ITEDES

Educación Digital

Módulo:

Fundamentos de Ingeniería de Software

Segmento:

Elementos Informáticos

Tema:

Sistemas Operativos

Prof. Germán C. Basisty
german.basisty@itedes.com

Índice de Contenidos

¿Qué es un Sistema Operativo?

Contextualización

En cencias computacionales se denomina **sistema operativo** al conjunto de programas que gestiona los recursos físicos de un sistema informático (memoria, capacidad de procesamiento, espacio en disco duro, acceso a la red, etc.) y provee servicios al *software de aplicación* para que éstos funcionen correctamente.

Aunque la mayoría estamos familiarizados con el uso de computadoras personales hay que tener en cuenta que el concepto de **sistema operativo** abarca a todos los sistemas informáticos (teléfonos móviles, televisores, servidores, electrodomesticos inteligentes, autos modernos, etc.)
.
Este sistema de software intermedio entre las aplicaciones que realmente quiere el usuario y el acceso a los recursos físicos, junto a la estandarización de las interconexiones, ha permitido el florecimiento de la industria del software ya que los desarrolladores de aplicaciones no necesitan controlar, ni siquiera conocer, el equipo físico que ejecuta su programa. Así se pueden desarrollar programas para la gestión empresarial, juegos o cualquier otro proyecto, sin preocuparse de qué tipo de disco duro, de qué acceso a memoria RAM hay disponible o de la velocidad del procesador.

Todas las interacciones con el equipo físico son gestionadas por el sistema operativo por lo que son casi transparentes para el equipo que desarrolla la aplicación funcional que utiliza el usuario. En los primeros sistemas (~1945) las máquinas eran operadas directamente por los programadores que conocían la estructura completa (física y lógica) de las grandes máquinas. Hacia 1955 los avances en componentes físicos fue el preludio de la aparición de los primeros programas que podían usarse con diferente hardware. Hasta finales de los años 1980 se precisaba de la actuación de un operador profesional que conociera el equipo físico utilizado para aprovecharlo con la máxima eficiencia.

Con la aparición de la informática personal (con el PC a la cabeza) se hizo necesario la creación de una capa intermedia entre las aplicaciones y los sistemas físicos para permitir que cualquier usuario, con un mínimo de adiestramiento, pudiera utilizar un ordenador.

Historia

La informática tal y como se le conoce hoy día, surgió a rfines de la II Guerra Mundial, en la década de los 40. En esos años no existía el concepto de **sistema operativo** y los programadores interactuaban directamente con el hardware de las computadoras trabajando en lenguaje máquina y en binario (programando únicamente con 0s y 1s).

El concepto de **sistema operativo** surge en la década de los 50. El primer **sistema operativo** de la historia fue creado en 1956 para un ordenador IBM 704, y básicamente lo único que hacía era comenzar la ejecución de un programa cuando el anterior terminaba.

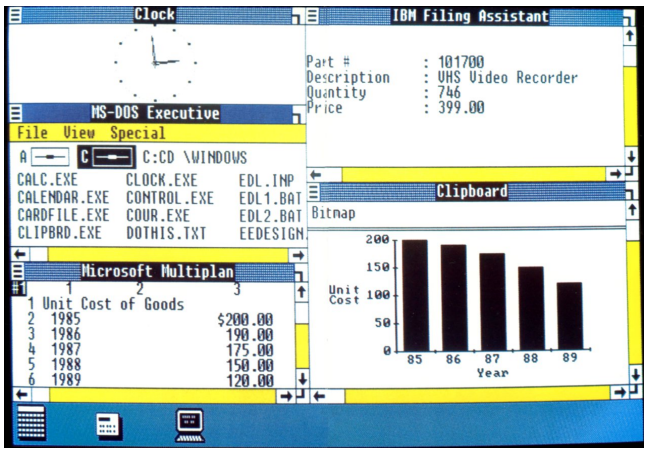
En los años 60 se produce una revolución en el campo de los **sistemas operativos**. Aparecen conceptos como sistema multitarea, sistema multiusuario, sistema multiprocesadores y sistema en tiempo real.

Es en esta década cuando aparece **UNIX**, la base de la gran mayoría de los Sistemas Operativos que existen hoy en día.

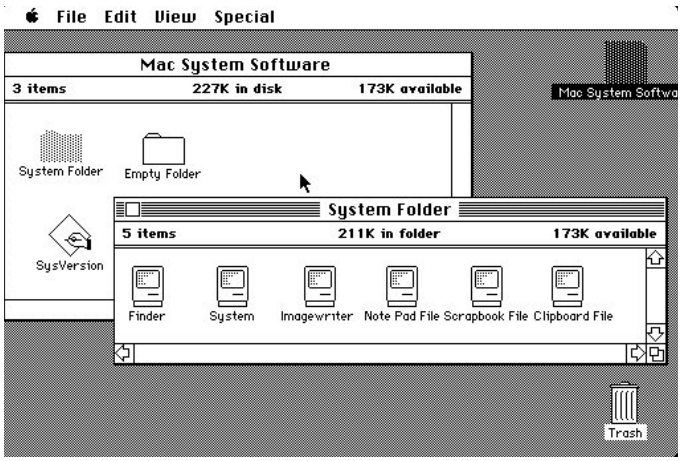
En los años 70 se produce un boom en cuestión de ordenadores personales, acercando estos al público general de manera impensable hasta entonces. Esto hace que se multiplique el desarrollo, creándose el lenguaje de programación **C** (diseñado específicamente para reescribir por completo el código **UNIX**).

Como consecuencia de este crecimiento exponencial de usuarios, la gran mayoría de ellos sin ningún conocimiento sobre lenguajes de bajo o alto nivel, hizo que en los años 80, la prioridad a la hora de diseñar un sistema operativo fuese la facilidad de uso, surgiendo así las primeras interfaces de usuario.

En los 80 nacieron sistemas como **MacOS**, **IBM OS2**, **MS-DOS** y **Windows**.



Windows 1.0 (1985)



Apple System 1 (1984)

En la década de los 90 hace su aparición **Linux**, publicándose la primera versión del núcleo en septiembre de 1991, que posteriormente se uniría al proyecto GNU, un sistema operativo completamente libre, similar a **UNIX**, al que le faltaba para funcionar un núcleo funcional. Hoy en día la mayoría de la gente conoce por **Linux** al **sistema operativo** que realmente se llama **GNU/Linux**.

Terminal Linux

Características

Multiusuario vs Monousuario

Los sistemas operativos monousuario son aquéllos que soportan a un usuario a la vez, sin importar el número de procesadores que tenga la computadora o el número de procesos o tareas que el usuario pueda ejecutar en un mismo instante de tiempo, siendo su uso comun en las computadoras personales primigenias.

Aunque un sistema monousuario se pueda conectar a otros sistemas por red, todavía es usado por un único usuario.

Se puede acceder al sistema por medio de un usuario único que tiene permiso para realizar cualquier operación. Este es el caso de los sistemas operativos más antiguos como MS-DOS y la serie **Windows 95/98/Me** de **Microsoft**, o **MacOS** (antes de **MacOS X**) de **Macintosh**. El usuario tiene acceso a todas las capacidades del sistema, pudiendo borrar o recunfigurar, incluso, información vital para su funcionamiento.

Un **sistema operativo** multiusuario permite el acceso concurrente de múltiples usuarios al computador. Este tipo de diseño debe ser capaz para manejar correctamente las tareas requeridas por los diferentes usuarios conectados al sistema.

A un sistema multiusuario puede accederse de forma local o remota, y de forma concurrente por varios usuarios. El sistema debe ser capaz de gestionar aquellos recursos privados de cada usuario asi como tambien los recursos compartidos de forma pública o en grupos de usuarios.

Con la aparición de los sistemas multiusuario surgen las jerarquias administrativas, en donde solo un determinado grupo de usuarios puede gestionar y configurar los diversos recursos disponibles, tales como quota de almacenamiento, comunicaciones, prioridad de ejecución de tareas, etc. A este grupo de usuarios especiales se los conoce como administradores.

En los **sistemas operativos** basados en **UNIX** por defecto se crea un usuario administrador principal llamado **root**, quien posteriormente podrá delegar tareas de administración en otros usuarios.

En los **sistemas operativos** basados en **MS Windows** el usuario administrador se llama **Administrador** o **Administrator** (versión en ingles), quien tambien posteriormente podrá delegar responsabilidades administrativas en otros usuarios.

Monotasking vs Multitasking

Los **sistemas operativos** pueden clasificarse también en monotasking o multitasking.
Monotasking son aquellos sistemas (primitivos) que solo pueden ejecutar una única tarea a la vez. Un ejemplo es **MS DOS**. Todos los sistemas operativos monotasking son monousuario.

Multitasking es la capacidad de un sistema operativo de realizar varias tareas al mismo tiempo. Todos los sistemas operativos multiusuario son multitasking.

Arquitectura

A continuación revisamos algunas de las componentes que debe incluir todo sistema operativo moderno.

El Núcleo

El núcleo (o **Kernel**) es la componente fundamental del sistema operativo que siempre está residente en la memoria real del computador. La función primordial del núcleo es operar los recursos físicos del ordenador.

La API del Núcleo

El conjunto de servicios que ofrece el núcleo a los procesos se denomina la **API del núcleo** (API, Interfaz de Programación de Aplicaciones). Está formada por procedimientos propios del núcleo, pero que se invocan desde un proceso cualquiera. La invocación de uno de estos procedimientos es una llamada al sistema. Ejemplos de llamadas al sistema en **UNIX** son:

- Manejo de Procesos:
  - creación (fork)
  - destrucción (kill)
  - término (exit)
  - sincronización (wait)
  - carga de un binario (exec)

- Manejo de memoria:
  - extensión de la memoria de datos (sbrk)

- Manejo de archivos y dispositivos:
  - open
  - read
  - write
  - close

Estas llamadas se implementan usualmente con una instrucción de máquina que provoca una interrupción. Esta interrupción hace que el procesador real invoque una rutina de atención perteneciente al núcleo y que ejecuta la llamada al sistema. Los argumentos de la llamada se pasan a través de los registros del procesador.

Controladores de Dispositivos

La operación de los dispositivos es altamente dependiente de su implementación. Es así como un disco SCSI se opera de una forma distinta de un disco IDE. Para independizar el código del núcleo de los variados mecanismos de interacción con los diversos dispositivos, el núcleo define clases de dispositivos. Ejemplos de clases son disco, cinta, puerta de comunicación, interfaz de red, etc. Para cada clase se define una interfaz estándar para interactuar con cualquier dispositivo que pertenezca a la clase. Esta interfaz corresponde a las declaraciones de un conjunto de procedimientos no implementados.

Un **controlador de dispositivo** o **driver** es el código que implementa una interfaz estándar para interactuar con un dispositivo específico, como por ejemplo un disco SCSI. Este código es por lo tanto altamente dependiente de los discos SCSI y no funcionará con discos IDE. Sin embargo, el núcleo interactúa con este driver para discos SCSI de la misma forma que lo hace con el driver para discos IDE, es decir a través de la misma interfaz.

La visión que tiene el núcleo de un disco a través de un driver es la de un arreglo de bloques de 512 o 1024 bytes de taman o fijo. El núcleo puede leer o escribir directamente cualquiera de estos bloques haciendo uso de la interfaz estándar de la clase disco.

Por otra parte, la visión que tiene el núcleo de una cinta es la de un conjunto de bloques de tamaño variable que sólo pueden leerse o grabarse de forma secuencial. También puede rebobinar esta cinta para volver a leerla o grabarla. Todo esto a través de la interfaz estándar de la clase cinta.

En **UNIX** una aplicación puede accesar una partición de un disco en su formato nativo abriendo por ejemplo */dev/sd0a*.

Es usual que los drivers estén siempre residentes en la memoria real y por lo tanto sean tratados como parte del núcleo. Sin embargo, en los sistemas operativos modernos, los drivers son módulos que se cargan dinámicamente si es necesario.

El sistema de archivos

El sistema de archivos es el componente del **sistema operativo** que estructura un disco en una jerarquía de directorios y archivos. Conceptualmente multiplexa un disco de tamaño fijo en una jerarquía de volúmenes de tamaño variable o archivos.

Dada esta equivalencia conceptual entre discos y archivos no es raro que ambos se manipulen con las mismas llamadas al sistema.

La responsabilidad de gestionar el sistema de archivos recae directamente en el propio núcleo.

El Shell

El shell se encarga de recibir las instrucciones interactivas de los usuarios y ejecutar las acciones que el usuario indique según su jerarquía lo permita. Usualmente, el intérprete de comandos es un componente más del sistema operativo y no forma parte del núcleo. El shell puede ser gráfico o basado en intérpretes de comandos. Por ejemplo **UNIX** ofrece varios intérpretes de comandos:

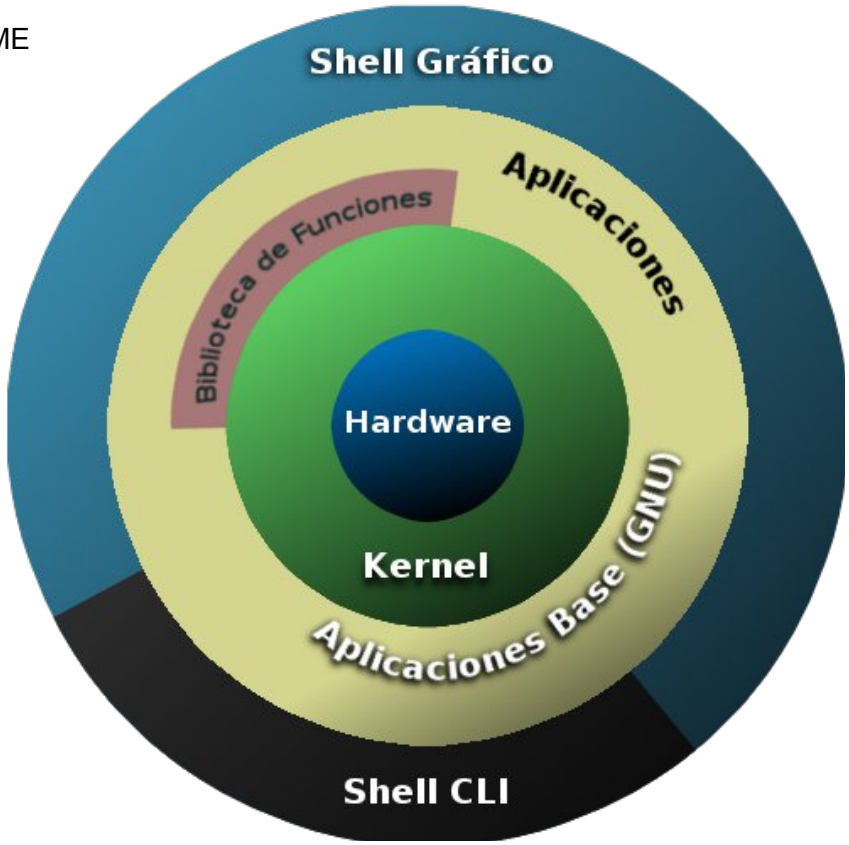
- sh
- bash
- csh
- zsh
- etc

MS-DOS ofrecía un único interprete de comandos: COMMAND.COM.
Los **sistemas operativos** modernos basados en **MS-Windows** cuentan con los interpretes de comandos CMD.EXE y POWERSHELL.EXE

A su vez, el entorno de escritorio de **MS-Windows** es en sí mismo un shell ya que permite la operación interactiva del sistema operativo.

En el caso de los sistemas basados en **UNIX**, es posible elegir entre una gran variedad de entornes de escritorios, siendo los más populares:

- KDE
- GNOME
- XFCE
- etc.



Arquitectura de SO Linux

Linux

GNU/Linux, también conocido como **Linux**, es un sistema operativo libre tipo **UNIX**, con características de multiplataforma, multiusuario y multitarea. El sistema es la combinación de varios proyectos, entre los cuales destacan **GNU** y el **núcleo Linux**.

Estructura del sistema de archivos

La estructura de directorios de **Linux** sigue todas las convenciones de **UNIX**, lo cual significa que tiene una dtribucion determinada, compatible y homogénea con el resto de los sistemas basados en **UNIX** (como **MacOS** por ejemplo). Al contrario que en **MS-Windows** o **MS-DOS** el sistema de archivos en cualquier **UNIX** no está ligado de una forma directa con la estructura del hardware, esto es, no depende de si un determinado ordenador tiene 1, 2 o 7 discos duros para crear las unidades C:\, D:\ o M:\

Toda la estructura de directorios **UNIX** tiene un origen único la raíz representada por **/**. Bajo este directorio se encuentran todos los ficheros a los que puede acceder el sistema operativo.

Estos ficheros se organizan en distintos subdirectorios cuya misión y nombre son estándar para todos los sistema **UNIX**. Algunos ejemplos son:

- / Raíz del sistema de archivos.
- /dev Contiene ficheros del sistema representando los dispositivos que estén físicamente instalados en el ordenador.
- /etc Este directorio esta reservado para los ficheros de configuración del sistema. En este directorio no debe aparecer ningún archivo binario (programas). Bajo este subdirectorio pueden existir otros subdirectorios que contengan más archivos de configuración específicos a diversos subsistemas.
- /lib Contiene las bibliotecas necesarias para que se ejecuten los programas que residen en
- /bin Contiene archivos binarios básicos de utilización común a todos los usuarios.
- /proc Contiene ficheros especiales que o bien reciben o envían información al kernel del sistema.
- /sbin Contiene programas que son únicamente accesibles al superusuario o root.
- /usr Contiene los programas de uso común para todos los usuarios.
- /var Este directorio contiene información vriable de los programas instalados.
- /tmp Espacio de almacenamiento temporal.

Vim

vim, del inglés **Vi Improved**, es una versión mejorada del editor de texto **Vi**, el cual, fue creado en 1976 por Bill Joy que tomó recursos de ed y ex, dos editores de texto para Unix. **Vim**, fue presentado en el año 1991 y desde entonces no ha dejado de experimentar infinidad de mejoras. La madurez de este editor de código es indiscutible, pues lleva desarrollándose más de 20 años.

La característica más destacable de este editor es su modo de edición modal.

Los diferentes modos de Vim

Modo Normal: Vim empieza en este modo. Se pueden emplear combinaciones de teclas para, por ejemplo, copiar líneas y trabajar en el formato del texto. Éste es el modo central, desde el que se cambia a los otros modos. Pulsando la tecla Escape siempre se puede volver al modo normal. Por defecto, podremos mover el cursor por el archivo con hjkl (izquierda, abajo, arriba y derecha respectivamente) o con las teclas de dirección.

Modo Insertar: Es el modo en el que podemos introducir texto. Se puede entrar a este modo desde el modo normal pulsando la tecla i. Existen otros comandos que nos llevarán al modo inserción pero se diferencian uno del otro por la acción que realizan, como cambiar una palabra dentro de unas comillas, cambiar el texto hasta el final de la línea o hasta el cierre de un corchete… etc. Un usuario avanzado ahorrará mucho tiempo utilizando estos atajos.

Modo de Comandos: Se accede pulsando ":". Permite introducir diferentes comandos, como buscar y reemplazar con expresiones regulares. También podremos personalizar aspectos de Vim para esa sesión, ya que no quedarán guardados los cambios permanentemente.

Modo Visual: Se entra pulsando la tecla v. Es como seleccionar texto con el cursor, solo que podremos escribir comandos para manipularlo.

Modo Selección: Se entra desde el modo visual pulsando Ctrl-G. Tiene un comportamiento similar al modo visual solo que al escribir no realizaremos comandos sino que reemplazaremos el texto, como en un editor de texto normal,

Ejercitación

1) Cree las siguiente carpetas en tu carpeta personal:

- Carpeta1
- Carpeta2
  - Carpeta21
- Carpeta3

2) Cree los siguientes ficheros dentro de cada carpeta:

- Carpeta1
  - Fichero1
- Carpeta2
  - Fichero2
  - Carpeta21
    - Fichero21
- Carpeta3
  - Fichero3

3) Visualizar el contenido de la carpeta personal en forma de lista.

4) Escriba “Hola soy un fichero” en el Fichero21.

5) Motrar por pantalla el contenido del Fichero21.

6) Borra la Carpeta3 y todo su contenido.

7) Posicionarse en la Carpeta21.

8) Verificar el directorio de trabajo actual.

9) Copiar la Carpeta1 y su contenido como Carpeta3.

10) Realizar una monografía de investigación respecto a los siguientes comandos:

- ls
- cp
- rm
- cat
- ssh

- cd
- clear
- pwd
- chmod
- chown
- find
- grep
- mkdir
- rmdir
- touch
- tar
- bg
- fg
- jobs
- top
- free
- su
- sudo
- vim
- tail
- ifconfig

Clasificarlos según su criterio. Ejemplos y casos de uso.