Introducción a los Sistemas Operativos

Este documento ofrece una aproximación a los sistemas operativos en lo que se refiere a su estructura y funcionamiento interno. Se dará un enfoque general a las funciones que realizan los sistemas operativos dentro de un sistema informático y finalmente se verán un resumen con las características generales de los sistemas operativos más utilizados.

Tabla de contenido

Introducción a los Sistemas Operativos

1 Introducción

El hardware de un sistema informático no tiene capacidad para funcionar por si solo. Necesita de otro elemento, el software, para poder hacerlo. Un sistema operativo (SO) es el software que toma el control del ordenador cuando se enciende y que permite al usuario su uso. Una definición más formal sería

> Un Sistema Operativo es el conjunto de programas que controlan, coordinan y dirigen el uso de los recursos hardware de un sistema informático por las aplicaciones y los usuarios.

El sistema operativo es un administrador de los recursos de hardware del sistema informático y tiene los siguientes objetivos:

1. Proveer un entorno amigable en el cual el usuario pueda ejecutar aplicaciones de manera cómoda, aislándolo de los detalles y complejidades del hardware. El SO hace que el ordenador sea más cómodo de utilizar por los usuarios.
2. Ofrecer una serie de servicios a las aplicaciones de usuario para que los desarrolladores de software se puedan centrar en la lógica de negocio dejando los detalles de más bajo nivel al sistema operativo.
3. Administrar eficientemente los recursos del sistema informático. Un SO permite que los recursos de un sistema informático se aprovechen de una manera más eficiente para conseguir un rendimiento óptimo del ordenador.
4. Evolucionar para dar respuesta a los avances tecnológicos que se producen en el hardware,

de forma que el usuario pueda utilizar ese nuevo hardware asegurando la compatibilidad hacia atrás, es decir, manteniendo sus funciones actuales.

Hoy día es habitual que en un ordenador se ejecuten múltiples aplicaciones de usuario: procesadores de texto, navegadores de Internet, reproducción de vídeo, etc. Estas aplicaciones necesitan el uso de los recursos hardware, como espacio en memoria, ciclos de ejecución en la CPU, acceso a datos en los dispositivos de almacenamiento. Es decir, diferentes aplicaciones compiten por el acceso a los recursos hardware del sistema y el SO se encarga de ofrecer una distribución ordenada y controlada a estos recursos hardware entre las diversas aplicaciones que compiten por ellos.

De lo anterior se deduce que las aplicaciones de usuario no interactúan directamente con el hardware, sino que deben hacerlo a través del sistema operativo. Por tanto, las aplicaciones han de estar programadas específicamente para el sistema operativo que se ejecuta en el ordenador.

1.1 Evolución histórica de los sistemas operativos

En el inicio del desarrollo de los ordenadores, estos no tenían un sistema operativo. Estos primeros ordenadores se diseñaban específicamente para realizar un tipo de tarea muy concreto y las aplicaciones que ejecutaban era codificadas por los usuarios en lenguaje máquina e introducidas dentro del ordenador. Todo esto estaba encuadrado en proyectos científicos y militares que tenían un control absoluto sobre todo el trabajo.

Sin embargo, años después, la evolución de la informática permitió a algunas empresas tecnológicas producir ordenadores de propósito general para su venta a otras empresas. Fue aquí donde surgió la necesidad de dotar al ordenador de un software básico que permitiera su uso. Este primer software básico supuso el primer sistema operativo.

A lo largo de la historia los SO han evolucionado para adaptarse a los cambios tecnológicos que se iban produciendo en los ordenadores y así poder aprovechar estos avances tecnológicos.

1.1.1 Década de 1950

En los ordenadores de los años 50 había un componente software que se encargaba de cargar un programa junto con sus datos (un trabajo) en la memoria del ordenador desde un dispositivo de entrada como una cinta magnética o un conjunto de tarjetas perforadas. Este componente software se conocía como *Monitor Residente* y puede considerarse el primer sistema operativo.

El problema es que al terminar de ejecutar un trabajo se tardaba demasiado en cargar el siguiente. Para solucionar este problema se crea el concepto de *procesamiento por lotes* o *batch* que consiste en grabar en una cinta o tarjetas perforadas un conjunto de trabajos que se cargaban juntos y se ejecutaban uno detrás de otro.

Otro avance que se produjo en esta época para aumentar la productividad consistió en poder cargar un trabajo o dar salida a los datos mientras se ejecutaba otro.

1.1.2 Década de 1960

La característica de los sistemas operativos en esta época fue el desarrollo del concepto de *multiprogramación*, y los principios del *multiprocesamiento*. En los sistemas de multiprogramación, varios programas de usuario se encuentran al mismo tiempo en memoria, y la CPU cambia

rápidamente de un trabajo a otro cuando el primero queda bloqueado por una operación de E/S. En los sistemas de multiprocesamiento se utilizan varias CPUs en un solo ordenador, con la finalidad de incrementar la capacidad de procesamiento de la maquina.

Se desarrolló el concepto de *tiempo compartido*, en la que los usuarios podían conectarse directamente al ordenador a través de terminales (teclado y pantalla). Surgieron sistemas de tiempo real, en que los ordenadores fueron utilizados en el control de procesos industriales. Los sistemas de tiempo real se caracterizan por proveer una respuesta a un evento dentro de un periodo de tiempo límite establecido de antemano.

1.1.3 Década de 1970

Se inicia con la introducción de la familia de ordenadores Sistema/360 de IBM. Los ordenadores de esta generación fueron diseñados como sistemas de propósito general, es decir, podían ejecutar diversos tipos de aplicaciones.

En 1969 Ken Thompson, Dennis Ritchie y Douglas McIlroy crean el sistema operativo UNIX en los Laboratorios Bell. También aparece un ordenador personal, el Xerox Alto que tenía su propio sistema operativo.

1.1.4 Década de 1980

En 1979 Tim Paterson crea su sistema operativo 86-DOS, que posteriormente pasó a llamarse QDOS (*Quick and Dirty Operative System*). Dos años después Bill Gates compra QDOS y lo rebautiza dos veces: en primer lugar como PC-DOS, el cual vende como sistema operativo a IBM para su IBM-PC, y un año más tarde como MS-DOS, el cual, siendo una copia casi idéntica a PC-DOS. Como consecuencia de la aparición del PC con su sistema operativo el porcentaje de la población que tiene acceso a un ordenador es mucho mayor que nunca y aumenta rápidamente.

En 1983 Apple crea el Apple Lisa System 1 y un año después el Mac OS. En 1985 aparece la primera versión de Windows, aunque por aquél entonces no era un sistema operativo sino un entorno gráfico que se ejecutaba bajo MS-DOS.

En 1987 Andrew S. Tanenbaum crea MINIX, un sistema operativo basado en Unix y escrito en lenguaje C, cuyo principal objetivo era aprender como funciona un sistema operativo por dentro. Este sistema llevó a Linus Torvalds para la creación del núcleo Linux.

Un interesante desarrollo que comenzó a llevarse a cabo a mediados de la década de los ochenta ha sido el crecimiento de las redes de ordenadores, con sistemas operativos de red y sistemas operativos distribuidos. En los sistemas operativos de red, los usuarios conectan sus PCs con otros remotos y copiar archivos de una maquina a otra. Cada maquina ejecuta su propio sistema operativo local y tiene su propio usuario.

Por el contrario, un sistema operativo distribuido es aquel que aparece ante sus usuarios como un sistema tradicional de un solo procesador, aun cuando esta compuesto por varios procesadores. En un sistema distribuido verdadero, los usuarios no deben ser conscientes del lugar donde su programa se ejecute o de lugar donde se encuentren sus archivos; eso debe ser manejado en forma automática y eficaz por el sistema operativo.

1.1.5 Década de 1990

En esta generación aparecen los sistemas operativos con interfaces gráficas de usuarios (GUI). En 1990 aparece Windows v3 y en 1995 Windows 95 de Microsoft. Ambos siguen siendo entornos gráficos para ser ejecutados bajo el sistema operativo MS-DOS en su última versión, la 6. En 1998 aparece Windows 98 como sistema operativo con GUI para PC.

En 1990 Richard Stallman crea el sistema GNU de software libre y el Linus Torvalds crearía un año después el núcleo Linux. En 1992, el sistema GNU y el Núcleo Linux se unen formalmente para crear GNU/Linux.

1.1.6 Actualidad

Los sistemas operativos evolucionan con nuevas interfaces de usuario, como la pantalla táctil, para nuevos dispositivos: PC's, portátiles, tablets, smartphones. Aparecen los sistemas operativos en la nube o sistemas operativos en la web (WebOS). Consisten en un escritorio virtual con aplicaciones integradas que permiten al usuario administrar sus datos sin necesidad de instalar aplicaciones. Todo ello mediante una conexión a Internet.

2 Clasificación de los sistemas operativos

Los sistemas operativos pueden clasificarse en función de diferentes criterios. Los más relevantes son los siguientes:

2.1 Tiempo de respuesta

El tiempo de respuesta es la cantidad de tiempo que transcurre desde que el usuario lanza el proceso hasta que obtiene los resultados. Según esto, el sistema operativo puede ser:

- ✔ Proceso por lotes.-El sistema operativo ejecuta un conjunto de trabajos, denominado lote, uno detrás de otro. El resultado se presenta al usuario cuando termina todos los trabajos. El usuario no interviene ya que los datos para realizar los trabajos están en el propio trabajo.
- ✔ Interactivo.- Existen interacción con el usuario. El SO ejecuta un trabajo y cuando necesita un dato interrumpe su ejecución para solicitarlo al usuario. Este se lo suministra por un dispositivo de E/S y el SO continúa con su ejecución hasta que necesite otro dato o finaliza. Para optimizar el uso de recursos el SO utiliza el tiempo compartido, de forma que puede repartir su tiempo entre varios trabajos.
- ✔ Tiempo real.- El SO debe ofrecer un tiempo de respuesta dentro de un plazo definido de antemano que generalmente suele ser bajo. Este tipo de SO se suele utilizar donde el tiempo de respuesta es crítico, como en procesos industriales o tráfico aéreo.

2.2 Número de usuarios

Dependiendo del número de usuarios que pueden utilizar simultáneamente el sistema informático, los sistemas operativos pueden ser:

- ✔ Monousuario.- En este SO solamente un usuario puede utilizar el sistema informático para el que están disponibles todos los procesos.

- ✔ Multiusuario.- Un SO multiusuario provee servicio a múltiples usuarios simultáneamente. Actualmente este tipo de sistemas se emplean especialmente en redes, pero los primeros sistemas multiusuario fueron sistemas centralizados que se compartían a través del uso de múltiples terminales.

2.3 Número de procesos

Este criterio indica el número de proceso simultáneos que puede gestionar un SO. En función de esto los sistemas operativos pueden ser:

- ✔ Monotarea.- Un SO es monotarea cuando solamente ejecuta un programa a la vez. La CPU ejecuta un programa y hasta que no acaba o ejecuta otro. Estos sistemas operativos están en desuso, como por ejemplo el MS-DOS.
- ✔ Multitarea.- Se pueden ejecutar varios procesos a la vez. Si el sistema informático solamente tiene una CPU, el SO dividirá su tiempo entre todos los procesos intentando maximizar el uso de la CPU. Los procesos en ejecución deberán permanecer en memoria principal mientras se ejecuten.

2.4 Número de procesadores

Indica el número de CPU que tenga el sistema informático. En función de este criterio existen los siguientes tipos de sistemas operativos:

- ✔ Monoproceso.- Un SO monoproceso solamente puede trabajar con un sistema informático que tenga una única CPU.
- ✔ Multiproceso.- Un SO multiproceso puede trabajar con varias CPUs con lo que podrá ejecutar varios procesos de manera simultánea.

3 Estructura de un Sistema Operativo

La siguiente tabla muestra una clasificación de los programas que forman un SO

Nombre	Descripción
Núcleo o kernel	Se encarga de interactuar con el hardware. Es la única parte del SO que tiene acceso directo al hardware.
Llamadas al sistema o primitivas	Funciones que invocan las aplicaciones de usuario para solicitar algún servicio al SO. Cada SO implementa un conjunto propio de llamadas al sistema. Ese conjunto de llamadas es la interfaz del SO frente a las aplicaciones. Constituyen el lenguaje que deben usar las aplicaciones para comunicarse con el SO.
Servidor	Programa que utiliza el kernel para suministrar algún servicio a las aplicaciones de usuario. Los servidores se emplean en estructuras microkernel que se verá a continuación.
Interrupciones y	El SO ocupa una posición intermedia entre los programas de aplicación y el

excepciones	hardware. No se limita a utilizar el hardware a petición de las aplicaciones ya que hay situaciones en las que es el hardware el que necesita que se ejecute código del SO. En tales situaciones el hardware debe poder llamar al SO, pudiendo deberse estas llamadas a dos condiciones: • Algún dispositivo de E/S necesita atención. • Se ha producido una situación de error al intentar ejecutar una instrucción del programa (normalmente de la aplicación). En ambos casos, la acción realizada no está ordenada por el programa de aplicación, es decir, no figura en el programa. Según los dos casos anteriores tenemos las interrupciones y la excepciones: • Interrupción: señal que envía un dispositivo de E/S a la CPU para indicar que la operación de la que se estaba ocupando, ya ha terminado. • Excepción: una situación de error detectada por la CPU mientras ejecutaba una instrucción, que requiere tratamiento por parte del SO.
Utilidades	Todos los SO incluyen un conjunto de programas para resolver pequeñas tareas de los usuarios, como el navegador de Internet, el explorador de archivos, accesorios, y también un conjunto de programas de ayuda a la programación, como editores, compiladores, enlazadores, etc.

El objetivo de la estructuración de un SO es buscar una organización interna que facilite la comprensión, incremente la portabilidad, extensión y favorecer su mantenimiento.

A continuación se describen las distintas estructuras que presentan los actuales sistemas operativos para satisfacer las necesidades que de ellos se quieren obtener. Éstas no son de ninguna manera las únicas posibles, pero nos darán una idea de algunos diseños que se han llevado a la práctica.

3.1 Sistema monolítico

Los sistemas monolíticos son la estructura más simple para un SO. para proporcionar una máxima funcionalidad dentro del menor espacio posible. Se caracteriza porque no tienen una estructura totalmente clara, con esto nos referimos a que sus rutinas y funcionalidades (manejo de drivers, sistemas de archivos, gestión de memoria, etc.), se encuentran agrupados en un solo programa.

Este sistema está descrito como un conjunto de procedimientos o rutinas entrelazadas de tal forma que cada una tiene la posibilidad de llamar a las otras rutinas cada vez que así lo requiera. Sin embargo, cabe destacar los defectos en este tipo de estructura que radica principalmente en la poca fiabilidad otorgada, ya que todo el sistema, al no tener una estructura definida, se ejecuta todo en el mismo nivel del núcleo (kernel) lo que lo hace altamente vulnerable, por esta razón cuando falla un programa se produce un error en todo el sistema.

Además, otro problema inherente al sistema monolítico es que si se modifica el hardware por lo general es necesario recompilar el kernel para poder disponer de las funcionalidades. Esto consume

tiempo y recursos porque la compilación de un nuevo kernel puede durar varias horas y necesita de una gran cantidad de memoria. Cada vez que alguien añade una nueva característica o corrige un error, significa que se necesitará hacer una recompilación del kernel entero, un ejemplo de esto podemos verlo en Linux. También el hecho de que en el espacio del kernel están incluidos todos los servicios básicos.

Este tipo de SO tiene tres grandes inconvenientes: el tamaño del núcleo, la falta de extensibilidad y la dificultad de mantenimiento.

Ejemplos de este tipo de estructura son Unix, MS-DOS, Mac OS (hasta v8.6), Linux, Windows 95, 98, 98SE, Me.

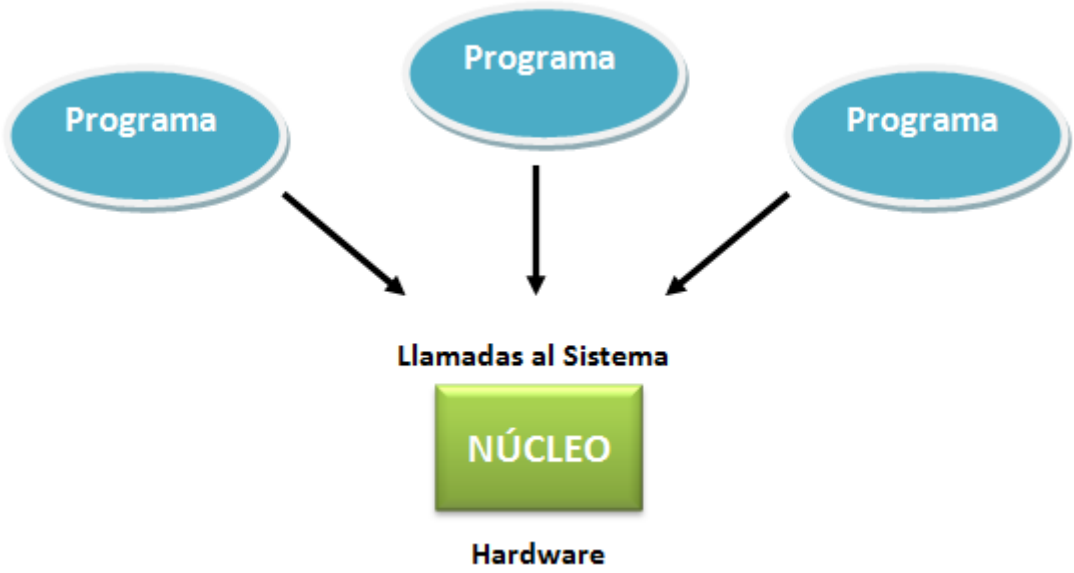

Figura 1.- Sistema monolítico

3.2 Sistemas microkernel

El microkernel es un enfoque que define un kernel muy simple (microkernel) que trabaja sobre el hardware. Solo las funciones absolutamente esenciales del sistema operativo deben permanecer en el microkernel. Los servicios y las aplicaciones menos esenciales se construyen sobre el microkernel y se ejecutan en modo usuario.

Este microkernel tiene un conjunto de primitivas y llamadas al sistema para implementar unos servicios mínimos del SO como comunicación entre procesos, gestión de memoria y planificación de la CPU. Los demás servicios que solían ser suministrados por el kernel, como por ejemplo el servicio de red, se implementan como programas de usuario conocidos como servidores.

En teoría este enfoque añade más estabilidad al sistema por que un fallo de un servidor solamente pararía un programa simple, en lugar de provocar la parada del SO completo. La siguiente figura muestra las diferencias entre un enfoque monolítico y el microkernel.

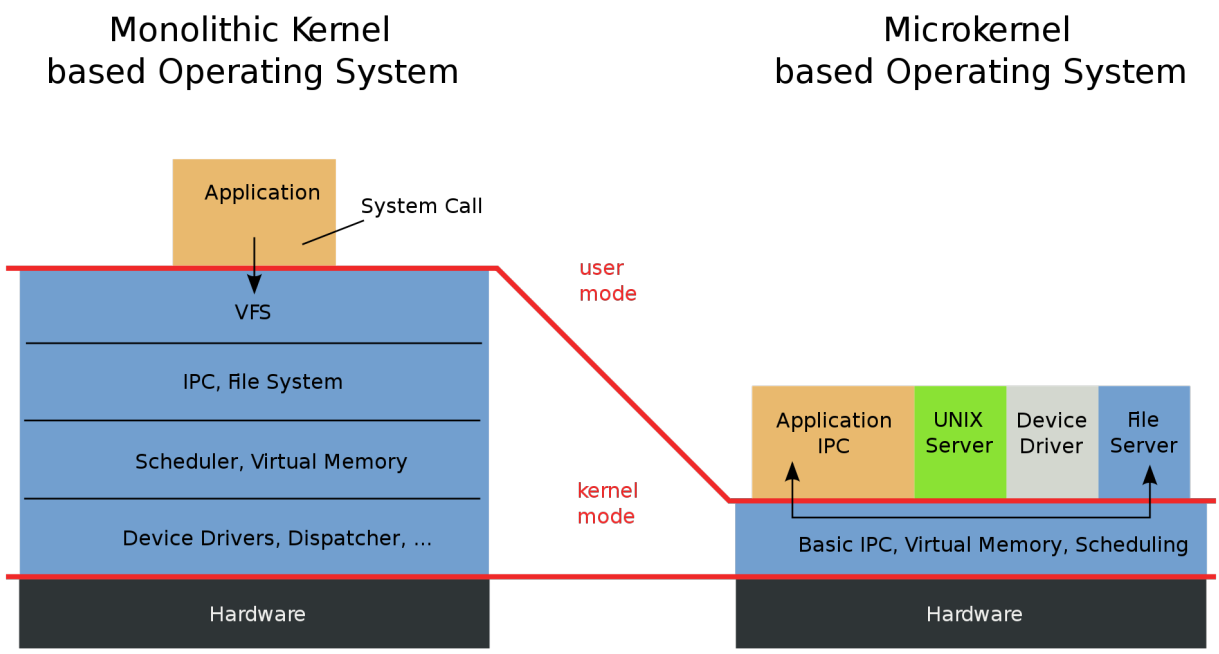

Figura 2.- Comparativa kernel monolítico y microkernel

En la figura anterior se aprecia que el kernel de un sistema monolítico es mayor. Todos los servicios que ofrece el SO a las aplicaciones de usuario se ejecutan dentro del núcleo. Sin embargo el kernel de un sistema micronúcleo solamente incluye funciones básicas de intercomunicación de procesos, asignación de memoria y planificación de procesos. El resto de servicios se implementan con programas servidores.

Un ejemplo de sistema operativo con esta estructura es Minix, desarrollado por Andrew S. Tanembaum.

3.3 Sistema por capas

En esta estructura el SO se divide en capas o niveles, cuya organización está dada como una jerarquía de capas donde cada una de ellas ofrece una interfaz clara y bien definida, la capa superior solamente utiliza los servicios y funciones que ofrece la capa inferior, es decir, la capa n sólo se comunica para obtener lo requerido con la capa n-1, donde la capa inferior es la más privilegiada. El encargado de que solamente haya comunicación entre capas adyacentes es el procesador.

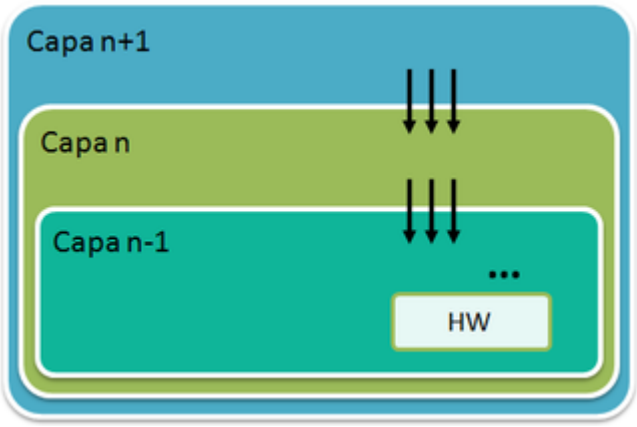

Figura 3.- Estructura por capas

La capa más interna o inferior (capa 0) corresponde al hardware, mientras que la más alta o externa corresponde a la interfaz de usuario. El sistema original consta de 6 capas:

- ✔ Capa 5: Se encuentra la interfaz de usuario.
- ✔ Capa 4: Aloja los programas de usuario.

- ✔ Capa 3: Se controlan los dispositivos E/S (entrada y salida).
- ✔ Capa 2: Se administra la comunicación entre procesos y la consola del operador.
- ✔ Capa 1: Administración de memoria y discos.
- ✔ Capa 0: Correspondiente al hardware, realizando asignación del procesador, también alterna entre procesos cuando ocurren interrupciones o se han expirado y proporciona multiprogramación básica de la CPU.

Como ventajas de este sistema podemos mencionar que al tener una organización por módulos, otorga facilidad en construcción y depuración del sistema. La facilidad de construcción se consigue al existir esta división en módulos (capas) se produce una abstracción del problema, simplificándose solamente a la función que realiza el módulo correspondiente a una capa N. También al lograr esta abstracción, no es necesario saber detalles de implementación de las capas inferiores, sólo se utilizan. La facilidad de depuración, quiere decir, que sea más simple la tarea de encontrar errores en el código y corregirlos. Otro aspecto positivo relacionado con la modularidad existente, cuando ocurre un error en una de las capas, no se compromete a todo el sistema, sólo a la capa relacionada con el error.

Con respecto a las desventajas de esta organización, al realizar la construcción de las capas, la problemática es la forma de realizar la división y definición de las funcionalidades, ya que se tiene considerar que las capas superiores solamente pueden utilizar los servicios de la capa que se encuentra inferior, por lo tanto, se debe tener mucho cuidado en la planificación del sistema para que exista un óptimo funcionamiento. Otra desventaja que podemos mencionar es el gasto de tiempo que se genera en ir de una capa a otra.

3.4 Sistema por módulos

La mayoría de los sistemas operativos modernos implementan este enfoque. Lo que caracteriza este tipo de estructura es que el kernel se compone de módulos, y cada uno de estos módulos se encuentra separado de forma independiente, tal que, si alguno falla no afecta a los otros, ni al núcleo. Los módulos se pueden cargar dinámicamente en el núcleo cuando se necesiten, es decir, en tiempo de ejecución o durante el arranque del sistema. El kernel dispone de los componentes fundamentales y se conectan directamente con servicios adicionales. Además otros componentes pueden cargarse dinámicamente al núcleo.

En general, esta estructura se parece bastante a la de capas, pero es mucho más flexible debido a que cualquier módulo de esta estructura puede llamar a otro.

Ejemplos de este tipo de estructura son Unix modernos, Solaris, Linux, Mac OSX.

4 Funciones de un sistema operativo

Un SO multiusuario proporciona un entorno dentro del cual se ejecutan las aplicaciones. Para construir este entorno se divide lógicamente al SO en pequeños módulos y se crea una interfaz bien definida entre estos módulos y las aplicaciones que se ejecutan. Cada uno de estos módulos se encarga de realizar una función específica. Entre estas están:

4.1 Administrador de procesos

La CPU ejecuta un gran número de aplicaciones, y aunque su preocupación principal es la ejecución de aplicaciones de usuario, hay otras aplicaciones cuya ejecución también es necesaria para otras actividades del sistema. Un proceso es un programa en ejecución con su entorno asociado. Suponemos que el proceso se ejecuta secuencialmente un programa puede generar varios procesos activos.

En general, para cumplir con su tarea un proceso necesita determinados recursos como tiempo de la CPU, memoria, archivos y acceso a dispositivos de E/S. Estos recursos se proporcionan al crear el proceso, o se le asignan mientras se ejecuta. Potencialmente, todos estos procesos pueden ejecutarse en forma concurrente, multiplexando la CPU entre ellos. El SO es responsable de las actividades relacionadas con la administración de procesos:

- ✔ Crear y eliminar procesos.
- ✔ Suspender y reanudar la ejecución de procesos.
- ✔ Sincronización de procesos.
- ✔ Comunicación entre procesos.
- ✔ Análisis de interbloqueos.

4.2 Administrador de la memoria principal

La memoria es el dispositivo de almacenamiento principal en un sistema informático. El procesador lee las instrucciones y lee o escribe datos de la memoria principal. Por otra parte, los dispositivos de E/S necesitan acceso a memoria para lectura y escritura de datos.

Para mejorar la utilización de la CPU y la velocidad de respuesta del ordenador a los usuarios, se debe conservar varios programas en memoria. Las actividades relacionadas con la administración de memoria son las siguientes:

- ✔ Llevar el control de las zonas de memoria usadas y quien las usa.
- ✔ Decidir que procesos se cargan en memoria habiendo espacio libre.
- ✔ Asignar y recuperar espacio de memoria.

4.3 Administrador del almacenamiento secundario

Cuando la memoria principal sea pequeña para almacenar todos los programas y los datos, el sistema informático debe ofrecer almacenamiento secundario que la respalde, es lo que llamamos memoria secundaria. Este almacenamiento se hace en general mediante disco donde se sitúan habitualmente las aplicaciones que se cargan en memoria principal solo cuando se ejecutan. Respecto a la memoria secundaria las funciones van a ser:

- ✔ Administración del espacio libre en dispositivos de almacenamiento.
- ✔ Asignación del almacenamiento.
- ✔ Planificación de las operaciones sobre el disco.

Una de las operaciones principales del sistema será resolver las operaciones de carga y descarga de información desde el almacenamiento secundario a la memoria principal.

4.4 Gestión de ficheros y directorios

Es una de las partes más visibles del SO por parte del usuario. Los ordenadores pueden almacenar información en varios dispositivos físicos, siendo los más comunes el disco duro, el disco óptico y las memorias flash. Cada uno de estos dispositivos tiene sus propias características y organización física.

Para usar cómodamente un sistema informático, el SO ofrece una perspectiva lógica uniforme del almacenamiento de información. El SO se vale de las propiedades físicas de sus dispositivos de almacenamiento para definir una unidad de almacenamiento lógico, el archivo o fichero. Por medio del SO, los archivos se relacionan con el dispositivo físico.

Un archivo es un conjunto de información relacionada y que se almacena y manipula por el sistema operativo como una entidad única. Comúnmente los archivos contienen programas y datos. Los archivos se organizan en directorios para facilitar su acceso. Como varios usuarios tienen acceso a ellos, es deseable controlar quién tiene acceso a los archivos y cómo puede hacerlo.

El SO es responsable de las siguientes actividades relacionadas con la administración de archivos:

- ✔ Crear y eliminar archivos.
- ✔ Crear y eliminar directorios.
- ✔ Control de operaciones para manipular archivos y directorios.
- ✔ Correspondencia entre archivos y almacenamiento secundario.
- ✔ Copia de seguridad de archivos en medios de almacenamiento estables.

4.5 Administrador de la E/S

Un sistema informático tiene una diversidad de dispositivos periféricos conectados. Uno de los objetivos del SO es ocultar al usuario las particularidades que estos dispositivos hardware y por tanto sus funciones básicas son:

- ✔ Un sistema de memoria caché mediante buffer.
- ✔ Una interfaz general con los controladores de dispositivo y unos controladores para dispositivos hardware específicos.

4.6 Protección

La protección se refiere al control del acceso de las aplicaciones y usuarios a los recursos hardware de un sistema informático. El SO tiene la responsabilidad de proteger un proceso de los otros. Por tanto debe asegurar que los ficheros, segmentos de memoria, CPU y otros recursos de E/S puedan ser únicamente usados por aquellos procesos que hayan recibido la correspondiente autorización del sistema. Por ejemplo, el direccionamiento de memoria asegura que un proceso sólo puede trabajar dentro del espacio de direcciones asignado a ese proceso. Hay que asegurar que

ningún proceso pueda obtener el control de la CPU sin que lo acapare indefinidamente. Por último, no se permite que los usuarios realicen por su cuenta sus operaciones de E/S, para proteger así la integridad de los dispositivos periféricos.

5 Interfaces de usuario en un SO

Actualmente, hay dos paradigmas fundamentales en el uso del SO: modo comando o intérprete de la línea de comando y modo gráfico o la interfaz gráfica de usuario (GUI), aunque la tendencia actual es combinar ambos paradigmas. Esto quiere decir que los sistemas operativos para usuario ofrecen una interfaz gráfica para que el usuario profano en informática pueda interactuar con el SO y además la posibilidad de utilizar una interfaz solo texto para introducir comandos ejecutables. Veamos con más detalle ambos paradigmas.

5.1 Intérprete de comandos

Antes de la aparición de las interfaces gráficas, los usuarios se comunicaban con el SO a través de comandos. Un comando (traducción literal del inglés *command*, «orden, instrucción») es una instrucción u orden que el usuario proporciona al SO, desde la línea de comandos (*shell*) o desde una llamada de programación. Estos comandos pueden ser:

- ✔ Interno.- El código del comando está contenido en el propio archivo del intérprete.
- ✔ Externo.- Cuando el código del comando no está en el archivo del intérprete, sino en un archivo ejecutable aparte.

Suele admitir parámetros o argumentos de entrada, lo que permite modificar su comportamiento predeterminado. Suelen indicarse tras una barra "/" (en sistemas operativos Windows) o un guión simple "-" o doble "--" (en sistemas operativos Unix/Linux).

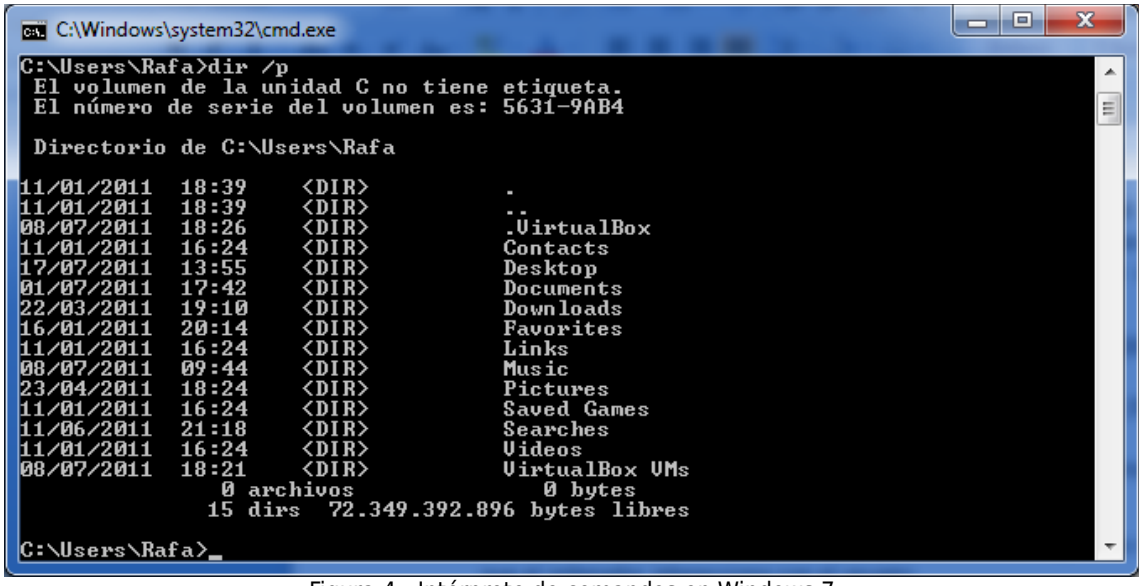

Figura 4.- Intérprete de comandos en Windows 7

Esta interacción en la ejecución de un comando la realiza un programa que se denomina intérprete de comandos. Este permite al usuario escribir comandos del sistema operativo y al pulsar la tecla Intro en el teclado, se ejecuten procesos internos al propio intérprete de comandos o se

lancen programas externos al mismo. Algunas de estas consolas son:

- ✔ **command.com** para los sistemas basados en DOS (MS-DOS, PC-DOS, etc.). Ya está en desuso.
- ✔ **cmd.exe** para los sistemas basados en Windows con la aplicación *Símbolo del sistema*.
- ✔ PowerShell para Windows 8 y Server 2008 o superior.
- ✔ **bash**, **sh**, **csh**, **ksh**, etc. para los sistemas basados en Unix y GNU/Linux.

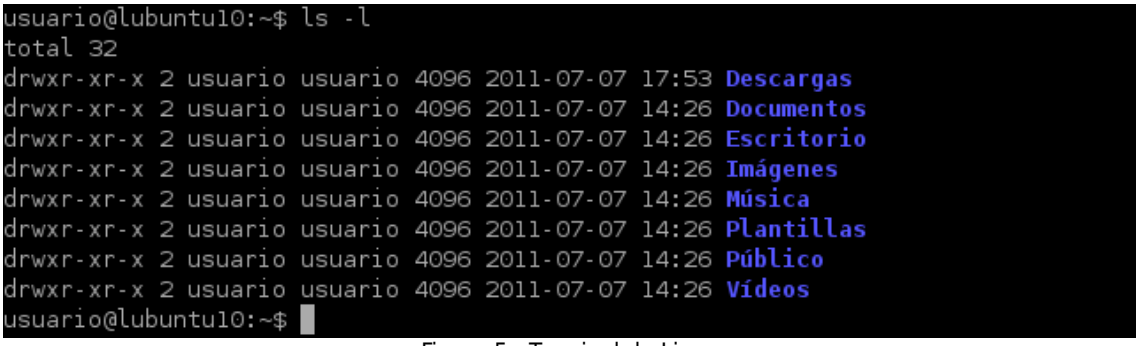

Figura 5.- Terminal de Linux

Este programa tiene dos formas de implementarse. En primer lugar, el intérprete de comandos contiene en sí el código para ejecutar el comando que se desea ejecutar. Por ejemplo, un comando para borrar un fichero puede hacer que el intérprete salte a una sección de su propio código que establece los parámetros necesarios y realiza la operación. En este caso, el número de comandos que pueden darse determina el tamaño del intérprete de comandos, puesto que cada comando requiere su propio fragmento de código del intérprete.

Un enfoque alternativo implementa todos los comandos mediante programas de sistema especiales y se almacenan cada uno en un archivo. En este caso, el intérprete de comandos no "comprende" el comando; simplemente utiliza el comando para identificar un fichero a cargar en memoria y ejecutar. Por ejemplo, un comando delete G buscaría un fichero llamado delete, el cual contiene el código para borrar al fichero G, lo cargaría en memoria y le pasaría el parámetro G. La función asociada con el comando delete estaría completamente definida por el código del fichero delete. De esta manera pueden añadirse fácilmente nuevos comandos al sistema, creando nuevos ficheros con el nombre adecuado. El programa intérprete de comandos, que en este caso puede ser bastante pequeño, no necesita ser modificado para añadir comandos nuevos.

5.2 GUI

La interfaz gráfica de usuario, conocida también como GUI (*Graphical User Interface*) es un programa informático que actúa de interfaz de usuario, utilizando un conjunto de imágenes y objetos gráficos para representar la información y acciones disponibles en la interfaz. Su principal uso, consiste en proporcionar un entorno visual sencillo para permitir la comunicación con el sistema operativo de un ordenador a los usuarios profanos en informática.

Habitualmente las acciones se realizan mediante manipulación directa para facilitar la interacción del usuario con el ordenador. Como ejemplos de interfaz gráfica de usuario, cabe citar los entornos de escritorio Windows, el X-Window de GNU/Linux o el de Mac OS X.

Figura 6.- GUI del Mac OSX

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 como los iconos y las listas de elementos del menú.

Para los autores de aplicaciones, las interfaces gráficas de usuario ofrecen un entorno que se encarga de la comunicación con el ordenador. Esto hace que el programador pueda concentrarse en la funcionalidad, ya que no esta sujeto a los detalles de la visualización ni a la entrada a través del ratón o el teclado. También permite a los programadores crear programas que realicen de la misma forma las tareas más frecuentes, como guardar un archivo, porque la interfaz proporciona mecanismos estándar de control como ventanas y cuadros de diálogo. Otra ventaja es que las aplicaciones escritas para una interfaz gráfica de usuario son independientes de los dispositivos: a medida que la interfaz cambia para permitir el uso de nuevos dispositivos de entrada y salida, como un monitor de pantalla grande o un dispositivo óptico de almacenamiento, las aplicaciones pueden utilizarlos sin necesidad de cambios.

6 Bibliografía

WIKIPEDIA, *Historia de los sistemas operativos* [acceso septiembre 2014]. Disponible en <http://es.wikipedia.org/wiki/Historia_de_los_sistemas_operativos>

WIKIPEDIA, *Estructura de los sistemas operativos* [acceso septiembre 2014]. Disponible en <http://wiki.inf.utfsm.cl/index.php?title=Estructura_de_un_sistema_operativo>

WIKIPEDIA, *Sistema Operativo* [acceso septiembre 2014]. Disponible en <http://es.wikipedia.org/wiki/Sistema_operativo>

WIKIBOOKS, *Operating System Design/Kernel Architecture/Microkernel* [acceso septiembre 2014]. Disponible en <http://en.wikibooks.org/wiki/Operating_System_Design/Kernel_Architecture/Microkernel>

TANENBAUM, A. *Sistemas operativos modernos – 3ª Edición. 2009* Autor - Editor ISBN 9786074420463