursos se asignan a procesos (y no a hilos), todos los hilos de un mismo proceso los comparten. En particular, todos los hilos de un proceso comparten el mismo espacio de memoria.

Se dice que los hilos de un mismo proceso cooperan entre sí, mientras los procesos compiten entre sí por el uso de los recursos. Nótese que los procesos pueden pertenecer incluso a usuarios distintos y que todos desean acaparar la mayor cantidad de recursos (los que le permita el sistema operativo).

5. Gestión de recursos y planificación

Una de las funciones más importantes de un sistema operativo es la de administrar los procesos y recursos. En esta sección se revisarán dos temas relativos a esta función: la **planificación del procesador** y los **problemas de concurrencia.**

5.1. Planificación del procesador

La planificación del procesador se refiere a la manera o técnicas que se usan para decidir cuánto tiempo de ejecución y cuándo se asigna a cada proceso del sistema. Obviamente, si el sistema es monousuario y monotarea no hay mucho que decidir, pero en el resto de los sistemas esto es crucial para el buen funcionamiento del sistema. El módulo encargado de realizar la entrega de un proceso al procesador recibe el nombre de **dispatcher** o **despachador**.

5.1.1. Niveles de planificación

En los sistemas de planificación generalmente se identifican tres niveles: el alto, el medio y el bajo.

El nivel alto decide qué trabajos (conjunto de procesos) son candidatos a convertirse en procesos compitiendo por los recursos del sistema; el nivel intermedio decide qué procesos se suspenden o reanudan para lograr ciertas metas de rendimiento mientras que el planificador de bajo nivel es el que decide qué proceso, de los que ya están listos (y que en algún momento pasaron por los otros dos planificadores) es al que le toca ahora estar ejecutándose en la unidad central de procesamiento. Los planificadores más interesantes para su estudio son los de bajo nivel porque son los que finalmente eligen el proceso en ejecución.

5.1.2. Objetivos de la planificación

Una estrategia de planificación debe buscar que los procesos obtengan sus turnos de ejecución apropiadamente, conjuntamente con un buen rendimiento y minimización de la sobrecarga *(overhead)* del planificador mismo. En general, se buscan cinco objetivos principales:

— **Justicia o imparcialidad:** todos los procesos son tratados de la misma forma, y en algún momento obtienen su turno de ejecución o intervalos de tiempo de ejecución hasta su terminación exitosa.

— **Maximizar la productividad:** el sistema debe finalizar el mayor número de procesos por unidad de tiempo.

— **Maximizar el tiempo de respuesta:** cada usuario o proceso debe observar que el sistema les responde consistentemente a sus requerimientos.

— **Evitar el aplazamiento indefinido:** los procesos deben terminar en un plazo finito de tiempo.

— **El sistema debe ser predecible:** ante cargas de trabajo ligeras, el sistema debe responder rápido y con cargas pesadas debe ir degradándose paulatinamente. Otro punto de vista de esto es que si se ejecuta el mismo proceso en cargas similares de todo el sistema, la respuesta en todos los casos debe ser similar.

5.1.3. Características a considerar de los procesos

No todos los sistemas procesan el mismo tipo de trabajos, por lo que un algoritmo de planificación que en un sistema funciona excelentemente, puede originar un rendimiento pésimo en otro, cuyos procesos tienen características diferentes. Estas características pueden ser:

— **Cantidad de entrada/salida:** existen procesos que realizan una gran cantidad de operaciones de entrada y salida (aplicaciones de bases de datos, por ejemplo).

— **Cantidad de uso de CPU**: existen procesos que no realizan muchas operaciones de entrada y salida, sino que usan intensivamente la unidad central de procesamiento. Por ejemplo, operaciones con matrices.

— **Procesos de lote o interactivos**: un proceso de lote es más eficiente en cuanto a la lectura de datos, ya que generalmente lo hace de archivos, mientras que un programa interactivo espera mucho tiempo

(no es lo mismo el tiempo de lectura de un archivo que la velocidad en que una persona teclea datos) por las respuestas de los usuarios.

— **Procesos en tiempo real**: si los procesos deben dar respuesta en tiempo real se requiere que tengan prioridad para los turnos de ejecución.

— **Longevidad de los procesos**: existen procesos que típicamente requerirán varias horas para finalizar su labor, mientras que existen otros que solo necesitan algunos segundos.

5.1.4. Planificación apropiativa o no apropiativa

La **planificación no apropiativa** es aquella en la cual, una vez que a un proceso le toca su turno de ejecución ya no puede ser suspendido por el sistema operativo, ya no se le puede arrebatar la unidad central de procesamiento. Este esquema puede ser peligroso, ya que si el proceso contiene accidental o deliberadamente ciclos infinitos, el resto de los procesos pueden quedar aplazados indefinidamente. Se utiliza en la llamada **multitarea colaborativa.**

Una **planificación apropiativa** *(preemptive)* es aquella en que existe un reloj que lanza interrupciones periódicas en las cuales el planificador toma el control y se decide si el mismo proceso seguirá ejecutándose o se le da su turno a otro proceso. Este mismo reloj puede servir para lanzar procesos manejados por el reloj del sistema. Por ejemplo en los sistemas UNIX existen los 'cronjobs' y 'atjobs', los cuales se programan en base a la hora, minuto, día del mes, día de la semana y día del año.

5.1.5. Asignación del turno de ejecución

Los algoritmos de la capa baja para asignar el turno de ejecución se describen a continuación:

— **Por prioridad:** los procesos de mayor prioridad se ejecutan primero. Si existen varios procesos de mayor prioridad que otros, pero entre ellos con la misma prioridad, pueden ejecutarse estos de acuerdo a su orden de llegada o por "round robin". La ventaja de este algoritmo es que es flexible en cuanto a permitir que ciertos procesos se ejecuten primero e, incluso, por más tiempo. Su desventaja es que puede provocar aplazamiento indefinido (inanición) en los procesos de baja prioridad. Por ejemplo, suponga que existen procesos de prioridad 20 y procesos de prioridad 10. Si durante todo el día terminan procesos de prioridad 20 al mismo ritmo que entran otros con esa prioridad, el efecto es que los de prioridad 10 estarán esperando por siempre. También provoca que el sistema sea impredecible para los procesos de baja prioridad.

— **El trabajo más corto primero:** es difícil de llevar a cabo porque se requiere saber o tener una estimación de cuánto tiempo necesita el proceso para terminar. Pero si se sabe, se ejecutan primero aquellos trabajos que necesitan menos tiempo y de esta manera se obtiene el mejor tiempo de respuesta promedio para todos los procesos. Por ejemplo, si llegan

5 procesos A, B, C, D y E cuyos tiempos de CPU son 26, 18, 24, 12 y 4 unidades de tiempo, se observa que el orden de ejecución será E, D, B, C y A (4, 12, 18, 24 y 26 unidades de tiempo respectivamente). En la tabla siguiente se muestra en qué unidad de tiempo comienza a ejecutarse cada proceso y cómo comenzaron todos a esperar desde la unidad cero, se obtiene el tiempo promedio de espera.

Proceso	Espera desde	Termina	Tiempo de Espera
A	0	4	4
B	0	16	16
C	0	34	34
D	0	58	58
E	0	84	84

Tiempo promedio = (4 + 16 + 34 + 58 + 84)/5 = 39 unidades.

— **El primero en llegar, primero en ejecutarse:** es muy simple, los procesos reciben su turno conforme llegan. La ventaja de este algoritmo es que es justo y no provoca aplazamiento indefinido. La desventaja es que no aprovecha ninguna característica de los procesos y puede no servir para un proceso de tiempo real. Por ejemplo, el tiempo promedio de respuesta puede ser muy malo comparado con el logrado por el del trabajo más corto primero. Retomando el mismo ejemplo que en el algoritmo anterior, obtenemos un tiempo de respuesta promedio (26+44+68+80+84)/5 = 60 unidades, el cual es muy superior a las 39 unidades que es el mejor tiempo posible.

— **Round Robin**: también llamada por turno, consiste en darle a cada proceso un intervalo de tiempo de ejecución (llamado *time slice*), y cada vez que se vence ese intervalo se copia el contexto del proceso a un lugar seguro y se le da su turno a otro proceso. Los procesos están ordenados en una cola circular. Por ejemplo, si existen tres procesos, el A, B y C, dos repasadas del planificador darían sus turnos a los procesos en el orden A, B, C, A, B, C. La ventaja de este algoritmo es su simplicidad, es justo y no provoca aplazamiento indefinido.

— **Selfish Round Robin o también conocida como "ronda egoísta":** es una variante del anterior. Este método busca favorecer los procesos que ya han pasado tiempo ejecutando en la CPU con respecto a los recién llegados mediante dos colas distintas. De hecho, los nuevos procesos no son programados directamente para su ejecución, sino que se les forma en la cola de procesos nuevos y se avanza únicamente con la cola de procesos aceptados.

— **El tiempo restante más corto:** es parecido al del trabajo más corto primero, pero aquí se está calculando en todo momento cuánto tiempo le resta para terminar a todos los procesos, incluyendo los nuevos, y a aquel que le quede menos tiempo para finalizar es escogido para ejecutarse. La ventaja es que es muy útil para sistemas de tiempo

compartido porque se acerca mucho al mejor tiempo de respuesta, además de responder dinámicamente a la longevidad de los procesos; su desventaja es que provoca más sobrecarga porque el algoritmo es más complejo.

— **La tasa de respuesta más alta:** este algoritmo concede el turno de ejecución al proceso que produzca el valor mayor de la siguiente fórmula:

$$\text{valor} = \frac{\text{tiempo que ha esperado} + \text{tiempo total para terminar}}{\text{tiempo total para terminar}}$$

Es decir, que dinámicamente el valor se va modificando y mejora un poco las deficiencias del algoritmo del trabajo más corto primero.

— **Por política:** una forma de asignar el turno de ejecución es por política, en la cual se establece algún reglamento específico que el planificador debe obedecer. Por ejemplo, una política podría ser que todos los procesos reciban el mismo tiempo de uso de CPU en cualquier momento. Esto significa, por ejemplo, que si existen 2 procesos y han recibido 20 unidades de tiempo cada uno (tiempo acumulado en *time slices* de 5 unidades) y en este momento entra un tercer proceso, el planificador le dará inmediatamente el turno de ejecución por 20 unidades de tiempo. Una vez que todos los procesos están "parejos" en uso de CPU, se les aplica *"round robin"*.

5.2. Problemas de concurrencia

En los sistemas de tiempo compartido (aquellos con varios usuarios, procesos, tareas, trabajos que reparten el uso de CPU entre estos) se presentan muchos problemas debido a que los procesos compiten por los recursos del sistema. Imagine que un proceso está escribiendo en la unidad de cinta y se le termina su turno de ejecución e inmediatamente después el proceso elegido para ejecutarse comienza a escribir sobre la misma cinta. El resultado es una cinta cuyo contenido es un desastre de datos mezclados. Así, en la cinta, existen una multitud de recursos cuyo acceso debe ser controlado para evitar los problemas de la concurrencia.

El sistema operativo debe ofrecer mecanismos para sincronizar la ejecución de procesos: semáforos, monitores, envío de mensajes, tuberías *(pipes)*, etc. Los semáforos son rutinas de software (que en su nivel más interno se auxilian del hardware) para lograr la exclusión mutua en el uso de recursos. Para entender este y otros mecanismos es importante entender los problemas generales de concurrencia, que describimos a continuación:

— **Condiciones de carrera o competencia:** la condición de carrera (*race condition*) ocurre cuando dos o más procesos acceden a un recurso compartido sin control, de manera que el resultado combinado de este acceso depende del orden de llegada. Suponga, por ejemplo, que dos clientes de un banco realizan cada uno una operación en cajeros diferentes al mismo tiempo.

El usuario A quiere hacer un depósito. El B, un retiro. El usuario A comienza la transacción y lee su saldo que es de 1.000 €. En ese momento pierde su turno de ejecución (y su saldo queda como 1.000 €) y el usuario B inicia el retiro: lee el saldo que es 1.000 €, retira 200 € y almacena el nuevo saldo que es de 800 y termina. El turno de ejecución regresa al usuario A que hace su depósito de 100, quedando saldo = saldo + 100 = 1.000 + 100 = 1.100 €. Como se ve, el retiro se perdió y eso le encanta a los usuarios A y B, pero al banquero no le convino esta transacción. El error pudo ser al revés, quedando el saldo final en 800 €.

Para evitar estas situaciones es necesario implementar mecanismos de exclusión mutua, que pueden ser mediante semáforos o monitores, para evitar que dos procesos accedan simultáneamente a recursos que pueden generar conflictos (secciones críticas).

— **Inanición o aplazamiento indefinido**: se produce cuando un proceso nunca recibe permiso para utilizar un recurso porque el algoritmo usado le asigna siempre el permiso a otros procesos. Las políticas de asignación de recursos y de planificación deben evitar que se pueda producir esta situación.

— **Condición de espera circular o interbloqueo** (*deadlock*): se produce, en un conjunto de procesos, cuando cada uno de ellos espera a un evento que solo puede generar otro del conjunto. Ninguno de los procesos avanza y todos se bloquean. Por ejemplo, suponga que el proceso A tiene asignado el recurso "cinta" y el proceso B tiene asignado el recurso 'disco'. En ese momento al proceso A se le ocurre pedir el recurso "disco" y al proceso B el recurso "cinta". Ahí se forma una espera circular entre esos dos procesos que se puede evitar quitándole a la fuerza, un recurso a cualquiera de los dos procesos.

— **Condición de no apropiación**: esta condición no resulta precisamente de la concurrencia, pero juega un papel importante en este ambiente. Esta condición especifica que si un proceso tiene asignado un recurso, dicho recurso no puede arrebatársele por ningún motivo, y estará disponible hasta que el proceso lo "suelte" por su voluntad.

— **Condición de espera ocupada**: esta condición consiste en que un proceso pide un recurso que ya está asignado a otro proceso y la condición de no apropiación se debe cumplir. Entonces el proceso estará gastando el resto de su *"time slice"* comprobando si el recurso fue liberado. Es decir, desperdicia su tiempo de ejecución en esperar. La solución más común a este problema consiste en que el sistema operativo se dé cuenta de esta situación y mande a una cola de espera al proceso, otorgándole inmediatamente el turno de ejecución a otro proceso.

— **Condición de exclusión mutua:** cuando un proceso usa un recurso del sistema realiza una serie de operaciones sobre el recurso y después lo deja de usar. A la sección de código que usa ese recurso se le llama «sección crítica».
La condición de exclusión mutua establece que solamente se permite a un proceso estar dentro de la misma sección crítica. Esto es, que en un momento dado, solamente un proceso puede usar un recurso

a la vez. Para lograr la exclusión mutua generalmente se usan algunas técnicas, como son: semáforos, monitores, algoritmo de Dekker y Peterson.

— **Condición de ocupar y esperar un recurso:** consiste en que un proceso pide un recurso y se le asigna. Antes de soltarlo, pide otro recurso que otro proceso ya tiene asignado.

6. Memoria

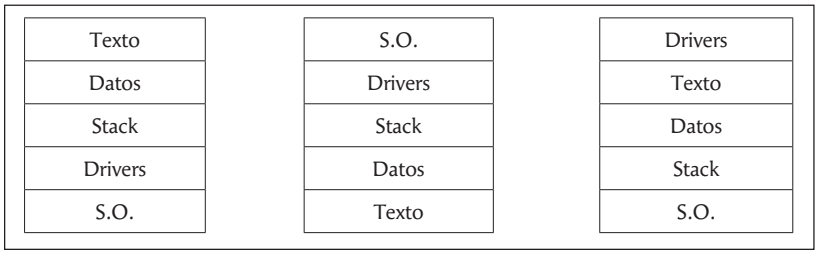
En esta sección se describirán las técnicas más usuales en el manejo de memoria, revisando los conceptos relevantes. Se abarcarán los esquemas de manejo simple de memoria real, la multiprogramación en memoria real con sus variantes, el concepto de *"overlays"*, la multiprogramación con intercambio y los esquemas de manejo de memoria virtual.

A continuación se describen los conceptos más importantes en cuanto a las técnicas empleadas en el manejo de memoria:

<table>
<tr><td>Real</td><td colspan="2">Real</td><td colspan="2">Virtual</td></tr>
<tr><td rowspan="4">Mono
usuario</td><td colspan="2">Multiprogramación</td><td colspan="2">Multiprogramación</td></tr>
<tr><td colspan="2" rowspan="2">Particiones</td><td>Páginación
Pura</td><td>Segmentación
Pura</td></tr>
<tr><td colspan="2">Combinación</td></tr>
<tr><td>Fijas</td><td>Variables</td><td colspan="2">Relocalización
Protección</td></tr>
</table>

6.1. Manejo de memoria en sistemas monousuario sin intercambio

Este esquema se usa principalmente en sistemas monousuario y monotarea, como son las computadoras personales con DOS. Bajo este esquema, la memoria real es tomada para almacenar el programa que se esté ejecutando en un momento dado, con la visible desventaja de que se está limitando a la cantidad de RAM disponible únicamente.



6.2. Multiprogramación en memoria real

En los años 60, las empresas e instituciones que habían invertido grandes sumas en la compra de equipo de cómputo se dieron cuenta rápidamente de

que los sistemas en lote invertían una gran cantidad de tiempo en operaciones de entrada y salida, donde la intervención de la unidad central de procesamiento era prácticamente nula, y se comenzaron a preguntar cómo hacer para que se mantuviera más tiempo ocupada. Fue así como nació el concepto de multiprogramación, que consiste en la idea de poner en la memoria física más de un proceso al mismo tiempo, de manera que si el que se estaba ejecutando en ese momento entraba en un período de entrada/salida, se podía tomar otro proceso para que usara la unidad central de procesamiento. De esta forma, la memoria física se dividía en secciones de tamaño suficiente para contener a varios programas.

Dentro del esquema de multiprogramación en memoria real surgieron dos problemas interesantes: la **protección** y la **relocalización**.

6.2.1. El problema de la relocalización

Este problema no es exclusivo de la multiprogramación en memoria real, sino que se presentó aquí pero se sigue presentando en los esquemas de memoria virtual también. Este problema consiste en que los programas que necesitan cargarse a memoria real ya están compilados y ligados, de manera que internamente contienen una serie de referencias a direcciones de instrucciones, rutinas y procedimientos que ya no son válidas en el espacio de direcciones de memoria real de la sección en la que se carga el programa. Esto es, cuando se compiló el programa se definieron o resolvieron las direcciones de memoria de acuerdo a la sección de ese momento, pero si el programa se carga otro día en una sección diferente, las direcciones reales ya no coinciden. En este caso, el manejador de memoria puede solucionar el problema de dos maneras: de manera estática o de manera dinámica.

La solución estática consiste en que todas las direcciones del programa se vuelvan a recalcular en el momento en que el programa se carga a memoria, esto es, prácticamente se vuelve a recompilar el programa.

La solución dinámica consiste en tener un registro que guarde la dirección base de la sección que va a contener al programa. Cada vez que el programa haga una referencia a una dirección de memoria, se le suma el registro base para encontrar la dirección real. Por ejemplo, suponga que el programa es cargado en una sección que comienza en la dirección 100. El programa hará referencias a las direcciones 50, 52, 54. Pero el contenido de esas direcciones no es el deseado, sino las direcciones 150, 152 y 154, ya que ahí comienza el programa. La suma de 100 + 50,...,etc., se hacen al tiempo de ejecución. La primera solución vale más la pena que la segunda si el programa contiene ciclos y es largo, ya que consumirá menos tiempo en la resolución inicial que la segunda solución en las resoluciones en línea.

6.2.2. El problema de la protección

Este problema se refiere a que, una vez que un programa ha sido cargado a memoria en algún segmento en particular, nada le impide al programador que intente direccionar (por error o deliberadamente) localidades de memoria menores que el límite inferior de su programa o superiores a la dirección

mayor; es decir, quiere referenciar localidades fuera de su espacio de direcciones. Obviamente, este es un problema de protección, ya que no es legal leer o escribir en áreas de otros programas.

La solución a este problema también puede ser el uso de un registro base y un registro límite. El registro base contiene la dirección del comienzo de la sección que contiene al programa, mientras que el límite contiene la dirección donde termina. Cada vez que el programa hace una referencia a memoria se comprueba si cae en el rango de los registros y si no es así se envía un mensaje de error y se aborta el programa.

6.2.3. Particiones fijas o particiones variables

En el esquema de la multiprogramación en memoria real se manejan dos alternativas para asignarle a cada programa su partición correspondiente: **particiones de tamaño fijo o particiones de tamaño variable**. La alternativa más simple son las particiones fijas.

Dichas particiones se crean cuando se enciende el equipo y permanecerán con los tamaños iniciales hasta que el equipo se apague. Es una alternativa muy vieja, quien hacía la división de particiones era el operador analizando los tamaños estimados de los trabajos de todo el día. Por ejemplo, si el sistema tenía 512 kilobytes de RAM, podía asignar 64 k para el sistema operativo, una partición más de 64 k, otra de 128 k y una mayor de 256 k. Esto era muy simple, pero inflexible, ya que si surgían trabajos urgentes, por ejemplo, de 400 k, tenían que esperar a otro día o reparticionar, inicializando el equipo desde cero.

La otra alternativa, que surgió después y como necesidad de mejorar la alternativa anterior, era crear particiones contiguas de tamaño variable. Para esto, el sistema tenía que mantener ya una estructura de datos suficiente para saber dónde había huecos disponibles de RAM y de dónde a dónde habían particiones ocupadas por programas en ejecución. Así, cuando un programa requería ser cargado a RAM, el sistema analizaba los huecos para saber si había alguno de tamaño suficiente para el programa que quería entrar, si era así, le asignaba el espacio. Si no, intentaba relocalizar los programas existentes con el propósito de hacer contiguo todo el espacio ocupado, así como todo el espacio libre y así obtener un hueco de tamaño suficiente. Si aún así el programa no cabía, entonces lo bloqueaba y tomaba otro. El proceso con el cual se juntan los huecos o los espacios ocupados se le llama **compactación**.

Sin embargo, surgen varios problemas con los esquemas de particiones fijas y particiones variables: ¿En base a qué criterio se elige el mejor tamaño de partición para un programa? Por ejemplo, si el sistema tiene dos huecos, uno de 18 k y otro de 24 k para un proceso que desea 20 k, ¿cuál se le asigna? Existen varios algoritmos para darle respuesta a la pregunta anterior, los cuales se enumeran y describen a continuación:

— **Primer Ajuste:** se asigna el primer hueco que sea mayor al tamaño deseado.

— **Mejor Ajuste:** se asigna el hueco cuyo tamaño exceda en la menor cantidad al tamaño deseado. Requiere de una búsqueda exhaustiva.

— **Peor Ajuste:** se asigna el hueco cuyo tamaño exceda en la mayor cantidad al tamaño deseado. Requiere también de una búsqueda exhaustiva.

— **El Siguiente Ajuste:** es igual que el "primer ajuste" con la diferencia que se deja un apuntador al lugar en donde se asignó el último hueco para realizar la siguiente búsqueda a partir de él.

— **Ajuste Rápido:** se mantienen listas ligadas separadas de acuerdo a los tamaños de los huecos, para así buscarle a los procesos un hueco más rápido en la cola correspondiente.

Otro problema que se vislumbra desde aquí es que, una vez asignado un hueco, por ejemplo, con "el peor ajuste", puede ser que el proceso requiriera 12 kilobytes y que el hueco asignado fuera de 64 kilobytes, por lo cual el proceso desperdiciaría una gran cantidad de memoria dentro de su partición. A esto se le llama **fragmentación interna.**

Por otro lado, conforme el sistema va avanzando en el día, finalizando procesos y comenzando otros, la memoria se va configurando como una secuencia contigua de huecos y de lugares asignados, provocando que existan huecos, por ejemplo, de 12 k, 28 k y 30 k, que sumados dan 70 k, pero que si en ese momento llega un proceso pidiéndolos, no se le pueden asignar ya que no son localidades contiguas de memoria (a menos que se realice la compactación). Al hecho de que aparezcan huecos no contiguos de memoria se le llama **fragmentación externa.**

De cualquier manera, la multiprogramación fue un avance significativo para el mejor aprovechamiento de la unidad central de procesamiento y de la memoria misma, así se dio pie a que surgieran los problemas de **asignación de memoria, protección** y **relocalización**, entre otros.

6.2.4. Los overlays

Una vez que surgió la multiprogramación, los usuarios comenzaron a explorar la forma de ejecutar grandes cantidades de código en áreas de memoria muy pequeñas, auxiliados por algunas llamadas al sistema operativo. Es así como nacen los **overlays** también conocidos como técnicas de solapamiento.

Esta técnica consiste en que el programador divide lógicamente un programa muy grande en secciones donde pueden almacenarse las particiones de RAM. Al final de cada sección del programa (o en otros lugares necesarios) el programador inserta una o varias llamadas al sistema con el fin de descargar la sección presente de RAM y cargar otra, que en ese momento residía en disco duro u otro medio de almacenamiento secundario. Aunque esta técnica era eficaz (porque resolvía el problema) no era eficiente (ya que no lo resolvía de la mejor manera). Esta solución requería que el programador tuviera un conocimiento muy profundo del equipo y de las llamadas al sistema operativo. Otra desventaja era la portabilidad de un sistema a otro: las llamadas cambiaban, los tamaños de particiones también. Resumiendo, con esta técnica se podían ejecutar programas más grandes que las particiones de RAM, donde la división del código corría a cuenta del programador y el control a cuenta del sistema operativo.

6.3. Multiprogramación en memoria virtual

La necesidad cada vez más imperiosa de ejecutar programas grandes y el crecimiento en capacidad de procesado de las unidades centrales empujaron a los diseñadores de los sistemas operativos a implantar un mecanismo para ejecutar automáticamente programas más grandes que la memoria real disponible, esto es, ofrecer "memoria virtual".

La **memoria virtual** se llama así porque el programador ve una cantidad de memoria mucho mayor que la real, y en realidad se trata de la suma de la memoria de almacenamiento primario y una cantidad determinada de almacenamiento secundario. El sistema operativo, en su módulo de manejo de memoria, se encarga de intercambiar programas enteros, segmentos o páginas entre la memoria real y el medio de almacenamiento secundario.

La memoria virtual se apoya en varias técnicas interesantes para lograr su objetivo. Una de las teorías más fuertes es la del **conjunto de trabajo**, la cual se refiere a que un programa o proceso no está usando todo su espacio de direcciones en todo momento, sino que existen un conjunto de localidades activas que conforman el "conjunto de trabajo". Si se logra que las páginas o segmentos que contienen al conjunto de trabajo estén siempre en RAM, entonces el programa se desempeñará muy bien.

Otro factor importante es el **principio de localidad**, lo cual quiere decir que algunos programas tienden a usar mucho las instrucciones que están próximas a la instrucción que se está ejecutando actualmente.

6.3.1. Paginación pura

La paginación pura en el manejo de memoria consiste en que el sistema operativo divide dinámicamente los programas en unidades de tamaño fijo (generalmente múltiplos de 1 kilobyte) que va a manipular de RAM a disco y viceversa. Al proceso de intercambiar páginas, segmentos o programas completos entre RAM y disco se le conoce como **intercambio** o ***swapping***. En la paginación, se debe cuidar el tamaño de las páginas, ya que si estas son muy pequeñas, el control por parte del sistema operativo para saber cuáles están en RAM y cuáles en disco, sus direcciones reales, etc., crece y provoca mucha "sobrecarga" *(overhead)*. Por otro lado, si las páginas son muy grandes, el *overhead* disminuye, pero entonces puede ocurrir que se desperdicie memoria en procesos pequeños. Debe haber un equilibrio.

Uno de los aspectos más importantes de la paginación, así como de cualquier esquema de memoria virtual, es la forma de traducir una **dirección virtual** a **dirección real**.

Cuando se está buscando una página cualquiera y esta no está cargada, surge lo que se llama un **fallo de página** *(page fault)*. Esto es caro para el controlador de memoria, ya que tiene que realizar una serie de pasos extra para poder resolver la dirección deseada y darle su contenido a quien lo pide. Primero, se detecta que la página no está presente y entonces se busca en la tabla la dirección de esta página en disco. Una vez localizada en disco se intenta cargar en alguna página libre de RAM. Si no hay páginas libres se tiene que escoger alguna para enviarla hacia el disco. Una vez escogida y enviada a disco, se marca su valor de control en la tabla de direcciones virtuales para indicar que

ya no está en esta memoria, mientras que la página deseada se carga en RAM y se marca su valor para indicar que ahora ya está en esta memoria RAM. Todo este procedimiento es caro, ya que se sabe que los accesos a disco duro son del orden de decenas de veces más lentos que en RAM.

Cuando se necesita descargar una página de RAM hacia un disco se debe hacer una elección. Para ello existen varios algoritmos, que se describen a continuación:

— **La primera en entrar, primera en salir (FIFO)**: se escoge la página que haya entrado primero y esté cargada en RAM. Se necesita que en los valores de control se guarde un dato de tiempo. No es eficiente porque no aprovecha ninguna característica de ningún sistema. Es justa e imparcial.

— **La no usada recientemente (NRU):** se escoge la página que no haya sido usada (referenciada) en el ciclo anterior. Pretende aprovechar el hecho de la localidad en el conjunto de trabajo.

— **La usada menos recientemente (LRU):** es parecida a la anterior, pero escoge la página que se usó hace más tiempo, pretendiendo que como ya tiene mucho sin usarse es muy probable que siga sin usarse en los próximos ciclos. Necesita de una búsqueda exhaustiva.

— **La no usada frecuentemente:** este algoritmo toma en cuenta no tanto el tiempo, sino el número de referencias. En este caso cualquier página que se use muy poco, menos que alguna otra.

— **La menos frecuentemente usada:** es parecida a la anterior, pero aquí se busca en forma exhaustiva aquella página que se ha usado menos que todas las demás.

— **En forma aleatoria:** elige cualquier página sin aprovechar nada. Es justa e imparcial, pero ineficiente.

Otro dato interesante de la paginación es que ya no se requiere que los programas estén ubicados en zonas de memoria adyacente, ya que las páginas pueden estar ubicadas en cualquier lugar de la memoria RAM.

6.3.2. Segmentación pura

La segmentación se aprovecha del hecho de que los programas se dividen en partes lógicas, como son las partes de datos, de código y de pila *(stack)*. La segmentación asigna particiones de memoria a cada segmento de un programa y busca como objetivos la facilidad, compartir segmentos (por ejemplo librerías compartidas) y el intercambio entre memoria y los medios de almacenamiento secundario.

6.3.3. Sistemas combinados

La paginación y la segmentación puras son métodos de manejo de memoria bastante efectivos, aunque la mayoría de los sistemas operativos modernos

implantan esquemas combinados, es decir, combinan la **paginación** y la **segmentación**. La idea de combinar estos esquemas se debe a que de esta forma se aprovechan los conceptos de la división lógica de los programas (segmentos) con la granularidad de las páginas. De esta manera, un proceso estará repartido en la memoria real en pequeñas unidades (páginas) cuyo enlace son los segmentos. También es factible así, el compartir segmentos a medida que las partes necesitadas de los mismos se van referenciando (páginas).

Un aspecto importante es la estrategia para cargar páginas (o segmentos) a la memoria RAM. Se usan más comúnmente dos estrategias: cargado de páginas por demanda y cargado de páginas anticipado.

La estrategia de **cargado por demanda** consiste en que las páginas solamente son llevadas a RAM si fueron solicitadas, es decir, si se hizo referencia a una dirección que cae dentro de ellas.

La **carga anticipada** consiste en tratar de adivinar qué páginas serán solicitadas en el futuro inmediato y cargarlas de antemano, para que cuando se pidan ya no ocurran fallos de página. Ese "adivinar" puede ser que se aproveche en el fenómeno de localidad y que las páginas que se cargan por anticipado sean aquellas que contienen direcciones contiguas a la dirección que se acaba de referenciar. De hecho, el sistema operativo VMS usa un esquema combinado para cargar páginas: cuando se hace referencia a una dirección cuya página no está en RAM, se provoca un fallo de página y se carga esa página junto con algunas páginas adyacentes. En este caso la página solicitada se cargó por demanda y las adyacentes se cargaron por anticipación.

7. Entrada/salida

El código destinado a manejar la entrada y salida de los diferentes periféricos en un sistema operativo es de una extensión considerable y sumamente complejo. Resuelve las necesidades de sincronizar, atrapar interrupciones y ofrecer llamadas al sistema para los programadores. En esta sección se repasarán los principios más importantes a tomar en cuenta en este módulo del sistema operativo.

7.1. Dispositivos de entrada/salida

Los dispositivos de entrada salida se dividen, en general, en dos tipos: **dispositivos orientados a bloques** y **dispositivos orientados a caracteres**. Los dispositivos orientados a bloques tienen la propiedad de que se pueden direccionar, esto es, el programador puede escribir o leer cualquier bloque del dispositivo realizando primero una operación de posicionamiento sobre el dispositivo. Los dispositivos más comunes orientados a bloques son los discos y la memoria. Por otro lado, los dispositivos orientados a caracteres son aquellos que trabajan con secuencias de bytes sin importar su longitud ni ninguna agrupación en especial. No son dispositivos direccionables. Ejemplos de estos dispositivos son el teclado, la pantalla o las impresoras.

La clasificación anterior no es perfecta, porque existen varios dispositivos que generan entrada o salida que no pueden englobarse en esas categorías.

Por ejemplo, un reloj que genera pulsos. Sin embargo, aunque existan algunos periféricos que no se puedan categorizar, todos están administrados por el sistema operativo por medio de una parte electrónica- mecánica y una parte de software.

7.2. Controladores de dispositivos (terminales y discos duros)

Los controladores de dispositivos (también llamados adaptadores de dispositivos) son la parte electrónica de los periféricos. Estos pueden tener la forma de una tarjeta o un circuito impreso integrado a la tarjeta maestra de la computadora. Por ejemplo, existen controladores de discos que se venden por separado y que se insertan en una ranura de la computadora, o existen fabricantes de computadoras que integran esa funcionalidad en la misma tarjeta en que viene la unidad central de procesamiento (tarjeta maestra).

Los controladores de dispositivos generalmente trabajan con voltajes de 5 y 12 voltios con el dispositivo, y con la computadora a través de interrupciones. Estas interrupciones viajan por el bus de la computadora y son recibidos por el CPU que a su vez pondrá en ejecución algún programa que sabrá qué hacer con esa señal. A ese programa se le llama controlador de dispositivo *(device driver)*. Algunas veces el mismo dispositivo contiene un pequeño programa en una memoria de solo lectura o en memoria de acceso aleatoria no volátil y re-escribible que interactúa con el correspondiente controlador de la computadora.

7.3. Acceso directo a memoria (DMA)

El acceso directo a memoria se inventó con el propósito de liberar al CPU de la carga de atender a algunos controladores de dispositivos. Para comprender su funcionamiento vale la pena revisar cómo trabaja un controlador sin DMA. Cuando un proceso requiere algunos bloques de un dispositivo, se envía una señal al controlador con la dirección del bloque deseado. El controlador lo recibe a través del bus y el proceso puede estar esperando la respuesta (trabajo síncrono) o puede estar haciendo otra cosa (trabajo asíncrono). El controlador recibe la señal y lee la dirección del bus. Envía a su vez una o varias señales al dispositivo mecánico (si es que lo hay) y espera los datos. Cuando los recibe los escribe en un buffer local y envía una señal al CPU indicándole que los datos están listos. El CPU recibe esta interrupción y comienza a leer byte por byte o palabra por palabra los datos del buffer del controlador (a través del device driver) hasta terminar la operación.

Como se ve, el CPU gasta varios ciclos en leer los datos deseados. El DMA soluciona ese problema de la manera siguiente. Cuando un proceso requiere uno o varios bloques de datos, el CPU envía al controlador la petición junto con el número de bytes deseados y la dirección de dónde quiere que se almacenen de regreso. El DMA actuará como una "CPU secundaria" en cuanto a que tiene el poder de tomar el control del bus e indicarle a la verdadera CPU que espere. Cuando el controlador tiene listos los datos, el DMA "escucha" si el bus está libre aprovechando esos ciclos para ir leyendo los datos del buffer del controlador e ir escribiéndolos en el área de memoria que la CPU le indicó. Cuando todos los datos fueron escritos, se le envía una interrupción a la CPU para que use los datos.
El ahorro con el DMA es que la CPU ya no es interrumpida (aunque sí puede ser retardada por el DMA) salvando así el "cambio de contexto" y además el DMA aprovechará aquellos ciclos en que el bus no fue usado por la CPU.

El hecho de que los controladores necesiten buffers internos se debe a que conforme ellos reciben datos de los dispositivos que controlan, los deben poder almacenar temporalmente, ya que la CPU no está lista en todo momento para leerlos.

8. Sistemas Windows

8.1. Evolución de los sistemas operativos de Microsoft

El primer sistema operativo de Microsoft, MS-DOS 1.0, fue lanzado en agosto de 1981 para instalarse en el nuevo ordenador personal IBM-PC.

Se trataba de un sistema operativo realmente sencillo y de características muy limitadas (monotarea, monousuario, monoprocesador, sin seguridad, etc.) que fue elegido por IBM por motivos de urgencia para el lanzamiento de su nueva máquina.

El primer intento de Microsoft de crear una interfaz gráfica (GUI, Graphical User Interface) para MS-DOS fue **Windows 1.0** en el año 1985. Pero no fue hasta Windows 3.0, en el año 1990, cuando comenzó a distribuirse ampliamente. Pronto evolucionó a **Windows 3.1**, la versión más conocida.

Windows NT (su primera versión fue la 3.1) fue lanzado en 1993 con la misma interfaz gráfica de Windows 3.1. Se trataba de un sistema operativo nuevo de 32 bits que aprovechaba las capacidades del procesador de Intel 80386.

Windows NT 4.0, en 1996, incluyó la interfaz gráfica de Windows 95. Los componentes gráficos, que en la versión anterior trabajan en modo usuario, se movieron al modo núcleo (llamado Windows NT Executive). Esta decisión fue muy polémica en su momento: aceleraba la interfaz gráfica pero podría generar riesgos de inestabilidad.

Windows 2000, lanzado en febrero de 2000, heredó la arquitectura de NT. Su mayor novedad fue la inclusión de un servicio de directorio llamado Active Directory.

Paralelamente al desarrollo de Windows NT, Microsoft lanzó **Windows 95** (1995) como evolución del sistema operativo MS-DOS con la interfaz gráfica de Windows 3.1. Windows 95 evolucionó en **Windows 98** y posteriormente, en **Windows Me**. Todos estos sistemas forman la llamada rama de sistemas operativos de consumo (uso doméstico) de Microsoft frente a los sistemas operativos profesionales basados en Windows NT.

Windows 95, 98 y Me mantienen código de 16 bits por lo que no aprovechan las características de los nuevos procesadores (Intel 80386 y siguientes). Tampoco incluyen seguridad: no soportan el sistema de archivos NTFS ni contemplan validación de usuarios.

Las dos ramas se fusionaron en un nuevo sistema operativo, Windows XP (2001), que hereda el código de Windows 2000 aunque incluye características para su uso en el hogar. El sucesor de Windows XP es Windows Vista (2006), que mejora el sistema de seguridad incluyendo lo que se conoce como control de cuentas de usuario de modo que avisa ante cualquier instalación. En el 2009 aparece Windows 7 que entre otras ventajas goza de una gran agilidad con respecto a Vista. En

el 2012 aparece Windows 8 con un nuevo interfaz pensado para que sea multiplataforma, incluyendo tablets, con la particularidad de que se suprime el conocido como botón de "Inicio" de Windows. Al año siguiente aparece Windows 8.1 que lo recupera además de incluir todo un elenco de aplicaciones y una mayor integración en lo que se conoce como "nube", es decir, delegar el almacenamiento de nuestros archivos en la red. En 2015 aparece Windows 10 con una nueva interfaz gráfica de usuario como resultado de la integración entre la interfaces Aero y Metro.

Paralelamente, en 2010, Microsoft anunciaba como producto comercial, *Azure*, una plataforma que aglutina distintos servicios en la red o como se denomina servicios en la nube. Entre ellos proporciona servicios de Active Directory, servidores web a través de IIS a partir de 2008, servicios de backup...

8.2. Interfaces gráficas de los sistemas operativos Windows

— **Aero Glass:** consiste en un efecto tridimensional de cristalizado imitando la transparencia en los bordes de las ventanas para poder ver a través de las mismas, en la barra de herramientas y en el menú Inicio. Presente en Windows Vista y Windows 7.

— **Metro:** es una forma de navegación muy natural que organiza listas en 2 dimensiones, puedes navegar tanto de forma horizontal como vertical, lo cual es ideal para dispositivos sin teclas como tablet o móviles. Presente en Windows 8 y 8.1.

— **Modern UI:** evolución de Metro llamada interfaz de azulejos optimizada para elementos sin teclado. Está presente en Windows Phone 8.

— **Continuum:** incluida en Windows 10 es una mezcla entre Windows Aero y Modern UI. Su principal características es la de renovar el botón de inicio mostrando Vista de Tareas, Centro de Actividades, Calendario, Predicción meteorológica asociada a tu ubicación y multitud de accesos directos a las aplicaciones Office, redes sociales, correo electrónico siempre que se encuentre configurado, etc.

8.3. Sistema de Archivos

— **FAT16 o FAT:** el tamaño máximo para un volumen FAT16 es 4,095 megabytes (MB).

— **FAT32:**

- Sistema de archivos que nace como evolución de FAT16 superando sus límites a la vez que sigue siendo compatible con MS-DOS.
- El tamaño máximo de archivo es de 4 GiB.
- El tamaño máximo de volumen es de 10 TiB.

— **FAT64 o extFAT** (*Extended File Allocation Table*, tabla extendida de asignación de archivos) es un sistema de archivos, patentado y privativo de Microsoft, especialmente adaptado para memorias flash presentado con Windows CE (Windows Embedded CE 6.0).

— **NTFS** (*New Technology File System*): es un sistema de almacenamiento de archivos que incorpora seguridad: solo los usuarios autorizados pueden acceder a los archivos. Se basa en la creación de listas de control de acceso (ACLs) para cada archivo o directorio. Windows NT también incluye soporte para FAT (aunque no para FAT32) y HPFS (sistema de archivos de OS/2, sistema operativo de IBM). A partir de Windows 2000 se incluye soporte para FAT32. Los objetivos de NTFS son proporcionar lo siguiente:

  • Confiabilidad, que es especialmente deseable para los sistemas avanzados y los servidores de archivos.

  • Puede manejar volúmenes de, teóricamente, hasta $2^{64} - 1$ clústeres. En la práctica, el máximo volumen NTFS soportado es de $2^{32} - 1$ clústeres (aproximadamente 16 TiB usando clústeres de 4 KiB).

  • Su principal inconveniente es que necesita para sí mismo una buena cantidad de espacio en disco duro, por lo que no es recomendable su uso en discos con menos de 400 MiB libres.

  • NTFS se incluye en las versiones de Windows 2000, Windows XP, Windows Server 2003, Windows Server 2008, Windows Vista, Windows 7, Windows 8 y Windows 10.

— **ReFS:** nuevo sistema de archivos de Windows Server 2012 *(Resilient File System*, originalmente con nombre en código «Protogon»).

  • Es un sistema de archivos compatible con NTFS pero aprovechando las características de las nuevas unidades de HD:

    – "Thin provisioning" (más espacio lógico del existente).

    – "Integrity streams" (cada archivo puede ser modificado en localización diferente).

  • ReFS utiliza árboles B+ para todas las estructuras en disco incluyendo metadatos y los datos de los archivos siendo organizados en tablas, de manera similar a una base de datos relacional.

  • El tamaño de archivo, el tamaño total de volumen, el número de archivos en un directorio y el número de directorios en un volumen están limitados a números de 64 bits.

  • Tamaño máximo de archivo de 16 Exbibytes.

  • Tamaño máximo de volumen de 1 Yobibyte (con clústeres de 64 kB), que permite gran escalabilidad prácticamente sin límites

en el tamaño de archivos y directorios (las restricciones de hardware siguen aplicando).

- El espacio libre se cuenta mediante un asignador jerárquico que comprende con tres tablas separadas para trozos grandes, medianos y pequeños.
- Los nombres de archivo y las rutas de acceso de archivo están limitados a una cadena de texto Unicode de 32 kB.

8.4. Registro de Windows

— El registro de Windows es una base de datos jerárquica que almacena los ajustes de configuración y opciones en los sistemas operativos Microsoft Windows. Contiene la configuración de los componentes de bajo nivel del sistema operativo, así como de las aplicaciones que hay funcionando en la plataforma: hacen uso del registro el núcleo *(kernel)*, los controladores de dispositivos, los servicios, el SAM, la interfaz de usuario y las aplicaciones de terceros. El registro también proporciona un medio de acceso a los contadores para generar un perfil del rendimiento del sistema.

— Hay siete claves raíz predefinidas, las cuales tradicionalmente se nombran según su identificador constante definido en la API de Win32, por sus abreviaturas correspondientes (dependiendo de las aplicaciones):

- HKEY_LOCAL_MACHINE o bien HKLM: contiene información de configuración específica del equipo (para cualquier usuario).
- HKEY_CURRENT_CONFIG o bien HKCC (únicamente en Windows 9x/Me y en las versiones basadas en NT de Windows): contiene información acerca del perfil de hardware que utiliza el equipo local cuando se inicia el sistema.
- HKEY_CLASSES_ROOT o bien HKCR. Es una subclave de HKEY_LOCAL_MACHINE\Software. La información que se almacena aquí garantiza que cuando abra un archivo con el Explorador de Windows se abrirá el programa correcto.
- HKEY_CURRENT_USER o bien HKCU. Contiene la raíz de la información de configuración del usuario que ha iniciado sesión. Las carpetas del usuario, los colores de la pantalla y la configuración del Panel de control se almacenan aquí. Esta información está asociada al perfil del usuario.
- HKEY_USERS o bien HKU. Contiene todos los perfiles de usuario cargados activamente en el equipo.
- HKEY_PERFORMANCE_DATA (únicamente en las versiones de Windows basadas en NT, pero invisible para el editor del registro).
- HKEY_DYN_DATA (únicamente en Windows 9x/Me y visible en el editor de registro de Windows).

8.5. Estado actual de los sistemas operativos Windows para PC

8.5.1. Windows 7

— Cuenta con el núcleo en su versión NT 6.1.

— Soporte arquitecturas tanto de 32 como de 64 bits.

— La interfaz gráfica de usuario se llama Aero al igual que en Windows Vista.

— Características principales que presenta:

- Mejoras en el reconocimiento de escritura a mano, soporte para discos duros virtuales, rendimiento mejorado en procesadores multinúcleo, mejor rendimiento de arranque, DirectAccess y mejoras en el núcleo.
- Soporte para sistemas que utilizan múltiples tarjetas gráficas de proveedores distintos (heterogeneous multi-adapter o multi-GPU).
- Una nueva versión de Windows Media Center y un gadget, y aplicaciones como Paint, Wordpad y la calculadora rediseñadas.
- Se añadieron varios elementos al Panel de control, como un asistente para calibrar el color de la pantalla, un calibrador de texto ClearType, Solución de problemas, Ubicación y otros sensores, Administrador de credenciales, iconos en el área de notificación, entre otros.
- El Centro de Seguridad de Windows se llama aquí Centro de actividades y se integraron en él las categorías de seguridad y el mantenimiento del equipo.
- La barra de tareas fue rediseñada, es más ancha, y los botones de las ventanas ya no traen texto, sino únicamente el icono de la aplicación, el cual se puede fijar en la barra, lo que se llama "anclar" una aplicación a la barra de tareas.
- Se colocó un botón para mostrar el escritorio en el extremo derecho de la barra de tareas, que permite ver el escritorio al posar el puntero del ratón por encima.
- Se añadieron las Bibliotecas, que son carpetas virtuales que agregan el contenido de varias carpetas y las muestran en una sola vista. Por ejemplo, las carpetas agregadas en la biblioteca Vídeos son: Mis vídeos y Vídeos públicos, aunque se pueden agregar más manualmente. Sirven para clasificar los diferentes tipos de archivos (documentos, música, vídeos, imágenes).
- Modo XP es una simulación de Windows XP para ejecutar programas antiguos.
- Incluye Microsoft Virtual PC 2007 para poder crear máquinas virtuales dentro de distintos sistemas operativos.

— Versiones:
  - Starter, indicada para NoteBook.
  - Home Basic.
  - Home Premium.
  - Professional.
  - Enterprise.
  - Ultimate.

8.5.2. Windows 8

— Cuenta con el núcleo en su versión NT 6.2.

— Soporta arquitecturas tanto de 32 como de 64 bits.

— La interfaz gráfica de usuario se llama Metro.

— Características principales que presenta:
  - Windows 8 incluye numerosas actualizaciones, entre las que se encuentran avances en reconocimiento de voz, táctil y escritura, soporte para discos virtuales, mejor desempeño en procesadores multinúcleo, mejor arranque y mejoras en el núcleo.
  - Las Bibliotecas son carpetas virtuales que agregan el contenido de varias carpetas y las muestran en una sola. Por ejemplo, las carpetas agregadas en la librería Vídeos por defecto son: Vídeos Personales (antes Mis Vídeos) y Vídeos Públicos, aunque se pueden agregar más manualmente. Sirven para clasificar los diferentes tipos de archivos (documentos, música, vídeos, fotos).
  - Multimedia: Windows 8 incluye consigo Windows Media Center y Windows Media Player 12.
  - Modo XP es una simulación de Windows XP para ejecutar programas antiguos.
  - Incluye Microsoft Virtual PC 2008.

— Versiones de Windows 8: Existen cuatro ediciones construidas una sobre otra de manera incremental:
  - Windows RT.
  - Windows 8.
  - Windows 8 Pro.
  - Windows 8 Enterprise.

8.5.3. Windows 10

— Es el último comercializado, lanzado el 29 de Julio de 2015.

— Cuenta con el núcleo en su versión NT 10.0.

— Soporte arquitecturas tanto de 32 como de 64 bits y ARM.

— La interfaz gráfica de usuario se llama Continuum.

— Paso de Smartphone a PC y viceversa: el concepto inicial de Continuum es permitir una transición fluida entre los dos modos de uso de Windows (el orientado a las pantallas táctiles y el del escritorio tradicional, con teclado y ratón).

— Características principales que presenta:

- **Cortana** (sistema de reconocimiento de voz) solo está disponible actualmente en Windows 10 para Estados Unidos, Reino Unido, China, Francia, Italia, Alemania y España.
- **Windows Hello** (reconocimiento biométrico) requiere una cámara de infrarrojos iluminada especial para el reconocimiento facial o la detección del iris, o bien un lector de huellas dactilares que sea compatible con Window Biometric Framework.
- **Continuum** (nueva interfaz gráfica) está disponible en todas las ediciones de escritorio de Windows 10 si activas y desactivas manualmente el "modo tableta" en el Centro de actividades. Las tabletas y los equipos convertibles con indicadores GPIO o los que tengan un indicador de portátil o tableta táctil podrán configurarse para entrar automáticamente en el "modo tableta".
- El **streaming de música y vídeo** a través de las aplicaciones Música y Películas y programas de TV. No está disponible en todas las regiones; para ver la lista más actualizada de regiones visita el sitio web de Xbox en Windows.
- **La aplicación Xbox requiere una cuenta de Xbox Live**, que no está disponible en todas las regiones. Para ver la lista más actualizada de regiones, visita el sitio web de países y regiones de Xbox Live.
- La autenticación en dos fases requiere el uso de un PIN, equipo biométrico (lector de huellas dactilares o cámara por infrarrojos con iluminación) o un teléfono con capacidades Wi-Fi o Bluetooth.
- El número de aplicaciones que se pueden acoplar dependerá de la resolución mínima de la aplicación.
- Para un arranque seguro se requiere firmware compatible con UEFI v2.3.1 Errata B y con la entidad de certificación de Microsoft Windows en la base de datos de firmas UEFI.

- Algunos administradores de TI pueden habilitar Inicio de sesión seguro (Ctrl + Alt + Supr) antes de que aparezca la pantalla de inicio de sesión. En las tabletas sin teclado, puede que sea necesaria una tableta con el botón Windows porque la combinación de teclas en una tableta es el botón Windows + botón de encendido.
- Es posible que algunos juegos y programas requieran tarjetas gráficas compatibles con DirectX 10 o superior para un rendimiento óptimo.
- BitLocker To Go requiere una unidad flash USB (solo para Windows 10 Pro).
- BitLocker requiere el Módulo de plataforma segura (TPM) 1.2, TPM 2.0 o una unidad flash USB (solo para Windows 10 Pro y Windows 10 Enterprise).
- Cliente **Hyper-V** requiere un sistema de 64 bits con servicios de traducción de direcciones de segundo nivel (SLAT) y 2 GB de RAM adicionales (solo para Windows 10 Pro y Windows 10 Enterprise).
- Miracast requiere un adaptador de pantalla que sea compatible con el Modelo de controladores de pantalla de Windows (WDDM) 1.3 y un adaptador Wi-Fi que admita Wi-Fi Direct.
- La impresión directa por Wi-Fi requiere un adaptador que admita Wi-Fi Direct y un dispositivo compatible con la impresión directa por Wi-Fi.
- Para instalar un sistema operativo de 64 bits en un PC de 64 bits, el procesador tiene que ser compatible con CMPXCHG16b, PrefetchW y LAHF/SAHF.
- InstantGo solo funciona con equipos diseñados para el modo de espera conectado.
- El cifrado de dispositivo requiere un PC con InstantGo y TPM 2.0.

8.6. Características de los sistemas operativos Windows de servidor

— **Soporte para arquitecturas de 32 y de 64 bits**, aunque a partir de Windows Server 2008R2 solo versionan para plataformas de 64 bits.

— **Memoria separada.** Cada programa se ejecuta en una región de memoria separada. Esto impide que el cuelgue de un programa acarree el cuelgue del resto de programas en ejecución. El sistema operativo tiene la capacidad de expulsar a un programa sin afectar al resto de programas que estén funcionando en ese momento. Aunque fallen programas aislados, el sistema operativo no debería perder el control de la máquina en ningún momento.

— **Multitarea preferente.** Permite la ejecución simultánea de aplicaciones. Al menos esta es la impresión que recibe el usuario aunque el procesador únicamente pueda estar atendiendo una tarea en cada

momento. El sistema operativo es el que asigna los ciclos de procesador para cada aplicación evitando que una de ellas lo monopolice por completo.

— **Multiprocesador.** Es capaz de utilizar varios procesadores en la misma máquina, asignando a cada uno de ellos tareas distintas.

— **Multiusuario**. Gestiona accesos concurrentes de varios usuarios al sistema desde distintos puestos de la red.

— **Portabilidad**. Windows NT fue diseñado para funcionar, sin apenas cambios, en distintos tipos de hardware. El único código específico del hardware es el conocido como HAL (*Hardware Abstraction Layer,* capa de abstracción del hardware); el resto de código es común para todos los sistemas de servidor.

— **Modelo de seguridad de dominio.** Es un sistema de acceso de usuarios a recursos de la red en el que es necesaria una autentificación previa. Los controladores de dominio son las máquinas que se encargan de la validación de los usuarios que inician sesión según la base de datos de usuarios o **SAM** (*Security Account Manager,* gestor de cuentas de seguridad). A partir de Windows 2000 Server, los dominios están integrados dentro de una nueva estructura llamada **Directorio Activo** (*Active Directory*).

— **Sistema de archivos NTFS**.

— **Sistema de archivos ReFS.**

— **Tolerancia a fallos**. Windows NT incorpora mecanismos para seguir funcionando aun en presencia de fallos. Incluye soporte para RAID (*Redundant Array of Inexpensive/Independent Disk,* matriz de discos redundantes baratos/independientes) que impide la pérdida de datos aunque falle uno de los discos duros del ordenador.

— **Versiones de Windows Server:**

- Windows 2008.
- Windows 2008R2, de aquí en adelante todas las versiones del sistema operativo de servidor son únicamente de 64 bits, con soporte para aplicaciones tanto de 32 como de 64 bits.
- Windows 2012.
- Windows 2012R2.
- Windows 2016.

8.7. Concepto de dominio

Se pueden crear redes entre iguales (grupos de trabajo) con sistemas operativos como Windows XP o Vista. Sin embargo, a medida que el número de

usuarios y de ordenadores aumenta, se hace necesario establecer unos mínimos de seguridad y centralizar toda la información en servidores. Los sistemas operativos Windows de servidor permiten crear redes con la arquitectura cliente/servidor. Este modelo de redes es, en la terminología de Microsoft, un **dominio**.

Un dominio es un conjunto de ordenadores que comparten una misma base de datos de directorio (base de datos de usuarios o SAM). Esta base de datos se almacena en los **controladores de dominio** (DC, *Domain Controller*) y está formada por cuentas de usuario y directivas de seguridad. Cuando un usuario inicia sesión en un ordenador cliente, debe indicar el nombre del dominio así como un nombre de usuario y contraseña válidos en ese dominio. Un controlador del dominio verificará en su base de datos las credenciales enviadas por el usuario.

Una máquina Windows NT Server dentro de un dominio puede actuar de una de estas tres maneras:

— **Controlador principal de dominio (PDC).** Es el servidor que contiene la SAM. Todo dominio Windows NT tiene que tener exactamente 1 controlador principal de dominio.

— **Controlador de reserva (BDC).** Contiene una copia de seguridad de la SAM. No es obligatorio que exista en una red, aunque es recomendable. Puede haber varios controladores de reserva. Se sincronizan cada 5 minutos con el controlador principal de dominio.

— **Servidor miembro.** No contiene ninguna copia de la SAM. Participa en el dominio para ofrecer recursos y servicios. Por ejemplo, un servidor de archivos.

Si el servidor no perteneciese a un dominio sería un **servidor independiente.** Por ejemplo, un servidor web público.

El controlador de reserva puede cambiar su función a controlador principal si este último tuviese una avería. Esto se consigue desde el Administrador de servidores / Menú Equipo / Promover a controlador principal de dominio.

En Windows 2000 Server y Windows Server 2003 no se distingue entre controladores principales y controladores de reserva, sino que sencillamente se habla de controladores de dominio. Un dominio Windows 2000 puede tener varios controladores de dominio para conseguir tolerancia a fallos y distribuir la carga entre todos. Se encargan de tareas como el inicio de sesión, la autentificación de usuarios y búsquedas en el directorio activo. Para configurar un servidor como controlador de dominio hay que ejecutar el comando DCPROMO (Asistente de instalación del directorio activo). Los nombres de dominio de Windows 2000 siguen la sintaxis del sistema DNS.

El **directorio activo** (AD, *Active Directory*), incluido a partir de Windows 2000, es una estructura jerárquica basada en el servicio de directorio LDAP (*Lightweight Directory Access Protocol*, protocolo de acceso a directorio ligero), que almacena información sobre componentes de red (dominios, cuentas de

usuario, grupos, ordenadores...) y facilita su búsqueda. Los componentes del directorio se denominan objetos y se almacenan en contenedores, siguiendo una estructura de árbol. Toda la información del directorio activo se replica entre los distintos controladores de dominio para mantener la consistencia. Para el funcionamiento del directorio activo, se requiere la instalación de un servidor DNS dinámico.

Otras estructuras que debemos conocer del modelo de dominio son:

— **Sitio**. Lugar físico donde reside un controlador de dominio. Los clientes tratan de iniciar sesión en controladores de dominio de su mismo sitio para acelerar la velocidad y optimizar el uso de la red.

— **Árbol**. Conjunto de dominios que comparten una misma jerarquía DNS (ejemplo: informatica3.aulas.adams.es y aulas.adams.es).

— **Bosque**. Conjunto de árboles con distintas jerarquías DNS (ejemplo: adams.es y ucm.es).

— **Unidades organizativas (OU, *Organizational Units*)**. Son divisiones de un dominio creadas para delegar su administración a distintas personas.

Para que los usuarios de un dominio puedan acceder a recursos de otro dominio debe existir una relación de confianza entre ambos dominios.

Un sistema de tipo Unix puede participar en una red Microsoft, actuando incluso como controlador de dominio. Se requiere instalar y configurar "samba" en el sistema Unix o Linux.

8.8. Cuentas de usuario y grupos

Cada usuario necesita una cuenta de usuario para poder iniciar sesión en el dominio. Los elementos básicos de una cuenta de usuario son su nombre de **usuario** y **contraseña**. La cuenta que dispone de los máximos privilegios en el sistema es la cuenta **Administrador** (equivalente a root en Unix).

Se pueden asignar privilegios a cuentas de forma independiente, aunque resulta más cómodo agrupar aquellos usuarios que deban tener unos mismos privilegios en grupos. Un grupo puede ser:

— **Global**. Puede contener usuarios de un mismo dominio.

— **Local**. Puede contener usuarios y grupos globales de distintos dominios.

— **Universal**. Puede contener usuarios, grupos globales y grupos universales de distintos dominios.

El sistema de archivos NTFS incorpora seguridad a nivel de archivos y carpetas. Permite definir listas de control de acceso (ACL, *Access Control List*) para definir de forma independiente los permisos para cada usuario o grupo. Los

permisos de Windows son más completos que los nativos de Unix, aunque Unix también tiene la opción de utilizar ACLs.

8.9. Herramientas administrativas

La administración de Windows se realiza mediante consolas administrativas situadas en el Menú Inicio / Programas / Herramientas administrativas. A continuación indicamos las más importantes:

— **Usuarios y equipos de Active Directory**. Es la herramienta más utilizada. Gestiona las cuentas de los usuarios en el dominio. En Windows NT era el Administrador de usuarios. Las cuentas de equipo se gestionaban en NT desde el Administrador de servidores.

— **Administración de equipos**. Permite administrar equipos locales o remotos. Entre las tareas de administración se encuentran la gestión de servicios iniciados, la administración de dispositivos y discos, el visor de sucesos y la gestión de recursos compartidos.

— **Windows PowerShell es una interfaz de consola** (CLI) con posibilidad de escritura y unión de comandos por medio de instrucciones *(scripts)*. En aspecto es similar al "cmd.exe" que abre una ventana del antiguo MSDOS, pero es mucho más potente e interactiva. Esta interfaz de consola está diseñada para su uso por parte de administradores de sistemas, con el propósito de automatizar tareas o realizarlas de forma más controlada. Además de interactuar con el sistema operativo, también lo hace con aplicaciones de Microsoft como SQL Server, MS-Exchange o IIS, Active Directory...

— **Administrador de servicios Internet (Internet Information Server)**. Permite configurar los servidores web, FTP, SMTP y NNTP de Windows.

— **DHCP**. Gestiona ámbitos de direcciones IP que se ofrecerán a los clientes DHCP. Permite definir opciones y reservar direcciones IP para ciertos hosts. En Windows NT era el Administrador DHCP.

— **Directivas de seguridad.** Define los privilegios de seguridad a distintos niveles: de dominio, del controlador de dominio y local.

— **DNS**. Configuración del servidor DNS. Permite definir nuevas zonas o modificar las ya existentes. En Windows NT era el Administrador de DNS.

— **Dominios y confianzas de Active Directory**. Establece relaciones de confianza entre dominios. En Windows NT se hacía desde el Administrador de usuarios.

— **Enrutamiento y acceso remoto**. Ofrece funciones de router, acceso remoto y redes privadas virtuales (VPN). En Windows NT era el Administrador de acceso remoto (RAS).

— **Rendimiento**. Muestra gráficas del rendimiento de distintos componentes del sistema (el uso de procesador, por ejemplo). Permite definir alertas de rendimiento. En Windows NT era el Monitor de sistema.

— **Servicios**. Permite iniciar y detener servicios.

— **Sistema de archivos distribuidos (DFS)**. Configura DFS para acceder a recursos compartidos en distintos hosts remotos.

— **Sitios y servicios de Active Directory**. Define los servidores que integran cada sitio y las comunicaciones entre los sitios.

— **Visor de sucesos**. Registra la actividad que se produce en el sistema. Para ello utiliza varios registros: aplicación, seguridad y sistema, entre otros.

— **WINS**. Administra el servicio de resolución de nombres de Windows. Su equivalente en Windows NT era el Administrador de WINS.

8.10. Comandos TCP/IP

— **arp**. Muestra o modifica la tabla ARP. El modificador -a muestra la tabla ARP indicando direcciones IP y direcciones físicas.

— **finger**. Muestra información de un usuario o de todos los usuarios de un sistema que ejecute el servicio finger.

— **ftp**. Cliente FTP.

— **hostname**. Muestra el nombre del host.

— **ipconfig**. Configuración IP en modo texto. Funciona en todos los sistemas operativos.

— **winipcfg**. Muestra la configuración IP en una ventana gráfica. Solo funciona en Windows 95/98/Me.

— **lpq**. Muestra el estado de una cola de impresión remota (lpd).

— **lpr**. Envía un trabajo de impresión a una impresora remota.

— **nbtstat**. Muestra estadísticas del protocolo NetBIOS.

— **netstat**. Muestra estadísticas de protocolos de red, conexiones (sockets) establecidas y puertos abiertos.

— **nslookup**. Cliente DNS. Permite realizar consultas a servidores DNS con fines de diagnóstico.

— **ping**. Envía un mensaje ICMP de solicitud de eco a un host remoto. Se utiliza con fines de diagnóstico.

— **rcp**. Copia archivos entre el ordenador local y un host remoto que ejecute el servicio RCP.

— **rexec**. Ejecuta un comando en un host remoto con el servicio REXEC habilitado.

— **route**. Muestra o modifica la tabla de encaminamiento.

— **rsh.** Similar a rexec. El host remoto debe estar ejecutando el servicio RSH.

— **tftp**. Cliente TFTP.

— **tracert**. Realiza una traza a un host remoto (es el equivalente del comando traceroute de Unix).

8.11. Comandos NET

— **net accounts**. Gestiona los valores predeterminados de cuentas de usuarios, como la longitud mínima de la contraseña.

— **net computer**. Agrega o elimina ordenadores de un dominio.

— **net config**. Muestra información de configuración de la parte cliente o la parte servidora. Modificadores: server y workstation.

— **net start.** Inicia un servicio. Si se utiliza sin parámetros, muestra los servicios en ejecución.

— **net stop.** Detiene un servicio.

— **net pause.** Suspende temporalmente la ejecución de un servicio.

— **net continue.** Reanuda un servicio suspendido con net pause.

— **net file.** Cierra un archivo compartido y lo desbloquea para que puedan utilizarlo otros usuarios. Si se utiliza sin parámetros, muestra los archivos compartidos que están abiertos en un servidor.

— **net group.** Agrega, muestra o modifica grupos globales.

— **net localgroup.** Agrega, muestra o modifica grupos locales.

— **net name.** Añade o elimina un alias de ordenador. Un alias es un nombre al que se envían mensajes mediante el servicio de mensajería.

— **net print**. Muestra trabajos de impresión y colas compartidas.

— **net send**. Envía mensajes a otros usuarios.

— **net session**. Muestra o desconecta sesiones entre el ordenador y otros hosts de la red. Si se utiliza sin parámetros, muestra las sesiones establecidas actualmente con nuestro ordenador.

— **net share**. Comparte recursos del servidor.

— **net statistics o net stats**. Muestra estadísticas de la parte cliente o la parte servidora. Modificadores: server y workstation.

— **net time.** Sincroniza el reloj del sistema con el de un host remoto o dominio.

— **net use.** Permite conectarse a un recurso remoto. El recurso remoto se mapea en una unidad de red. Ejemplo: NET USE X: \\servidor\recurso.

— **net user**. Crea y modifica cuentas de usuario. Si se utiliza sin parámetros, muestra todas las cuentas del sistema.

— **net view**. Muestra el listado de recursos que se están compartiendo en un servidor.

9. Sistemas UNIX y LINUX

9.1. Historia de Unix

En la década de los 60 se diseñaron los sistemas de tiempo compartido para paliar las deficiencias de los sistemas de procesamiento por lotes. Hasta ese momento solo era posible introducir una secuencia de programas que se ejecutaban secuencialmente. Los resultados se recogían al final de todo el proceso. A partir de los sistemas de tiempo compartido se pudo interactuar con el ordenador y obtener resultados como respuesta a las órdenes introducidas.

El primer sistema de tiempo compartido fue **CTSS** (*Compatible Time-Sharing System*, sistema de tiempo compartido compatible), desarrollado en el MIT (*Massachusetts Institute of Technology*, Instituto de Tecnología de Massachusetts).

Los investigadores del MIT trabajaron junto a los Laboratorios Bell y General Electrics para diseñar un nuevo sistema operativo: **MULTICS** (*Multiplexed Information and Computing Service*, servicio de información y cómputo multiplexado). Se programó en PL/1. MULTICS fue un sistema operativo muy adelantado a su época por sus características avanzadas de seguridad e interfaz de usuario, entre otras. Sin embargo, fracasó.

Un investigador de los Laboratorios Bell, Ken Thompson, rescribió un nuevo MULTICS pero en lenguaje ensamblador, para la máquina de escritorio DEC PDP-7. El resultado fue **UNICS**, llamado así ("uniplexado") en contraposición al enorme MULTICS ("multiplexado"). Más tarde este nombre evolucionó en "Unix" (1970).

DENNIS RITCHIE se unió al trabajo de Thompson y se trasladó Unix para la PDP-11. **Unix** se rescribió en un **lenguaje de alto nivel** llamado "B". Esto evitaba tener que reescribir el código fuente cada vez que el sistema se migraba a una máquina distinta, mejorando así su portabilidad. Recordemos que el compilador de cada máquina traduce el código de alto nivel al código fuente

que la máquina precisa. La programación en lenguaje ensamblador aunque genera códigos optimizados dificulta seriamente los traslados del sistema a otras máquinas con hardware distinto. Los sistemas operativos modernos están escritos en su mayor parte en un lenguaje de alto nivel, exceptuando algunas rutinas críticas muy dependientes del sistema, como el tratamiento de interrupciones o la gestión de memoria.

Se desarrolló un nuevo lenguaje de alto nivel llamado "C", basado en B, y se reescribió Unix en este lenguaje. Esto fue un hito en la historia de Unix. Quedaba demostrado, en contra de la creencia de la época, que un sistema operativo podía escribirse en un lenguaje de alto nivel. El lenguaje C ha continuado utilizándose en las versiones modernas de Unix y otros sistemas operativos.

Unix fue proporcionado por AT&T a universidades, acompañado de su código fuente. Esto originó que cada universidad realizara mejoras en el código del sistema operativo y se favoreciera su difusión. La universidad que más contribuyó a la mejora del código de Unix fue la Universidad de Berkeley (California). Incluyó memoria virtual, mejoró el sistema de archivos, incorporó el editor de textos vi, el nuevo shell csh y la pila de protocolos TCP/IP. Sus sistemas operativos recibieron los nombres de 1BSD, 2BSD, ... , 4BSD (4ª distribución del software de Berkeley). Se programaron primero para PDP y luego para VAX.

AT&T terminó comercializando Unix. La versión más destacable fue Unix System V.

A finales de los años 80 existían 2 versiones de Unix destacables: **4.3BSD de la Universidad de Berkeley** y la versión 3 de **Unix System V de AT&T.**

Otras organizaciones también habían modificado el código. Esto originó problemas de compatibilidad de programas entre las distintas versiones Unix. Se hizo necesario algún tipo de estandarización de Unix. El IEEE abrió el proyecto **POSIX** (*Portable Operating System unIX*, sistema operativo Unix portable) para tratar de estandarizarlo, dando lugar al estándar IEEE 1003.

Este estándar se ha ampliado en distintas direcciones:

— IBM, DEC y Hewlett-Packard formaron el consorcio OSF (Open Software Foundation).

— AT&T creó UI (Unix Internacional).

— IBM desarrolló AIX.

En la actualidad existe una gran variedad de versiones de Unix. Casi todas son fruto de la combinación y evolución de los sistemas operativos BSD y Unix System V:

— AIX, de IBM.

— HP/UX, de Hewlett Packard.

— SCO Unix.

— CrayOS.

— OpenBSD, NetBSD y FreeBSD. Versiones gratuitas bajo licencia de Berkeley. Esta licencia no obliga a que las versiones modificadas continúen siendo software libre.

— Mac OS X. Versión para ordenadores Macintosh basada en BSD.

— Solaris, de Sun Microsystems que compró Oracle en 2009, es el Unix comercial más ampliamente utilizado.

9.2. Linux

Es una variante de código abierto (*open-source*) para la plataforma IBM-PC. Fue escrita originalmente por el finlandés LINUS TORVALDS. Actualmente el desarrollo de Linux está coordinado por FSF (*Free Software Foundation*, Fundación de software abierto) mediante el proyecto GNU ("GNU's Not Unix", www.gnu.org) y licencia GPL (GNU *General Public License*).

La licencia GPL impide que versiones modificadas se conviertan en códigos propietarios, obligando a que todas las versiones que procedan de un código con licencia GPL conserven la misma licencia.

Existen para Linux diferentes distribuciones que varían en aspectos como la instalación y la interfaz, aunque el núcleo del sistema sea siempre el mismo. A continuación se enumeran las ramas más representativas de las distribuciones:

— Red Hat y distribuciones derivadas:
  - REHL: Red Hat Enterprise Linux
  - Fedora
  - CentOS
    - AsteriskNOW: Funcionalidad VoIP
    - Elastix: Funcionalidad VoIP
    - VicidialNOW
    - Rocks clusters
    - ClearOS
  - Mandrake
  - Oracle Linux Enterprise
  - Pie Box Enterprise Linux
  - Pingo Linux
  - Scientific Linux
  - Yellow Dog

— Debian y distribuciones derivadas obtenidas de debian.org.
  - Astra-Linux
  - Aptosid
  - Collax
  - Cumulus Linux
  - Damn Small Linux
  - Debian JP
  - DoudouLinux
  - Finnix
  - Grml
  - Kali
  - Kanotix
  - Knoppix
  - LMDE
  - openSUSE
  - Parsix
  - Tails
  - Ubuntu y distribuciones derivadas:
    – Linux Mint
    – Elementary OS
    – Peppermint OS
    – Trisquel
    – Zorin OS
  - Univention Corporate Server
— Arch Linux y distribuciones derivadas:
  - Manjaro
— Gentoo y distribuciones derivadas:
  - Funtoo Linux
  - Google Chrome OS

- Pentoo
- Ututo
- Zynot

— Slackware y distribuciones derivadas:

- SuseLinux

9.3. Software libre

En primer lugar se han de conocer las libertades en las que se sustenta el software libre, que son:

— La libertad de ejecutar el programa como se desea, con cualquier propósito (libertad 0).

— La libertad de estudiar cómo funciona el programa y modificarlo. El acceso al código fuente es una condición necesaria para ello.

— La libertad de redistribuir copias.

— La libertad de distribuir copias de sus versiones modificadas a terceros. Esto le permite ofrecer a toda la comunidad la oportunidad de beneficiarse de las modificaciones. El acceso al código fuente es una condición necesaria para ello.

Según el proyecto GNU, se entiende por software libre aquel que otorga las anteriores libertades, es decir, libertad de ejecución, libertad para estudiar el código fuente y adaptarlo, libertad para distribuir copias y libertad para mejorar el programa y distribuirlo.

Las principales licencias del software libre son BSD, GPL y LGPL, Apache y Creative Commons.

1. La **licencia BSD** es original de la Universidad de Berkeley. Permite utilizar, modificar y distribuir libremente el software. Sin embargo, no obliga a distribuir el código fuente, puesto que los trabajos derivados de un producto con licencia BSD pueden seguir siendo libres o dejar de serlo.

2. La **licencia GNU GPL** es más restrictiva que la licencia BSD, puesto que obliga a proporcionar el código fuente y a mantener la licencia en todos los trabajos derivados (el software no puede dejar de ser libre). Si se incluye una porción de código GPL en un proyecto, todo el código pasará a tener licencia GPL. Según esta condición, no se permite que software comercial utilice códigos GPL. La aplicación estricta de esta licencia conllevaría problemas con las bibliotecas del compilador de C incluido en Linux, puesto que no permitiría compilar y enlazar programas comerciales. Para solucionar esta dificultad, se ideó la licencia GNU LGPL.

3. **GNU LGPL** es una variante menos restrictiva que la licencia GPL, que permite integrar partes LGPL sin que todo el código pase a ser software libre. La primera L procede de *Lesser* (menor) o *Library* (biblioteca).

4. **Licencia Apache:** el software bajo este tipo de licencia permite al usuario distribuirlo, modificarlo, y distribuir versiones modificadas de ese software pero debe conservar el copyright y el disclaimer. La licencia Apache no exige que las obras derivadas (las versiones modificadas) se distribuyan usando la misma licencia, ni siquiera que se tengan que distribuir como software libre, solo exige que se informe a los receptores que en la distribución se ha usado código con la licencia Apache. En este sentido, al crear nuevas piezas de software, los desarrolladores deben incluir dos archivos en el directorio principal de los paquetes de software redistribuidos: una copia de la licencia y un documento de texto que incluya los avisos obligatorios del software presente en la distribución.

5. **Licencias Creative Commons:** su definición se basa en cuatro condiciones:

   a) "Atribución", con la cual se puede distribuir, exhibir, representar... siempre y cuando se reconozca y se cite a su autor (CC BY).

   b) "No comercial", que no permite usar el software con fines comerciales (CC NC)

   c) "No derivadas", con la cual no se puede modificar dicha obra (CC ND).

   d) "Compartir igual", que incluye la creación de obras derivadas siempre que mantengan la licencia original (CC SA).

9.4. Capas de Unix

Los sistemas operativos Unix se pueden dividir en las siguientes capas:

— (5) Programas de usuario.

— (4) Programas estándares (shells, editores y compiladores).

— (3) Llamadas al sistema (open, read, close, fork, etc).

— (2) Sistema operativo Unix (procesos, gestión de memoria, archivos y entrada/salida).

— (1) Hardware (CPU, memoria, discos, etc).

La capa 2 constituye el **núcleo (kernel)** del sistema operativo. Las rutinas de este nivel funcionan en modo núcleo. En cambio, las capas superiores trabajan en modo usuario.

9.5. Kernel

El kernel o núcleo de Linux se puede definir como el corazón de este sistema operativo. Es el encargado de que el software y el hardware puedan trabajar juntos.

A fecha de edición de este libro, la última versión estable del kernel es la 4.12.5 y como curiosidad indicar que antes del desarrollo de la serie 2.6 del núcleo, existieron dos tipos de versiones del mismo:

— Versión de producción: era la versión estable hasta el momento.

— Versión de desarrollo: esta versión era experimental y era la que utilizaban los desarrolladores para programar, comprobar y verificar nuevas características, correcciones, etc. Estos núcleos solían ser inestables y no se debián usar sin saber lo que se hacía.

Cómo interpretar los números de las versiones de las series por debajo de la 2.6:

— Las versiones del núcleo se numeraban con 3 números, de la siguiente forma: AA.BB.CC

  • AA: indicaba la serie/versión principal del núcleo. Solo han existido 1 y 2. Este número cambiaba cuando la manera de funcionamiento del kernel había sufrido un cambio muy importante.

  • BB: indicaba si la versión era de desarrollo o de producción. Un número impar significaba que era de desarrollo, uno par que era de producción.

  • CC: indicaba nuevas revisiones dentro de una versión, en las que lo único que se había modificado eran fallos de programación.

  Unos ejemplos nos ayudarán a entenderlo mejor:

  • Ejemplo 1: versión del núcleo 2.4.0. Núcleo de la serie 2 (AA=2), versión de producción 4 (BB=4 par), primera versión de la serie 2.4 (CC=0).

  • Ejemplo 2: versión del núcleo 2.4.1. Núcleo de la serie 2, versión 4, en la que se han corregido errores de programación presentes en la versión 2.4.0 (CC=1).

  • Ejemplo 3: versión del núcleo 2.5.0. Versión 0 del núcleo de desarrollo 2.5.

  Con la serie 2.6 del núcleo el sistema de numeración y el modelo de desarrollo han cambiado. Las versiones han pasado a numerarse con 4 dígitos y no existen versiones de producción y de desarrollo. Las versiones del núcleo se numeran hoy en día con 4 dígitos, de la siguiente forma: AA.BB.CC.DD.

  • AA: indica la serie/versión principal del núcleo.

  • BB: indica la revisión principal del núcleo. Números pares e impares no tienen ningún significado hoy en día.

  • CC: indica nuevas revisiones menores del núcleo. Cambia cuando nuevas características y drivers son soportados.

- DD: este dígito cambia cuando se corrigen fallos de programación o fallos de seguridad dentro de una revisión.

El núcleo se divide en dos: dependiente e independiente. El núcleo **dependiente** es el que se encarga de manejar las interrupciones del hardware, hacer el manejo de bajo nivel de la memoria y de los discos y trabajar con los controladores de dispositivos de bajo nivel, principalmente. Por otro lado, el núcleo **independiente** del hardware se encarga de ofrecer las llamadas al sistema, manejar los sistemas de archivos y planificar los procesos.

9.6. Shells

En Unix, los usuarios pueden elegir el intérprete de comandos que prefieren utilizar. Los distintos intérpretes difieren básicamente en la introducción y edición de órdenes y la programación de **scripts** (pequeños programas interpretados que se construyen con órdenes del sistema). Los más importantes son:

— **Bourne Shell** (/bin/sh). Es el shell más antiguo.

— **C-Shell** (/bin/csh). Está basado en el lenguaje C.

— **Tab C-Shell** (/bin/tcsh). Es una mejora del C-Shell.

— **Korn Shell** (/bin/ksh). Incorpora mejoras de programación.

— **Bourne Again Shell** (/bin/bash). Incluye características avanzadas de C-Shell y Korn Shell, aunque sigue siendo compatible con Bourne Shell.

9.7. Entornos de escritorio

Los shells explicados anteriormente son en modo texto. En las distintas versiones de Unix y Linux se pueden utilizar también entornos gráficos de escritorio, siendo los más habituales:

— **KDE**. Utiliza **Konqueror** como gestor de archivos y navegador web.

— **GNOME**. Utiliza **Nautilus** como gestor de archivos. El usuario puede elegir su navegador web como, por ejemplo, Firefox.

— **Xfce**. Utiliza **Thunar** como gestor de archivos.

— **Unity.** Es un entorno de escritorio utilizado por las distribuciones de Ubuntu.

— **LXDE** es un entorno de escritorio libre para Unix y otras plataformas POSIX, como Linux o BSD. El nombre corresponde a "Lightweight X11 Desktop Environment", es decir, "Entorno de escritorio X11 ligero.

— **MATE** es un entorno de escritorio derivado del código base de GNOME 2.

— **Cinnamon** es un entorno de escritorio para el sistema operativo GNU/Linux, desarrollado inicialmente por el proyecto Linux Mint como una bifurcación de GNOME Shell.

9.8. Editores de texto

Las opciones de configuración en Unix se almacenan en general en archivos de texto, por lo que se requiere un editor de texto para su modificación. Los editores de texto también son necesarios para crear los códigos fuente de los programas antes de compilarlos.

Los editores de texto más conocidos son **vi** (compacto y universal pero difícil de manejar) y **emacs** (más funciones y sencillo de manejar).

Otros editores de texto son: vim, nvi, xemacs, pico, nano y joe.

Editor vi

Existen tres modos o estados de vi, para realizar las tareas de edición el usuario va cambiado entre un modo u otro en función de lo que desee realizar. Los modos son:

— **Modo comando:** este es el modo en el que se encuentra el editor cada vez que se inicia. Las teclas ejecutan acciones (comandos) que permiten mover el cursor, ejecutar comandos de edición de texto, salir de vi, guardar cambios, etc.

— **Modo inserción o texto:** este es el modo que se usa para insertar el texto. Existen varios comandos que se pueden utilizar para ingresar a este modo. Para pasar de modo texto a modo comando simplemente se debe apretar la tecla ESC.

— **Modo línea o ex:** se escriben comandos en la última línea al final de la pantalla. Para volver al modo comando desde el modo última línea se debe apretar la tecla ENTER (al finalizar el comando) o la tecla ESC (que interrumpe el comando).

Algunas de las **acciones de edición** que nos permite este editor son:

— **Cambio de modo comando a modo texto**

Comando	Acción
i	Inserta texto a la izquierda del cursor.
a	Inserta texto a la derecha del cursor.
A	Inserta texto al final de la línea donde se encuentra el cursor.
I	Inserta texto al comienzo de la línea donde se encuentra el cursor.
o	Abre una línea debajo de la actual.
O	Abre una línea encima de la actual.

— **Borrado de texto**

Comando	Acción
x	Borra el carácter bajo el cursor.
dd	Borra la línea donde se encuentra el cursor.
ndd	Borra las próximas n líneas.
D	Borra desde donde se encuentra el cursor hasta el final de la línea.
dw	Borra desde donde se encuentra el cursor hasta el final de una palabra.

— **Cortar y pegar:** esto implica mover partes del archivo de un lugar a otro del mismo. Para esto se debe:

- Cortar el texto que se desea mover utilizando alguno de los comandos usados para borrar texto.
- Mover el cursor (con alguno de los comandos utilizados para desplazar el cursor en el texto) hasta el lugar donde se desee pegar el texto.
- Pegar el texto con el comando **p**.

— **Copiar y pegar:** esta operación difiere de la anterior. En este caso lo que se hace es repetir partes del texto en otro lugar del archivo. Para esto se debe:

- Utilizar el comando yy cuya función es copiar la línea donde se encuentra situado el cursor.
- Mover el cursor (con alguno de los comandos utilizados para desplazar el cursor en el texto) hasta el lugar donde se desee pegar el texto.
- Pegar el texto con el comando **p**.

— **Deshacer cambios:** se puede deshacer el último cambio realizado, utilizando el comando **u**.

— **Buscar texto**

Comando	Acción
/texto	Busca hacia adelante la cadena de caracteres “texto”
?texto	Busca hacia atrás la cadena de caracteres “texto”

— **Salir salvando o no los cambios**

Comando	Acción
:q	Salir si no hubo cambios
:q!	Salir sin guardar cambios
:w	Guardar cambios
:w archivo1	Guardar cambios en archivo1
:wq	Guardar cambios y salir

9.9. Gestores de arranque

Si además de Unix/Linux, queremos instalar otro sistema operativo en la misma máquina, necesitaremos un gestor de arranque que nos permita elegir el sistema operativo que queremos arrancar en cada momento.

Los dos más conocidos son **LILO** (LInux LOader) y **GRUB** (GRand Unified Bootloader). Estos gestores se instalan habitualmente en el sector de arranque del disco duro (MBR, Master Boot Record). Mediante LILO o GRUB podemos elegir entre arrancar Linux y otros sistemas operativos.

LILO, en su última versión (v24.2), nos permite elegir entre 16 imágenes posibles de arranque. En el caso de GRUB actualmente se encuentra en su versión 2 renombrando la versión 1 como GRUB Legacy.

Los sistemas operativos de Microsoft tienen su propio gestor de arranque denominado NT Loader (NTLDR) que se instala en el sector de arranque de la partición primaria de Windows (C:). Es posible instalar ambos gestores en la misma máquina, puesto que se ubican en lugares diferentes.

9.10. Sistema de archivos

Linux soporta gran variedad de sistemas de ficheros, desde sistemas basados en discos, como pueden ser ext2, ext3, ReiserFS, XFS, JFS, UFS, ISO9660, FAT, FAT32 o NTFS, a sistemas de ficheros que sirven para comunicar equipos en la red de diferentes sistemas operativos, como NFS (utilizado para compartir

recursos entre equipos Linux) o SMB (para compartir recursos entre máquinas Linux y Windows). De entre los primeros, tenemos:

— **ext2:** hasta hace poco era el sistema estándar de Linux. Tiene una fragmentación muy baja, aunque es algo lento manejando archivos de gran tamaño. Fue la continuación del sistema de ficheros ext, implementado en 1992 e integrado en Linux 0.96. Las principales ventajas que tenía sobre ext eran las siguientes:

  • Compatible con sistemas de ficheros grandes, admitiendo particiones de disco de hasta 4TB y ficheros de hasta 2GB de tamaño.

  • Proporciona nombres de ficheros largos, de hasta 255 caracteres.

  • Tiene una gran estabilidad.

— **ext3:** es la versión mejorada de ext2, con previsión de pérdida de datos por fallos del disco o apagones que se conoce como *journaling,* que viene a ser como un registro de transacciones como se utiliza en bases de datos. Se guarda información en cuanto a transacciones que afecten a estructuras de directorio, bloques libres de disco y descriptores de archivo (tamaño, fecha de modificación...). Utiliza como estructuras de datos árboles equilibrados en altura (AVL). En contraprestación, es totalmente imposible recuperar datos borrados. Es compatible con el sistema de ficheros ext2. Actualmente es el más difundido dentro de la comunidad GNU/Linux y es considerado el estándar. Sus ventajas frente a ext2 son:

  • Actualización: debido a que los dos sistemas comparten el mismo formato, es posible llevar a cabo una actualización a ext3, incluso aunque el sistema ext2 esté montado.

  • Fiabilidad y mantenimiento.

— **ext4:** es la última versión de la familia de sistemas de ficheros ext. Sus principales ventajas radican en su eficiencia (menor uso de CPU, mejoras en la velocidad de lectura y escritura) y en la ampliación de los límites de tamaño de los ficheros, ahora de hasta 16TB, y del sistema de ficheros, que puede llegar a los 1.024 PB (PetaBytes).

— **ReiserFS:** es el sistema de ficheros de última generación para Linux. Organiza los ficheros de tal modo que se agilizan mucho las operaciones con estos. El problema de ser tan actual es que muchas herramientas (por ejemplo, para recuperar datos) no lo soportan.

Los archivos de Unix se organizan de forma jerárquica (forma de árbol invertido). En el punto más alto se sitúa el directorio **raíz** (/) del cual depende todo el sistema de archivos. Al contrario que en los sistemas Windows, las unidades de disco también dependen de este directorio raíz. La estructura de directorios y subdirectorios varía ligeramente según la versión Unix que estemos utilizando. Sin embargo, podemos indicar algunos de los directorios más frecuentes:

- **/bin**. Archivos binarios. Contiene ficheros de comandos ejecutables utilizables por todos los usuarios. Aquí tenemos los programas que pueden lanzar todos los usuarios del sistema.
- **/sbin**. Para ejecutables de uso exclusivo por el superusuario root. Son los necesarios para arrancar y montar el directorio /usr.
- **/dev**. Archivos de dispositivos. Contiene archivos especiales de bloques y caracteres asociados a dispositivos hardware. Aquí se encuentran todos los dispositivos físicos del sistema (todo el hardware de la máquina).
- **/etc**. Archivos de configuración.
- **/proc.** Contiene los archivos que reciben o envían información al núcleo. No deberíamos modificar el contenido de este directorio.
- **/boot.** Contiene los archivos de configuración del arranque del sistema, como por ejemplo GRUB.
- **/media.** Contiene todas las unidades físicas que tenemos montadas: discos duros, unidades de DVD, pen drives, etc.
- **/opt.** Sirve para admitir ficheros nuevos creados tras la modificación del sistema. Es un punto de montaje desde el que se instalan los paquetes de aplicación adicionales. Podemos usarla para instalar aplicaciones que no vienen en los repositorios, por ejemplo, aquellas que compilamos a mano.
- **/lib**. Bibliotecas compartidas.
- **/tmp**. Archivos temporales.
- **/var**. Archivos variables (por ejemplo, colas de impresión).
- **/usr**. Datos, programas y librerías utilizadas por aplicaciones del usuario.
  - **/usr/bin.**
  - **/usr/lib.**
  - **/usr/man**. Documentación de programas.
  - **/usr/src**. Códigos fuente de programas.
  - **/usr/tmp.**
  - **/usr/var**.
- **/mnt.** Montaje de dispositivos.
- **/root**. Directorio de la cuenta del superusuario (root).
- **/home**. Directorios de usuarios.

El directorio **/mnt** se utiliza, como hemos indicado, para el montaje de dispositivos, como unidades de disquete, particiones y unidades de

CD-ROM. Esta operación es específica de sistemas Unix (no existe en Windows). Para poder acceder a un dispositivo primero hay que montarlo en un directorio. En este momento, el sistema lee el dispositivo y lo refleja en el directorio indicado. Todas las operaciones las realizamos en el directorio indicado (los datos modificados quedan en memoria). El sistema irá almacenando realmente la información en el dispositivo físico cuando no tenga otras tareas que realizar. Finalmente, hay que desmontar la unidad para que la información se sincronice definitivamente en el disco. Si se apaga un sistema Unix sin haber desmontado previamente las unidades, probablemente perdamos información. Obsérvese que es necesario desmontar dispositivo extraíble antes de retirarlo físicamente.

9.10.1. Discos duros y particiones

— Todos los discos duros en Linux se encuentran localizados bajo la carpeta /**dev**.

— Las dos primeras letras del nombre de la partición indican el tipo de dispositivo donde reside la partición. Normalmente serán **hd** (para discos IDE) o **sd** (para discos SCSI).

— La tercera letra indica qué dispositivo contiene la partición. Por ejemplo, /dev/hda (el primer disco duro IDE) o /dev/sdb (segundo disco SCSI).

  • La nomenclatura cambia si se trata de un disco duro IDE:

    – ide0 = primario maestro = hda.

    – ide1 = primario esclavo = hdb.

    – ide2 = secundario maestro = hdc.

    – ide3 = secundario esclavo= hdd.

  • Para dispositivos SCSI o SATA exactamente igual que para dispositivos IDE:

    – /dev/sda o /dev/scd0 (dispositivo 1).

    – /dev/sdb o /dev/scd1 (dispositivo 2).

    – /dev/sdc o /dev/scd2 (dispositivo 3).

— El cuarto carácter es un número que indica la partición:

  • Las primeras cuatro (primarias o extendidas) particiones son numeradas de 1 a 4.

  • Particiones lógicas empiezan en 5 hasta 16.

Por ejemplo, /dev/hda3 es la tercera primaria o extendida en el primer disco IDE; /dev/sdb6 es la segunda partición lógica del segundo disco duro SCSI.

Ejemplos:

— **/dev/hda1**. Primera partición del dispositivo maestro del canal 1 IDE.

— **/dev/hda2.** Segunda partición del dispositivo maestro del canal 1 IDE.

— **/dev/hda5.** Primera unidad lógica del dispositivo maestro del canal 1 IDE.

— **/dev/hdb5.** Primera unidad lógica del dispositivo esclavo del canal 1 IDE.

— **/dev/hdd3**. Partición 3 del disco esclavo del segundo bus IDE.

— **/dev/sda**. Primer disco SCSI.

— **/dev/sdb**. Segundo disco SCSI.

— **/dev/sdb6**. Partición 6 del segundo disco SCSI.

— **/dev/scd0**. Primer CD-ROM SCSI.

— **/dev/scd1**. Segundo CD-ROM SCSI.

— **/dev/fd0**. Primera disquetera.

— **/dev/ttyS0**. Primer puerto de comunicaciones serie (COM1).

— **/dev/ttyS1**. Segundo puerto de comunicaciones serie (COM2).

— **/dev/lp0**. Puerto de impresora 1 (LPT1).

— **/dev/null**. Dispositivo nulo.

— **/dev/tty1.** Terminal 1. Un terminal es básicamente una forma de introducir órdenes (teclado) y obtener resultados (monitor).

— **/mnt/floppy**. Si se monta el disquete en este directorio, encontraremos aquí los archivos del disquete.

— **/mnt/cdrom**. Si se monta el CD-ROM en este directorio, encontraremos aquí los archivos del CD-ROM.

— **fdisk -l.** Un buen comando para mostrar un listado con todos los discos duros y particiones es fdisk; el modificador -l es para mostrar el listado de los dispositivos instalados.

9.10.2. Tipos de archivos

Los archivos de un sistema de archivos Unix pueden ser de los siguientes tipos:

— **Ordinarios**. Archivos de datos o programas.

— **Directorios**. Contienen una lista de archivos con punteros a sus inodos.

— **Especiales**. En este grupo se incluyen los dispositivos que hemos visto anteriormente, como puertos de comunicación y discos.

— **Tuberías con nombre *(named pipes)***. Comunican dos procesos.

Los nombres de los archivos pueden tener hasta 255 caracteres. Se pueden crear archivos ocultos colocando un punto como primer carácter de su nombre. Por ejemplo, si creamos el archivo /tmp/.secreto, este no se listaría entre los demás archivos del directorio.

Otro concepto que debemos conocer es el de **enlace**. Un enlace (vínculo o link) permite que un mismo archivo pueda llamarse desde varios directorios. En realidad, solo existe una copia del archivo, aunque podamos verlo desde varios directorios. Supongamos que tenemos 2 enlaces al mismo archivo: /tmp/texto y /root/texto. Un listado de cualquiera de los dos directorios mostraría que existen 2 enlaces al archivo texto. Si intentamos borrar el archivo en uno de los dos directorios, en realidad solo se borrará el enlace correspondiente y los datos no se verán afectados. Finalmente, si borramos el archivo en el otro directorio, el archivo desaparecerá completamente (era el último enlace). Cuando se habla de enlaces en general, nos estamos refiriendo a enlaces duros (enlaces físicos o *hard link*).

El término de enlaces duros se utiliza en oposición a enlaces simbólicos (*symbolic link*). Un enlace simbólico es otro archivo que apunta al primero. Esto es similar a los accesos directos de Windows. Si borrásemos el archivo original, perderíamos los datos. Los posibles enlaces simbólicos quedarían apuntando a un archivo inexistente.

Un **subdirectorio** es un directorio dentro de otro. En realidad, todos los directorios excepto el raíz son subdirectorios de otro. Los subdirectorios contienen como mínimo 2 entradas:

— "." (**un punto**). Se corresponde con el directorio actual.

— ".." (**dos puntos seguidos**). Se corresponde con su directorio padre.

9.10.3. Organización del disco. Inodos

Los archivos Unix están gestionados por inodos (nodos índice). Un inodo es una estructura de 64 bytes que contiene la siguiente información:

— Tipo de archivo y permisos.

— Número de referencias del archivo en directorios (enlaces).

— Identificador del propietario.

— Identificador del grupo del propietario.

— Tamaño del archivo en bytes.

— Fecha de último acceso del archivo.

— Fecha de última modificación del archivo.

— Fecha de última modificación del inodo.

— Dirección. Está formado por 39 bytes divididos en 13 punteros de 3 bytes.

  • Los 10 primeros punteros son directos. Contienen direcciones de bloques de datos.

  • Los otros 3 son punteros indirectos:

    - Indirecto simple. Contiene un puntero a un bloque de 256 punteros directos (total: 256 bloques de datos).

    - Indirecto doble. Contiene un puntero a un bloque de 256 punteros indirectos simples (total: 65536 bloques de datos).

    - Indirecto triple. Contiene un puntero a un bloque de 256 punteros indirectos dobles (total: 16 millones de bloques de datos).

La asignación de bloques para un archivo se realiza de forma dinámica, esto es, según se necesita. Debido a este dinamismo, los bloques de datos de un archivo pueden no asignarse seguidos, originando fragmentación. La fragmentación de los discos obliga a un mayor movimiento de las cabezas de lectura y escritura de los discos duros para recorrerse un archivo completo y, por lo tanto, se pierde velocidad.

En Unix System V se utilizan bloques de datos pequeños (1 kB), al contrario que en el sistema de archivos de Windows llamado FAT (*File Allocation Table*, tabla de asignación de archivos) que pueden llegar a tener hasta 32 KB. Los bloques pequeños evitan que se desaproveche espacio en disco. Los punteros de dirección de un archivo se van utilizando a medida que se necesiten, comenzando siempre por los punteros directos (que son más rápidos). Suponiendo bloques de 1 kB las capacidades que ofrece cada sistema de direccionamiento son:

— Directo: 10 kB (10 bloques).

— Indirecto simple: 256 kB (256 bloques).

— Indirecto doble: 65 MB (65536 bloques).

— Indirecto triple: 16 GB (16 millones de bloques).

El tamaño máximo teórico de un archivo es la suma de los capacidades de los 4 tipos de direccionamiento: 16 GB aproximadamente.

Las ventajas de la utilización de este sistema de organización de archivos son:

— Los inodos tienen un tamaño pequeño y pueden mantenerse en memoria.

— El acceso a los archivos pequeños es rápido, ya que no se utilizan punteros indirectos.

— Casi no se desaprovecha espacio en disco como ocurre en FAT.

— El tamaño límite teórico de un archivo es elevado.

La organización de un disco en los sistemas Unix tradicionales tiene la siguiente estructura:

— Bloque de arranque (bloque 0). Contiene información para el arranque del sistema operativo.

— Superbloque (bloque 1). Contiene información crítica de la organización del sistema de archivos, como el número de inodos y el número de bloques.

— Inodos. Vector de inodos con la estructura que hemos visto anteriormente para cada uno de ellos.

  • Inodo 1. Está reservado para manejo de bloques defectuosos.

  • Inodo 2. Lo utiliza siempre el directorio raíz.

— Bloques de datos. Almacenan el contenido de los archivos. Cada archivo utiliza 1 o más bloques de datos.

9.11. Cuentas de usuario

Para iniciar sesión en un sistema Unix, el usuario debe disponer de una cuenta en el sistema. Existe una cuenta especial que dispone de los máximos privilegios llamada **root** (superusuario). El root del sistema puede acceder a todos los archivos y realizar tareas administrativas como la apertura o cierre de cuentas.

Los elementos básicos de una cuenta de usuario son:

— **Nombre de usuario** (login).

— **UID** (*user ID*, identificador de usuario). Número que identifica al usuario.

— **GID** (*group ID*, identificador de grupo). Número que identifica al grupo principal del usuario.

— **Contraseña** *(password)*.

— **Shell**. Intérprete de comandos (interfaz de órdenes) que ejecutará el usuario al iniciar sesión (ejemplo: /bin/bash).

— **Directorio home.** Directorio del usuario (ejemplo: /home/maria).

— **Comentario**.

La información relativa a cuentas de usuario se almacena en los siguientes archivos:

— **/etc/passwd**. Contiene el listado de usuarios. Para cada usuario se almacena un registro con los siguientes campos:

  • Nombre_usuario: contraseña: uid:gid:comentario:directorio_home:shell

    El campo de contraseña suele contener una "x" para indicar que se encuentra encriptada en el archivo /etc/shadow.

— **/etc/shadow**. Contiene las contraseñas encriptadas de los usuarios. Para cada usuario se almacena un registro con los siguientes campos:

  • nombre_usuario:contraseña_encriptada:fecha_última_mod:mín_días_entre_mod:máx_días_entre_mod:días_aviso_expiración: máx_días_cuenta_inactiva:fecha_expiración.

— **/etc/group.** Contiene el listado de grupos de usuarios. Para cada grupo se almacena un registro con los siguientes campos:

  • Nombre_grupo: contraseña: gid: lista_usuarios.

    La contraseña no se utiliza, contiene un asterisco o bien queda en blanco. En lista_usuarios se indican los UID de los miembros secundarios del grupo, separados por comas. Obsérvese que un usuario puede pertenecer a varios grupos. Su grupo principal se indica en el archivo /etc/passwd y sus grupos secundarios se indican incluyendo su UID en las listas de usuarios correspondientes del archivo /etc/group.

9.12. Permisos

Los permisos básicos de Unix son **lectura** (r), **escritura** (w) y **ejecución** (x). Se definen para el propietario del archivo (u), para el grupo al que pertenece (g) y para el resto de usuarios (o). El propietario de un archivo es habitualmente el usuario que lo ha creado. Los permisos se acostumbran a representar en un vector de 9 bits: rwxrwxrwx. Los 3 primeros bits se corresponden con el propietario, los 3 siguientes con el grupo y los 3 últimos con el resto de usuarios.

Por ejemplo, un archivo con permisos **rw-r——** significa que el propietario puede leer y escribir, los usuarios de su grupo solo pueden leer y el resto de usuarios no tiene ningún permiso. Este vector puede representarse mediante un dígito octal para cada grupo de 3 bits. En el ejemplo anterior, los permisos podrían representarse como 640 (en binario es 110 100 000). Delante del vector de permisos se suele añadir un bit "d" para indicar si el archivo es un directorio. Por ejemplo, un directorio podría tener los permisos: **drwxr-xr-x**.

Otros ejemplos: el permiso 500 permite únicamente lectura y ejecución para el propietario del archivo. El permiso 777 indica acceso completo para todos los usuarios.

Los permisos básicos también se aplican a directorios. El significado de cada permiso aplicado a un directorio se explica a continuación:

— **r**. Permite listar el contenido del directorio.

— **w**. Permite crear o eliminar archivos. Obsérvese que si no tenemos permiso de escritura en un archivo pero sí en su directorio, no podríamos modificarlo aunque sí eliminarlo.

— **x**. Permite cambiar a ese directorio (ubicarse dentro del directorio con la orden cd) y hacer operaciones con sus archivos y subdirectorios.

Además de los permisos básicos, existen 3 permisos avanzados:

— **suid** (*set user id*, fijar identificador de usuario). Cuando un usuario normal ejecuta un programa con el permiso suid activado, se considera como si fuese el propietario del archivo. Si el propietario del programa es root, entonces el programa se ejecutará con privilegios de root, aunque el usuario no lo sea. Este permiso lo requieren algunos comandos del sistema para que usuarios normales puedan ejecutarlos. Este es el caso de las órdenes lp, lpstat y passwd. Se representa con la letra "s" en lugar de la "x" del usuario. Por ejemplo, la orden /usr/bin/lp tiene los permisos -r-s—x—x (el primer bit es el de directorio que está desactivado).

— **sgid** (*set group id*, fijar identificador de grupo). Cuando un usuario normal ejecuta un programa con el permiso sgid activado, se considera como si perteneciese al grupo del propietario del archivo. Se representa con la letra "s" en lugar de la "x" de grupo. Por ejemplo, la orden /usr/bin/write tiene los permisos -r-xr-sr-x.

— **sticky bit** (identificador de permanencia). Su significado es distinto en un programa y en un directorio. Se representa con una "t" en lugar de la "x" de otros usuarios. Por ejemplo, el directorio /tmp tiene los permisos drwxrwxrwt.

  • En un programa, hace que permanezca en memoria al terminar su ejecución (consigue más velocidad).

  • En un directorio, añade seguridad impidiendo que un usuario borre los archivos de otro usuario. Este permiso se aplica a directorios públicos como /tmp.

9.13. Listado de comandos

9.13.1. Comandos del sistema de archivos

— **ls**. Muestra los archivos de un directorio.

— **pwd**. Muestra el directorio actual (ubicación dentro del sistema de archivos en que nos encontramos).

— **cd**. Cambia el directorio actual.

— **cat**. Muestra el contenido de un archivo de texto.

— **mv**. Mueve o renombra archivos.

— **cp**. Copia archivos.

— **rm**. Borra archivos.

— **ln**. Crea un enlace a un archivo. Para crear enlaces simbólicos hay que indicar el modificador -s.

— **touch**. Pone la fecha actual a un archivo. Si no existe, lo crea.

— **chmod**. Cambia los permisos de un archivo. Ejemplos:

- chmod go-rx texto

  Quita los permisos de lectura y ejecución al grupo y al resto de usuarios del archivo texto.

- chmod ugo+rwx texto

  Pone todos los permisos al archivo texto.

- chmod 700 texto

  Cambia los permisos del archivo texto a: rwx———

- chmod 750 texto

  Cambia los permisos del archivo texto a: rwxr-x—

- chmod 774 texto

  Cambia los permisos del archivo texto a: rwxrwxr–

- chmod u+s programa

  Añade el suid a un archivo. También podría indicarse en notación octal: 4711 (se añade un dígito octal al principio para representar los permisos avanzados: suid-sgid-stickybit).

- chmod g+s programa

  Añade el sgid a un archivo. También podría indicarse en notación octal: 2751.

- chmod +t compartido

  Añade el sticky bit al directorio compartido. También podría indicarse en notación octal: 1777.

— **umask**. Establece la máscara de permisos. Se utiliza para indicar los permisos iniciales que tendrán los archivos creados después de la ejecución de umask. Para que este comando se ejecute cada vez que se inicie sesión se debe incluir en un archivo especial llamado **profile**. Se indica un código numérico que representa el valor que hay que restar de 777. Ejemplos:

- umask 022

  Crea archivos con permisos 755 (777-022=755).

- umask 077

  Crea archivos con permisos 700 (777-077=700).

— **chown**. Cambia el propietario de un archivo.

- chown isabel texto

— **find**. Busca archivos con un determinado nombre.

- find . -name perdido -print

  Busca todos los archivos llamados "perdido" comenzando por el directorio actual (.) y mostrando la ruta completa de los archivos encontrados (-print).

— **pg y more**. Visualizan un archivo de texto página a página.

— **head**. Muestra las primeras líneas de un archivo de texto.

— **tail**. Muestra las últimas líneas de un archivo de texto.

— **mount**. Monta una unidad en un directorio.

- mount /dev/fdo /mnt/floppy

Monta el dispositivo /dev/fdo (disquetera) en el directorio /mnt/floppy.

— **umount**. Desmonta un dispositivo.

- umount /dev/fdo

  Desmonta la disquetera

— **tar**. Compacta (empaqueta) un árbol de directorios en un archivo.

— **gzip**. Comprime un archivo.

9.13.2. Comandos de impresión

— **lp y lpr**. Imprimen un archivo.

— **lpstat**. Muestra información sobre la cola de impresión.

— **cancel**. Cancela un trabajo de impresión. Hay que indicar su id de impresión (tiene la sintaxis de nombreimpresora-númtrabajo).

- cancel laser-183

— **lpadmin**. Configura el servicio de impresión.

— **lpq**. Muestra cola de impresión.

— **lprm**. Elimina un trabajo de la cola de impresión.

— **accept**. Permite que la impresora acepte trabajos.

— **enable**. Activa impresora.

9.13.3. Control de trabajos y procesos

Se pueden ejecutar trabajos en segundo plano escribiendo el símbolo "&" al final de la línea de órdenes. El símbolo del sistema se presentará inmediatamente sin esperar a que termine la ejecución de la orden. Ejemplo: cp * /tmp/pruebas &

— **jobs**. Muestra todos los trabajos.

— **kill**. Finaliza un trabajo o proceso.

- kill %2

  Finaliza el trabajo número 2.

- kill 1834

  Finaliza el proceso con PID (process ID, identificador de proceso) 1834. El listado de procesos se obtiene con la orden ps.

- kill -9 1834

  Finaliza inmediatamente (señal -9) el proceso con PID 1834.

— **fg**. Pone un trabajo como trabajo principal (permite la interacción con el trabajo).

— **bg**. Lleva un trabajo a segundo plano.

— **stop**. Detiene la ejecución de un trabajo (no lo finaliza). Puede ser reanudado mediante fg o bg.

— **ps**. Muestra el listado de procesos en ejecución.

pstree. Muestra el listado de procesos en forma de árbol.

— **top**. Muestra el estado de los procesos de forma dinámica, ordenados en cada instante por uso de CPU.

— **nice y renice**. Cambian la prioridad de un proceso.

9.13.4. Herramientas

— **grep**. Busca un patrón en uno o más archivos. Modificadores: -v, busca las líneas que no encajan con el patrón; -i, no distingue mayúsculas de minúsculas.

- grep wally personas

  Busca el patrón wally en el archivo personas.

- grep 'm*c.' *

  Busca la expresión regular m*c. en todos los archivos. Los resultados podrían ser palabras como: música, músico o marca.

— **fgrep**. Similar a grep. Permite buscar varios patrones. Es más rápida. No admite expresiones regulares.

— **egrep**. Incorpora las ventajas de grep y fgrep.

— **sort**. Ordena las líneas de un archivo.

  • sort nombres1 nombres2 >nombres1-2

    Ordena los archivos nombres1 con nombres2 y deja el resultado en nombres1-2.

  • sort -ur nombres* >finalnombres

    Ordena de forma descendente (-r) y eliminando las líneas repetidas (-u) de todos los archivos cuyo nombre empiece por "nombres" y deja el resultado en el archivo finalnombres.

— **uniq**. Elimina líneas repetidas.

  • sort nombres* | uniq >finalnombres

    Ordena todos los archivos cuyo nombre empiece por nombres, elimina las líneas repetidas y deja el resultado en el archivo finalnombres.

— **cmp y diff**. Comparan 2 archivos.

— **od**. Muestra archivo byte a byte incluyendo caracteres no imprimibles.

— **dc y bc**. Calculadoras.

— **tee**. Descompone la entrada en 2 flujos de salida, una es la salida estándar y la otra, un archivo.

  • ls | tee listado

    Muestra el listado de archivos por pantalla y lo guarda en el archivo llamado listado.

— **cal**. Calendario.

— **date**. Muestra o cambia la fecha y hora del sistema.

9.13.5. Comandos TCP/IP

— **rlogin**. Inicio de sesión remoto.

— **rcp**. Copia remota.

— **rsh**. Shell remoto.

— **ftp**. Cliente de FTP.

— **tftp**. Cliente de TFTP.

— **ping**. Envía un mensaje ICMP de solicitud de eco a otro host. Se utiliza con fines de diagnóstico.

— **traceroute**. Realiza una traza a un host. Indica los routers intermedios hasta alcanzar el host. Equivale a la orden TRACERT de Windows.

— **netstat**. Estadísticas de red.

— **ifconfig**. Configuración TCP/IP.

— **rwho**. Muestra información de los usuarios que hay conectados en las distintas máquinas de la red.

— **finger**. Muestra información de un usuario.

9.13.6. Compartición de archivos

Para acceder a archivos ubicados en máquinas remotas, Unix utiliza la interfaz de archivos distribuidos DFS (*Distributed File Systems*) junto a un sistema de archivos distribuido como puede ser RFS (*Remote File Sharing*, compartición de archivos remotos), desarrollado por los Laboratorios Bell o más frecuentemente, NFS (*Network File System*, sistema de archivos de red), desarrollado por Sun. En el servidor se indican los archivos que se desean compartir. Después, los clientes se conectan a los recursos ofrecidos por el servidor.

— **share**. Comparte un recurso a usuarios de otros sistemas.

  • share -F nfs -o rw -d "informe" /usr/informe

    Comparte el directorio /usr/informe mediante el sistema de archivos NFS, con permisos de lectura y escritura y con el comentario "informe".

  • share

    Muestra los recursos que tenemos compartidos.

— **shareall**. Comparte un conjunto de recursos almacenados en un archivo.

— **unshare**. Deja de compartir un recurso.

— **unshareall**. Deja de compartir todos los recursos.

— **dfmounts**. Monitoriza la utilización de recursos compartidos en un servidor. Indica los usuarios que los están utilizando.

— **dfshares**. Muestra los recursos compartidos en un servidor.

  • dfshares -F nfs pluton

    Muestra los archivos compartidos por NFS en la máquina pluton.

— **mount**. Monta un recurso compartido en un directorio local.

— **mountall**. Monta los recursos indicados en un archivo.

9.13.7. Administración del sistema

— **su.** Permite cambiar de usuario. Al ejecutarlo (*su nombreusuario*) solicitará la contraseña del usuario al cual queremos cambiar y que por tanto debemos conocer su contraseña.

— **sudo.** Permite suplantar temporalmente al superusuario root para poder ejecutar comandos o cambiar configuraciones cuyo propietario de los archivos es root y por tanto solo él puede modificar. Al ejecutarlo solicitará la contraseña del usuario con el que previamente nos hemos logado en el sistema, es decir, no solicitará la de root porque estará restringida a los administradores del sistema. Para que un usuario pueda suplantar a root debe estar declarado como suplantador en el archivo "/etc/sudoers".

— **logout y exit.** Cierran la sesión.

— **shutdown, halt, reboot, init 0.** Apagan el sistema.

— **useradd**. Añade un usuario al sistema. Modificadores: -g, especifica el grupo al que pertenece el usuario; -d, indica el directorio del usuario; -s, shell del usuario; -c, comentario.

  • useradd -m andres

    Crea el usuario andres. Con el modificador -m se crea el directorio /home/andres.

— **passwd**. Cambia contraseña.

— **userdel**. Elimina un usuario.

— **usermod**. Modifica un usuario.

— **groupadd**. Añade un grupo de usuarios.

— **groupdel**. Elimina un grupo de usuarios.

— **groupmod**. Modifica un grupo de usuarios.

9.13.8. Otros comandos

— **wall**. Envía un mensaje a todos los usuarios.

— **news**. Muestra noticias contenidas en el directorio /usr/news. Además de las noticias, existe un archivo llamado /etc/motd (mensaje del día) cuyo contenido se muestra al iniciar sesión.

— **uname**. Muestra datos del sistema (sistema operativo, versión, hardware y nombre del host).

— **who**. Muestra usuarios conectados al sistema y el tiempo de conexión.

— **whoami**. Indica nuestro nombre de usuario.

— **logins**. Muestra un listado de usuarios y grupos a los que pertenecen.

— **fsck**. Chequea el sistema de archivos.

— **df**. Muestra el espacio en disco disponible.

— **du**. Muestra espacio ocupado por los archivos del directorio.

— **wc**. Cuenta caracteres, palabras y frases de un archivo.

— **startx**. Inicia modo gráfico.

10. Sistemas operativos para dispositivos móviles

10.1. Android

Es una plataforma de software para dispositivos móviles, que incluye un sistema operativo y aplicaciones de base.

10.1.1. Características principales

— Núcleo: Linux de estructura monolítica.

— Almacenamiento: BD Ligera SQLite.

— Interfaz de Usuario: Material Design y HOLO anteriores.

— Conectividad: GSM/EDGE, UMTS, Bluetooth, Wi-Fi, LTE, HSDPA, HSPA+...

— Navegador web integrado basado en el motor del proyecto abierto Web Kit.

— Multitáctil para pantallas capacitivas.

— Tethering: actúa como punto de acceso inalámbrico para otros dispositivos usando la conexión de datos.

— Aplicaciones: paquetes con extensión .APK.

— Actualización: a través de Google Play u otras aplicaciones del fabricante del terminal.

— Aplicaciones: Play Store y otras.

— Máquina Virtual Dalvik hasta v.4.3 (Jelly Bean). A partir de la v.4.4 (Kit Kat) ART – Android Run Time.

— Descargables para desarrolladores: SDK Android.

10.1.2. Versiones con nombre de postre

— **A: Apple Pie (v1.0):** tarta de manzana.

- Lanzado el 23 septiembre 2008.
- Es la primera versión, no hay mejoras.
- No se utilizó comercialmente.

— **B: Banana Bread (v1.1):** pan de plátano.

- Lanzado el 9 febrero 2009.
- Corrigieron errores de la 1.0.
- Tampoco se usó comercialmente.

— **C: Cupcake (v1.5):** magdalena.

- Lanzado el 30 abril 2009.
- Basado en el kernel de Linux 2.6.27.
- Posibilidad de grabar y reproducir videos a través del modo *camcorder* (igual que una cámara de video convencional).
- Capacidad de subir videos a YouTube e imágenes a Picasa.
- Teclado con predicción de texto.
- Soporte para Bluetooth.
- Capacidad de conexión automática para conectar a auricular Bluetooth.

— **D: Donut (v1.6):** rosquilla.

- Basado en el kernel de Linux 2.6.29.
- Lanzado el 15 septiembre 2009.
- Actualización de soporte para CDMA/EVDO, 802.1x, VPN y TTS (Text-To-Speech).
- Soporte para resoluciones de pantalla WVGA.

- Mejoras de velocidad en las aplicaciones de búsqueda y cámara.
- Framework de gestos y herramienta de desarrollo GestureBuilder.

— **E: Éclair (v2.0/v2.1):** profiterol.

- Basado en el kernel de Linux 2.6.29.
- Lanzado el 26 octubre 2009.
- Velocidad de hardware optimizada.
- Soporte para más tamaños de pantalla y resoluciones.
- Interfaz de usuario renovada.
- Nuevo interfaz de usuario en el navegador y soporte para HTML5.
- Soporte para Microsoft Exchange.
- Soporte integrado de flash para la cámara.
- Bluetooth 2.1.

— **F: Froyo (v2.2):** yogur helado.

- Basado en el kernel de Linux 2.6.32.
- Lanzado el 20 mayo 2010.
- Optimización general del sistema Android, la memoria y el rendimiento.
- Mejoras en la velocidad de las aplicaciones, gracias a la implementación de JIT.
- Integración del motor JavaScript V8 del Google Chrome en la aplicación Browser.
- Soporte mejorado de Microsoft Exchange (reglas de seguridad, reconocimiento automático, GAL look-up, sincronización de calendario, limpieza remota).
- Lanzador de aplicaciones mejorado con accesos directos a las aplicaciones de teléfono y Browser.
- Funcionalidad de Wi-Fi hotspot y tethering por USB.
- Permite desactivar el tráfico de datos a través de la red del operador.
- Actualización del Market con actualizaciones automáticas.

- Marcación por voz y compartir contactos por Bluetooth.
- Soporte para Adobe Flash 10.1.
- Soporte para pantallas de alto número de Puntos por pulgada, tales como 4" 720p.

— **G: Gingerbread (v2.3):** pan de jengibre.

- Basado en el kernel de Linux 2.6.35.7.
- Lanzado el 6 diciembre 2010.
- Soporte para pantallas extra grandes y resoluciones WXGA y mayores.
- Soporte nativo para telefonía VoIP SIP.
- Soporte para reproducción de videos WebM/VP8 y decodificación de audio AAC.
- Nuevos efectos de audio como reverberación, ecualización, virtualización de los auriculares y refuerzo de graves.
- Soporte para Near Field Communication (=ISO 14443 (RFID, radio-frequency identification).
- Soporte nativo para más sensores (como giroscopios y barómetros).
- Cambio de sistema de archivos de YAFFS a ext4.

— **H: Honeycomb (v3.0/v3.1/v3.2):** panal de miel.

- Lanzado el 22 febrero 2011.
- Mejor soporte para tablets.
- Escritorio 3D con widgets rediseñados.
- Sistema multitarea mejorado.
- Mejoras en el navegador web predeterminado, entre lo que destaca la navegación por pestañas, autorelleno de formularios, sincronización de favoritos con Google Chrome y navegación privada.
- Soporte para videochat mediante Google Talk.
- Mejor soporte para redes Wi-Fi, así como guardar una configuración independiente para cada SSID.
- Añade soporte para una gran variedad de periféricos y accesorios con conexión USB: teclados, ratones, hubs, dispositivos de juego y cámaras digitales. Cuando un accesorio está conectado, el sistema busca la aplicación necesaria y ofrece su ejecución.

— **I: Ice Cream Sandwich (v4.0):** sandwich de helado.

- Lanzado el 19 octubre 2011.
- Versión que unifica el uso en cualquier dispositivo, tanto en teléfonos, tablets, televisores, netbooks, etc.
- Interfaz limpia y moderna llamada "Holo" con una nueva fuente llamada "Roboto", muy al estilo de Honeycomb.
- Soporte de aceleración gráfica por hardware, lo que significa que la interfaz podrá ser manejada y dibujada por la GPU y aumentando notablemente su rapidez, su respuesta y evidentemente, la experiencia de usuario.
- Multitarea mejorada, estilo Honeycomb. Añadiendo la posibilidad de finalizar una tarea simplemente desplazándola fuera de la lista.
- Android Beam es la nueva característica que permite compartir contenido entre teléfonos. Vía NFC *(Near Field Communication)*.
- Reconocimiento de voz del usuario.
- Aplicación de teléfono nuevo con la funcionalidad de buzón de voz visual que permite adelantarlo o retroceder los mensajes de voz.
- Reconocimiento facial, que permite cambiar la vista.
- Soporte nativo del contenedor MKV.
- Soporte nativo para el uso de Stylus (lápiz táctil).

— **J: Jelly Bean (v4.1/v4.2/v4.3):** gragea.

- Lanzado el 9 julio 2012.
- Mejora de la fluidez y de la estabilidad gracias al proyecto "Project Butter".
- Ajuste automático de widgets cuando se añaden al escritorio, cambiando su tamaño y lugar para permitir que los nuevos elementos se puedan colocar.
- Dictado por voz mejorado con posibilidad de utilizarlo sin conexión a Internet.
- Nuevas lenguas no occidentales.
- Android Beam mejorado con posibilidad de transmitir vídeo por NFC.
- Cifrado de aplicaciones.
- Google Chrome se convierte en el navegador por defecto de Android.
- Se pone fin al soporte de Flash Player para Android a partir de esta versión.

— **K: KitKat (v4.4).**

- Compatibilidad con el perfil MAP (*Message Access Profile*, MAP) de Bluetooth por lo que los coches con tecnología Bluetooth pueden intercambiar mensajes con otros dispositivos.
- Compatibilidad con Chromecast: con el dispositivo Android y un Chromecast, se puede acceder a contenido online de Netflix, YouTube, Hulu Plus y Google Play.
- Administración de dispositivos integrada: si se pierde el dispositivo se puede encontrar o borrar con el Administrador de dispositivos Android.
- Conexión por infrarrojos: en los dispositivos que admiten conexiones por infrarrojos (IR), Android admite aplicaciones para controlar el televisor y otros dispositivos cercanos de forma remota.
- Más seguridad en las zonas de pruebas de aplicaciones: se ha incrementado la seguridad de las zonas de pruebas de aplicaciones con el módulo de seguridad Security-Enhanced Linux.
- Podómetro integrado: al utilizar aplicaciones de fitness como Moves en Nexus 5, el teléfono actúa como un podómetro para contar pasos.
- Tocar para pagar: es una nueva forma de pago vía NFC que funciona con cualquier operador móvil y permite que las aplicaciones administren la información sobre pagos en la nube o en cualquier dispositivo.

— **L: Lollipop (v5.0/v5.1):** piruleta.

- Material Design: nueva interfaz de usuario.
- Notificaciones: las llamadas entrantes no interrumpen la visualización ni la reproducción de contenido. Se puede elegir entre responder la llamada o seguir con lo que se estaba haciendo.
- Batería: función de ahorro de energía, alarga la duración de la batería hasta 90 minutos.
- Seguridad:
  - La implementación de SELinux en todas las aplicaciones implica una mayor protección frente a vulnerabilidades y software malicioso.
  - La función Smart Lock de Android permite proteger el teléfono o tablet vinculándolo con un dispositivo de confianza (como un wearable o incluso un coche).
- Conectividad:
  - La mejora de las transferencias entre estaciones produce menos interrupciones en la conectividad. Por ejemplo, se puede continuar con una llamada VoIP o con un chat de

vídeo sin interrupciones cuando se cambia de la conexión Wi-Fi a la de datos móviles.

- Búsqueda eficiente de dispositivos Bluetooth Low Energy ("BLE") cercanos (como modelos o wearables).
- Nuevo modo periférico BLE.

- Tiempo de ejecución y rendimiento:
  - ART totalmente renovado, mejora la respuesta y rendimiento hasta cuatro veces mayor de las aplicaciones.
  - La compatibilidad con dispositivos de 64 bits, como el Nexus 9, lleva las CPU de los ordenadores a Android.
  - Compatibilidad con SoCs de 64 bits con núcleos ARM, x86 y MIPS.
  - Inclusión de aplicaciones nativas de 64 bits como Chrome, Gmail, Calendar y Google Play Music.
  - Ejecución automática de aplicaciones en lenguaje Java puro como aplicaciones de 64 bits.
- Multimedia:
  - La función de mezcla de sonido de varios canales permite que las aplicaciones de audio profesionales puedan combinar hasta ocho canales, incluidos los canales 5.1 y 7.1.
  - Gracias a la función USB Audio, se pueden conectar altavoces, micrófonos y otra gran cantidad de dispositivos de audio USB (como amplificadores y mezcladores) al dispositivo Android.
  - OpenGL ES 3.1 y Android Extension Pack sitúan a Android a la vanguardia de los gráficos para móviles con un rendimiento equiparable al de las consolas y los ordenadores.
  - Android Lollipop incorpora nuevas funciones de fotografía profesionales que permiten:
    * Capturar fotogramas de resolución completa a unos 30 fps.
    * Utilizar formatos sin procesar, como YUV y Mosaico de Bayer.
    * Controlar los ajustes de captura del sensor, la lente y el flash de cada fotograma.
    * Capturar metadatos como modelos de ruido e información óptica.
    * Tecnología de vídeo de última generación compatible con HEVC para permitir la reproducción de vídeo UHD 4K, vídeo redireccionado para reproducir vídeos de alta

calidad en Android TV y mejora de la compatibilidad con HLS para emisión de contenido.

- Ok Google:
  - Reconocimiento de búsquedas por voz.
  - Aunque la pantalla esté desactivada se puede decir "Ok Google" en dispositivos compatibles con el procesamiento digital de señales, como Nexus 6 y Nexus 9.
  - Se pueden hablar con Google desde cualquier lugar para recibir respuestas rápidas, enviar un mensaje de texto, obtener indicaciones, etc.
- Android TV:
  - Se pueden enviar aplicaciones de entretenimiento al televisor con la extensión Google Cast compatible con dispositivos Android TV.
- Accesibilidad:
  - Funciones mejoradas para usuarios con problemas de visión y daltónicos.
  - Aumento del contraste del texto o inversión de los colores para mejorar la legibilidad.
  - Ajuste de la resolución para mejorar la diferencia de color.
- Disponible en más de 68 idiomas.
- Configuración del dispositivo:
  - Rápido encendido que permite utilizar el dispositivo en unos segundos.
  - Se puede configurar el nuevo teléfono o tablet Android al instante juntando ambos dispositivos (se necesita NFC).
  - Permite recuperar las aplicaciones de Google Play automáticamente a través de cualquier dispositivo Android que se haya utilizado anteriormente.
  - Administra varias aplicaciones de pago y pasa rápidamente de una a otra.
  - Permite compartir un archivo con un usuario cercano juntando los dispositivos (Android Beam).

— **M: Marshmallow (v6.0):** nube. A fecha de edición de este libro, esta es la última versión estable.

- Ayuda contextual:

  - Google Now con un toque: se puede obtener ayuda sin tener que dejar lo que se está haciendo, tanto la utilización de una aplicación como en un sitio web, solo con tocar el botón de inicio y mantenerlo pulsado.
  - Haz más cosas con tu voz. Por ejemplo, si un usuario dice "Reproduce música en TuneIn", TuneIn responderá preguntando "¿Qué género?".

- Batería:
  - Función Doze: un modo de sistema que ahorra batería al aplazar actividades de CPU y red de las aplicaciones cuando el dispositivo se encuentra inactivo; por ejemplo, al hallarse sobre una mesa o en un cajón.
  - Descanso: cuando no se está utilizando el dispositivo la función Descanso lo pondrá automáticamente en estado de suspensión para aumentar la duración de la batería en modo de espera.
  - Aplicaciones en espera: permite ahorrar batería con aplicaciones que apenas se utilizan.
  - Compatibilidad con USB tipo C: transfiere rápidamente energía y datos a través del mismo cable. La carga ultrarrápida permite disponer de horas de autonomía en cuestión de minutos.

- Privacidad y seguridad:
  - Las aplicaciones diseñadas para esta versión de Android solo piden permiso cuando es necesario. Se puede denegar cualquier permiso y seguir utilizando la aplicación.
  - Controles avanzados que permiten activar o desactivar permisos para todas las aplicaciones instaladas.
  - Reinicio verificado: cuando se reinicia el dispositivo Android aparece un mensaje de advertencia que indica si se ha modificado la versión de fábrica del firmware y del sistema operativo Android.
  - Biometría: utiliza los sensores de huellas digitales para desbloquear el dispositivo, realizar compras en Google Play, autenticar transacciones en aplicaciones y pagar en tiendas.

- Tiempo de ejecución de Android (ART):
  - Se ha mejorado el rendimiento de las aplicaciones para reducir el consumo de memoria y agilizar la multitarea.

- Productividad:
  - Compatibilidad con lápices ópticos Bluetooth™, incluidas teclas modificadoras y sensibilidad a la presión.

  - Se ha mejorado la composición tipográfica y el rendimiento del renderizado de texto.
  - Acciones de selección de texto, como una nueva opción de traducción que permite traducir texto de un idioma a otro directamente en el sitio.
  - Ahorra papel con la función de impresión a doble cara.
  - Mejoras en la usabilidad del sistema.
  - Permite configurar o activar y desactivar el modo "No molestar" desde los ajustes rápidos.
  - Si alguien llama dos veces en un período de 15 minutos permite seleccionar que la llamada suene mientras está habilitado el modo "No molestar".
  - Los controles de volumen simplificados permiten administrar el volumen de las notificaciones, la música y las alarmas con las teclas de volumen.
  - Los Ajustes simplificados permiten administrar los ajustes de una aplicación en un mismo lugar, desde el consumo de batería y de memoria hasta los controles de permisos y notificaciones.
  - La lista de aplicaciones de Google Now Launcher se ha actualizado mediante desplazamiento alfabético y sugerencias de aplicaciones.
- Conectividad:
  - Búsqueda de bajo consumo de emisores beacon y de accesorios a través de Bluetooth de baja energía.
  - Hotspot 2.0: permite conectarse a redes Wi-Fi compatibles de forma fácil y segura.
  - SAP Bluetooth: permite llamadas desde el teléfono del coche utilizando la tarjeta SIM del teléfono.
  - Las zonas Wi-Fi portátiles admiten bandas de frecuencia de 5 GHz.
  - Espacio de almacenamiento ampliable.
  - Almacenamiento flexible: esta función hace que resulte mucho más fácil utilizar tarjetas SD o dispositivos de almacenamiento externo como almacenamiento cifrado ampliado para aplicaciones y juegos.
- Migración y configuración de dispositivo:
  - Diciendo "Ok Google, configura mi dispositivo" se transfieren cuentas, aplicaciones y datos a un dispositivo nuevo.
  - Mientras se utiliza el asistente de configuración, se puede añadir otra cuenta de correo electrónico, ya sea personal o corporativa (por ejemplo, IMAP).

  - Copia de seguridad automática de aplicaciones: permite realizar copias de seguridad y restauraciones de los datos de las aplicaciones fácilmente.
  - Permite realizar copias de seguridad y restauraciones de ajustes adicionales del sistema, como los ajustes de sincronización, las aplicaciones preferidas, la configuración del modo "No molestar", los ajustes de accesibilidad y los IME habilitados.
- Multimedia:
  - Compatibilidad con MIDI: permite crear, reproducir e interpretar música con el dispositivo mediante dispositivos MIDI a través de USB o de Bluetooth LE y dispositivos MIDI basados en software.
  - Internacionalización.
  - Android ahora está disponible en más de 74 idiomas, con seis novedades.
- Android for Work:
  - Al recibir llamadas o al leer los mensajes recibidos, se pueden ver detalles completos de contacto profesional aunque no se haya iniciado sesión en el perfil de trabajo.
  - Notificación de estado de trabajo: ahora se muestra un icono con forma de maletín en la barra de estado cuando se está utilizando una aplicación desde el perfil de trabajo y, si el dispositivo está desbloqueado directamente en una aplicación en el perfil de trabajo, se mostrará una alerta para informar al usuario.
  - Las aplicaciones de redes VPN ahora se pueden ver en Ajustes > Más > VPN. Además, las notificaciones que utilizan las VPN distinguen entre si esa VPN está configurada para un perfil de trabajo o para todo el dispositivo.

— **N: Nougat (v7.0): turrón**

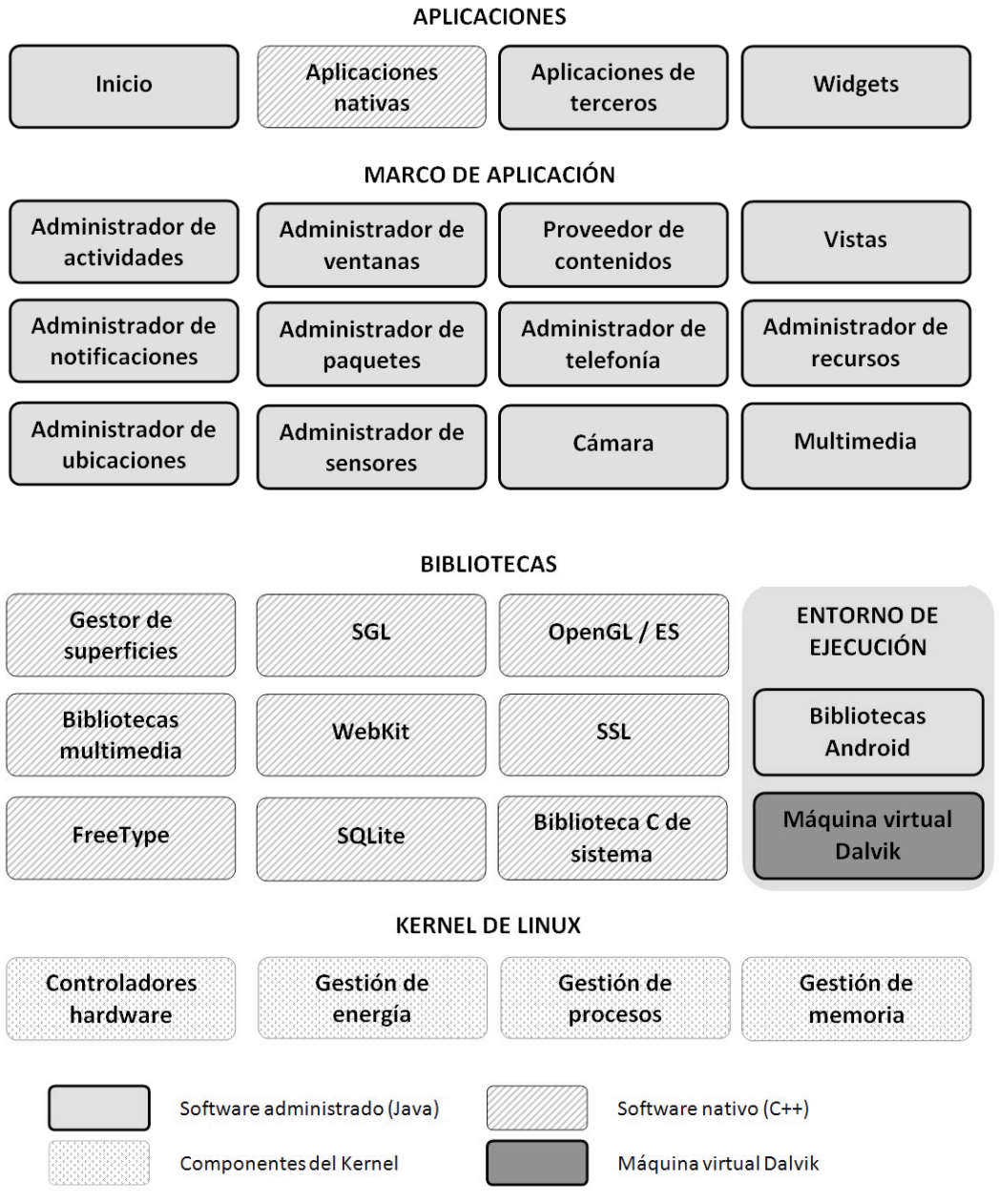
- Compatibilidad con ventanas múltiples: los usuarios ahora pueden abrir dos aplicaciones al mismo tiempo en la pantalla en paralelo una encima de otra en el modo de pantalla dividida. También tienen la posibilidad de modificar el tamaño de las aplicaciones arrastrando la línea divisoria que se encuentra entre ellas.
- Ahorro de datos: es una funcionalidad que, activada, hace que las aplicaciones que se ejecutan en segundo plano no puedan acceder al uso de datos móviles.
- Cambia el tamaño de los elementos de la pantalla. Además del tamaño del texto del dispositivo, podrá cambiar el de los iconos y el resto de elementos que se muestran en la pantalla.
- Control de las notificaciones a nivel de aplicación.
- Cifrado a nivel de archivos.

- En los dispositivos con Android TV, las aplicaciones pueden habilitar en forma automática el modo *"picture-in -picture"*. Esto les permite continuar mostrando contenido mientras el usuario explora otras aplicaciones o interactúa con ellas.
- Compilación de JIT y AOT guiada por perfiles:
  - Agregar un compilador *"Just in Time"* (JIT) con generación de perfiles de código para ART permite mejorar el rendimiento de las aplicaciones mientras se ejecutan. El compilador JIT complementa al compilador *"Ahead of Time"* (AOT) actual de ART y permite mejorar el rendimiento del tiempo de ejecución, ahorrar espacio de almacenamiento y acelerar las actualizaciones de aplicaciones y del sistema.
  - El JIT permite un acceso rápido a la instalación de aplicaciones.
- Doze en movimiento:
  - El modo Doze permite ahorrar batería en movimiento: Cuando la pantalla permanezca apagada durante un tiempo y el dispositivo esté desenchufado, Doze aplicará un subconjunto de restricciones de CPU y red conocidas a las aplicaciones. Esto significa que los usuarios pueden ahorrar batería aun cuando lleven sus dispositivos en los bolsillos.

— **O: Futura versión Android O (v8.0):**

- Actualizaciones más rápidas: Google dividirá el sistema operativo en módulos para reducir el tiempo que necesitan los fabricantes de Smartphone en actualizar sus dispositivos.
- Límites de apps en segundo plano: con un foco en extender la duración de la batería y el rendimiento del dispositivo, Android O ahora integrará límites automáticos para controlar lo que los apps pueden hacer en segundo plano.
- Mejores notificaciones, que permiten definirlas según categorías. Permite también iconos de notificaciones no visibles cuando se despliega el panel, posponer notificaciones.
- Picture in Picture: con esta función tendremos una ventana de vídeo independiente en cualquier parte de la pantalla con la que podremos interactuar sin importar la aplicación en la que estemos.
- Autocompletado en todo el sistema con sugerencias inteligentes en función del texto que se seleccione.
- Mejoras calidad de sonido, debido a que incorpora compatibilidad con códecs de sonido de alta calidad a través de Bluetooth y por otro lado Sony ha habilitado su códec LDAC.
- Mejoras en conectividad con el NAN *(Neighborhood Aware Networking)* funcionalidad a través de WI-FI.

10.1.3. Arquitectura



— **Aplicaciones:** este nivel contiene tanto las incluidas por defecto de Android como aquellas que el usuario vaya añadiendo posteriormente, ya sean de terceras empresas o de su propio desarrollo. Todas estas aplicaciones utilizan los servicios, las API y las librerías de los niveles anteriores.

— **Framework de Aplicaciones:** representa el conjunto de herramientas de desarrollo de cualquier aplicación. Toda aplicación que se desarrolle para Android, ya sean las propias del dispositivo, las desarrolladas por Google o terceras compañías, o incluso las que el propio usuario cree, utilizan el mismo conjunto de API y el mismo "framework", representado por este nivel. Entre las API más importantes ubicadas aquí se pueden encontrar las siguientes:

  - Activity Manager: conjunto de API que gestionan el ciclo de vida de las aplicaciones en Android.
  - Window Manager: gestiona las ventanas de las aplicaciones y utiliza la librería Surface Manager.
  - Telephone Manager: incluye todas las API vinculadas a las funcionalidades propias del teléfono (llamadas, mensajes, etc.).
  - Content Provider: permite a cualquier aplicación compartir sus datos con las demás aplicaciones de Android. Por ejemplo, gracias a esta API la información de contactos, agenda, mensajes, etc. será accesible para otras aplicaciones.
  - View System: proporciona un gran número de elementos para poder construir interfaces de usuario (GUI), como listas, mosaicos, botones, "checkboxes", tamaño de ventanas, control de las interfaces mediante teclado, etc. Incluye también algunas vistas estándar para las funcionalidades más frecuentes.
  - Location Manager: posibilita a las aplicaciones la obtención de información de localización y posicionamiento.
  - Notification Manager: mediante el cual las aplicaciones, usando un mismo formato, comunican al usuario eventos que ocurran durante su ejecución: una llamada entrante, un mensaje recibido, conexión Wi-Fi disponible, ubicación en un punto determinado, etc. Si llevan asociada alguna acción, en Android denominada Intent (por ejemplo, atender una llamada recibida), esta se activa mediante un simple clic.
  - XMPP Service: colección de API para utilizar este protocolo de intercambio de mensajes basado en XML.

— **Librerías:** la siguiente capa se corresponde con las librerías utilizadas. Estas han sido escritas utilizando C/C++ y proporcionan a Android la mayor parte de sus capacidades más características. Junto al núcleo basado en Linux, estas librerías constituyen el corazón de

Android. Entre las librerías más importantes ubicadas aquí, se pueden encontrar las siguientes:

- Librería libc: incluye todas las cabeceras y funciones según el estándar del lenguaje C. Todas las demás librerías se definen en este lenguaje.

- Librería Surface Manager: es la encargada de componer los diferentes elementos de navegación de pantalla. Gestiona también las ventanas pertenecientes a las distintas aplicaciones activas en cada momento.

- OpenGL/SL y SGL: son las librerías gráficas. OpenGL/SL maneja gráficos en 3D y permite utilizar, en caso de que esté disponible en el propio dispositivo móvil, el hardware encargado de proporcionar gráficos 3D. Por otro lado, SGL proporciona gráficos en 2D, por lo que será la librería más habitualmente utilizada por la mayoría de las aplicaciones. Una característica importante de la capacidad gráfica de Android es que es posible desarrollar aplicaciones que combinen gráficos en 3D y 2D.

- Librería Media Libraries: proporciona todos los códecs necesarios para el contenido multimedia soportado en Android (vídeo, audio, imágenes estáticas y animadas, etc.).

- FreeType: permite trabajar de forma rápida y sencilla con distintos tipos de fuentes.

- Librería SSL: posibilita la utilización de dicho protocolo para establecer comunicaciones seguras.

- Librería SQLite: creación y gestión de bases de datos relacionales.

- Librería WebKit: proporciona un motor para las aplicaciones de tipo navegador y forma el núcleo del actual navegador incluido por defecto en la plataforma Android.

— **Tiempo/Entorno de ejecución de Android (ART - Android Runtime):** al mismo nivel que las librerías de Android se sitúa el entorno de ejecución. Este lo constituyen las Core Libraries, que son librerías con multitud de clases Java y la máquina virtual Dalvik (archivos con extensión .dex - dalvik executable).

— **Núcleo Linux:** Android utiliza el núcleo de Linux 2.6 como una capa de abstracción para el hardware disponible en los dispositivos móviles. Esta capa contiene los drivers necesarios para que cualquier componente hardware pueda ser utilizado mediante las llamadas correspondientes. Siempre que un fabricante incluye un nuevo elemento de hardware, lo primero que se debe realizar para que pueda ser utilizado desde Android es crear las librerías de control o drivers necesarios dentro de este kernel de Linux embebido en el propio Android.

10.2. iOS

Es una plataforma de software inicialmente para el iphone, posteriormente para el iPod Touch y finalmente utilizada también en el iPad.

10.2.1. Características

— Sistema Operativo derivado de la familia OS X basado en Darwin BSD y por tanto de Tipo UNIX. Es de estructura híbrido.

— Conectividad: GSM/EDGE, UMTS, Bluetooth, Wi-Fi, LTE, HSDPA, HSPA+...

— Navegador Web por defecto: Safari.

— Multitáctil para pantallas capacitivas.

— Tethering: actua como punto de acceso inalámbrico para otros dispositivos usando la conexión de datos.

— Actualización: a través de itunes.

— Aplicaciones: APP Store.

— Aplicaciones: paquetes con extensión .IPA.

— iOS no permite ni Adobe Flash ni Java.

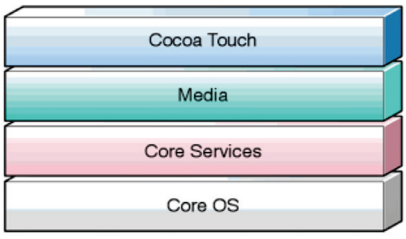
— Descargables para desarrolladores: XCODE.

— Sujeto a las limitaciones impuestas por Apple en dispositivos que utilicen el sistema operativo iOS mediante el uso de kernels modificados. Tales dispositivos incluyen el iPhone, iPod Touch, iPad y la Apple TV de segunda generación. La manera de saltarse estas limitaciones se conoce como jailbreak y permite a los usuarios acceder por completo al sistema operativo, permitiendo al usuario descargar aplicaciones, extensiones y temas que no estén disponibles a través de la App store oficial.

10.2.2. Versiones

— iOS 11 es la última versión estable a fecha de edición de este libro.

10.2.3. Arquitectura



Arquitectura de capas iOS.

— Core OS: la primera es la capa del núcleo del sistema la cual contiene las características de bajo nivel, los archivos del sistema, el manejo del procesador, la memoria, las seguridad, el manejo de archivos, la administración de la energía, en general todo lo referente al hardware del dispositivo. Este sistema operativo está basado en el sistema operativo Unix.

— Core Services: contiene los servicios fundamentales del sistema que usan todas las aplicaciones.

— Capa de Medios: provee los servicios de gráficos y multimedia a la capa superior.

— Capa de Cocoa Touch: esta capa contiene todas las funciones y herramientas para desarrollar aplicaciones para el sistema operativo iOS, posee un conjunto de frameworks que proporciona el API de Cocoa, que provienen de la plataforma del MAC. Aquí se encuentran todas las funcionalidades para desarrollar aplicaciones móviles en iOS, como por ejemplo el acelerómetro, los eventos multi Touch, cámara, localización, entre otros. Esta capa está formada por dos Frameworks fundamentales:

  • UIKit: contiene todas las clases que se necesitan para el desarrollo de una interfaz de usuario.

  • Foundation Framework: define las clases básicas, acceso y manejo de objetos, servicios del sistema operativo.

10.3. Windows Mobile

Windows Mobile es un sistema operativo móvil desarrollado por Microsoft diseñado principalmente para teléfonos inteligentes y tabletas.

10.3.1. Versiones

Windows 10 Mobile que puede venir instalado en Smartphone nuevos o presentarse como una actualización de Windows Phone 8.1, siendo esta la última versión estable a fecha de edición de este libro.

10.3.2. Características de Windows 10 Mobile

— Sistema Operativo: Kernel Híbrido Windows NT.

— Conectividad: GSM/EDGE, UMTS, Bluetooth, Wi-Fi, LTE, HSDPA, HSPA+...

— Interfaz de usuario Continium.

— Navegador Microsoft Edge.

— Multitáctil para pantallas capacitivas.

— Tethering: actúa como punto de acceso inalámbrico para otros dispositivos usando la conexión de datos.

— Aplicaciones: paquetes con extensión .XAP.

— Descargables para desarrolladores: SDK de Windows 10 actúa como emulador proporcionando los encabezados, las librerías, los metadatos y las herramientas más recientes para compilar aplicaciones de Windows 10.

— Aplicaciones: Windows Mobile Store.

— Actualización: Windows Mobile Update.

10.3.3. Arquitectura

— **Aplicaciones:** esta es la capa donde se ejecutan las aplicaciones y sobre la que van a ejecutar las aplicaciones que se desarrollen y que tienen accesibles los servicios ofrecidos por las capas inferiores para, por ejemplo, ejecutar la interfaz de usuario y de manera más indirecta utilizar el acelerómetro o la radio.

  En el siguiente nivel conviven 3 funcionalidades diferentes: Modelo de la aplicación (App Model), Modelo de la Interfaz de Usuario (UI Model) y la integración con servicios en la nube (Cloud Integration).

— **Modelo de la aplicación:** aquí es donde se gestiona la aplicación, sus actualizaciones o donde se comparten datos con otros procesos/aplicaciones.

— **Modelo de la Interfaz de Usuario:** ofrece la posibilidad de crear interfaces de usuario ricas y complejas, basadas en eventos táctiles.

— **Integración con la nube:** mediante esta capa podemos interactuar con diferentes servicios en la nube como: Xbox Live, Bing, Location, Push Notifications. Como ya se ha comentado anteriormente, el sistema basa sus búsquedas en la red en el sistema Bing. Esta es la capa que hace posible esta integración.

— **Kernel:** mediante esta capa tendremos acceso a los servicios que ofrece el hardware del dispositivo, como el GPS, la brújula, el Wi-Fi o el acelerómetro.

10.4. Blackberry BB10

Blackberry OS desarrollado por la empresa canadiense RIM (Research In Motion) para sus dispositivos. El sistema permite multitarea y tiene soporte para diferentes métodos exclusivos de RIM como trackwheel, trackball, touchpad y pantallas táctiles.

10.5. Otros sistemas operativos para móviles

— Firefox OS.

— Replicant.

— Chromium OS.

— Ubuntu para dispositivos BQ.