

CURSO PARA LA OBTENCIÓN DEL

DIPLOMA DE INFORMÁTICA MILITAR

(59130)

Teoría de Sistemas Operativos

**PORFOLIO**

Alumno: **Xavier Guerrero Fernández**

Fecha de cierre del porfolio: 31 de octubre de 2023

Un porfolio es una colección de documentos del trabajo del estudiante que exhibe su esfuerzo, progreso y logros. Es una forma de recopilar la información que demuestra las habilidades y logros de los estudiantes. Puede servir como forma de evaluación y de autoevaluación.

Índice

[**1.**](#_heading=h.30j0zll) **Capítulo 1 4**

[1.1.](#_heading=h.3dy6vkm) Preguntas 4

[1.2.](#_heading=h.1t3h5sf) Ejercicios 4

[**2.**](#_heading=h.4d34og8) **Capítulo 2 4**

[2.1.](#_heading=h.2s8eyo1) Preguntas 4

[2.2.](#_heading=h.17dp8vu) Ejercicios 4

[**3.**](#_heading=h.tyjcwt) **Referencias: 5**

# Capítulo 1

## Explicar el propósito principal de un sistema operativo

Los sistemas operativos existen para dos propósitos principales. Una es que está diseñado para asegurarse de que un sistema informático funcione bien mediante la gestión de sus actividades informáticas. Otra es que proporciona un entorno para el desarrollo y ejecución de programas.

El propósito principal de un sistema operativo es permitir al usuario entenderse con el Hardware, llevando a cabo la administración de manera eficiente del mismo y proporcionando al usuario herramientas para los programas de aplicación.

## Características de la vista de usuario del SO. ¿Qué herramientas utiliza el usuario en esta vista para enviar órdenes al sistema operativo? Algunas computadoras tienen poca o ninguna vista de usuario. Pon ejemplos

Características de la vista de usuario del SO:

* Desde una vista usuario se dispone del modo usuario para manejar un subconjunto de instrucciones del SO.
* La vista del usuario de la computadora varía según la interfaz que se utilice.
* Actúan de forma ergonómica, directa y sencilla para sacar el mejor partido de los recursos.
* Es el programa de interfaz de usuario, shell o GUI, entendido como el nivel más bajo de software en modo de usuario que permite al usuario iniciar otros programas.

Herramientas para enviar ordenas al SO: navegador web, un lector de correo electrónico o un reproductor de música. Estos programas también hacen un uso intensivo del sistema operativo.

Computadores con poco o ninguna vista de usuario: Microcontroladores empotrados en domótica y automóvil.

## Al parecer no hay una estricta separación entre los programas kernel y los programas de aplicación. Exponga cuatro tipos de programas que podíamos incluir en el SO:

Programas de ayuda, programas de administración, programas de entorno gráfico de gestión, y programas intérpretes de comandos.

## Comentar las peculiaridades de los SO en general en cuanto a su dimensionamiento en líneas de código, su prolongado ciclo de vida y su complejidad

Un SO implica un desarrollo de gran complejidad. Esto supone numerosas líneas de código, así como numerosos recursos materiales y humanos, que también se traducen en términos económicos. Por ello, su elaboración desde cero supone un despliegue de tiempo, presupuesto, medios y personas que no muchas empresas están dispuestas a asumir. Es por ello que un SO se mantiene modificando e implementando funcionalidades a medida que aparecen nuevas tecnologías para poder utilizar periféricos y protocolos de comunicación entre otros. Por ello, se concluye que, aunque estos sistemas operativos tengan en un momento dado un cambio en la arquitectura como el caso de Windows NT, el kernel a penas cambia con el paso del tiempo en muchos de los SSOO que conocemos a día de hoy.

## Modelo conceptual del sistema operativo en una perspectiva de arriba hacia abajo. Ídem de abajo hacia arriba.

## Aunque hay muchos profesionales de las ciencias de la computación, solo un pequeño porcentaje de ellos estará involucrado en la creación o modificación de un sistema operativo. Indique cuatro razones para estudiar la teoría de los sistemas operativos en este curso.

Obtener el máximo rendimiento de los recursos que ofrece el hardware y software, eludiendo la configuración en serie y adaptándolos a las necesidades de cada usuario.

Fortalecer la seguridad del propio SO, dificultando accesos no autorizados, ya que el SO instalado de serie ofrece vulnerabilidades.

Es una herramienta fundamental en el fortalecimiento de la seguridad de los servidores.

Permite crear un SO propio, partiendo de distribuciones existentes, aumentando las funcionalidades de protección frente a ataques.

## En 1968 Ken Thompson con una pequeña minicomputadora PDP-7 abandonada se propuso escribir una versión simplificada de MULTICS para un solo usuario. ¿Por qué tenía la capacidad de desarrollar un SO? ¿Qué importancia tuvo este hecho en la historia de los SO?

Ken Thomson trabajaba en los laboratorios Bell, desarrollando el SO MULTICS. Se considera el padre del lenguaje B en el que estaba escrito dicho SO y continuó desarrollando este sistema hasta que se abandonó debido a su creciente complejidad. En 1969, por su cuenta y junto a Dennis Ritchie trató de convertir el SO MULTICS abandonado en un sistema monousuario más simple y que tuviera mucho más rendimiento.

Este sistema se acabó transformado formalmente en UNIX en 1970 cuando consiguieron el apoyo económico de los laboratorios Bell. En 1972 decidieron reescribir completamente el código usando lenguaje C haciendo que se pudiera modificar fácilmente para trabajar en múltiples computadoras y convirtiéndose en el padre de una gran familia de sistemas operativos multisistema y multiusuario como System V, XENIX, BSD y LINUX, entre otros.

## Enunciado falso:

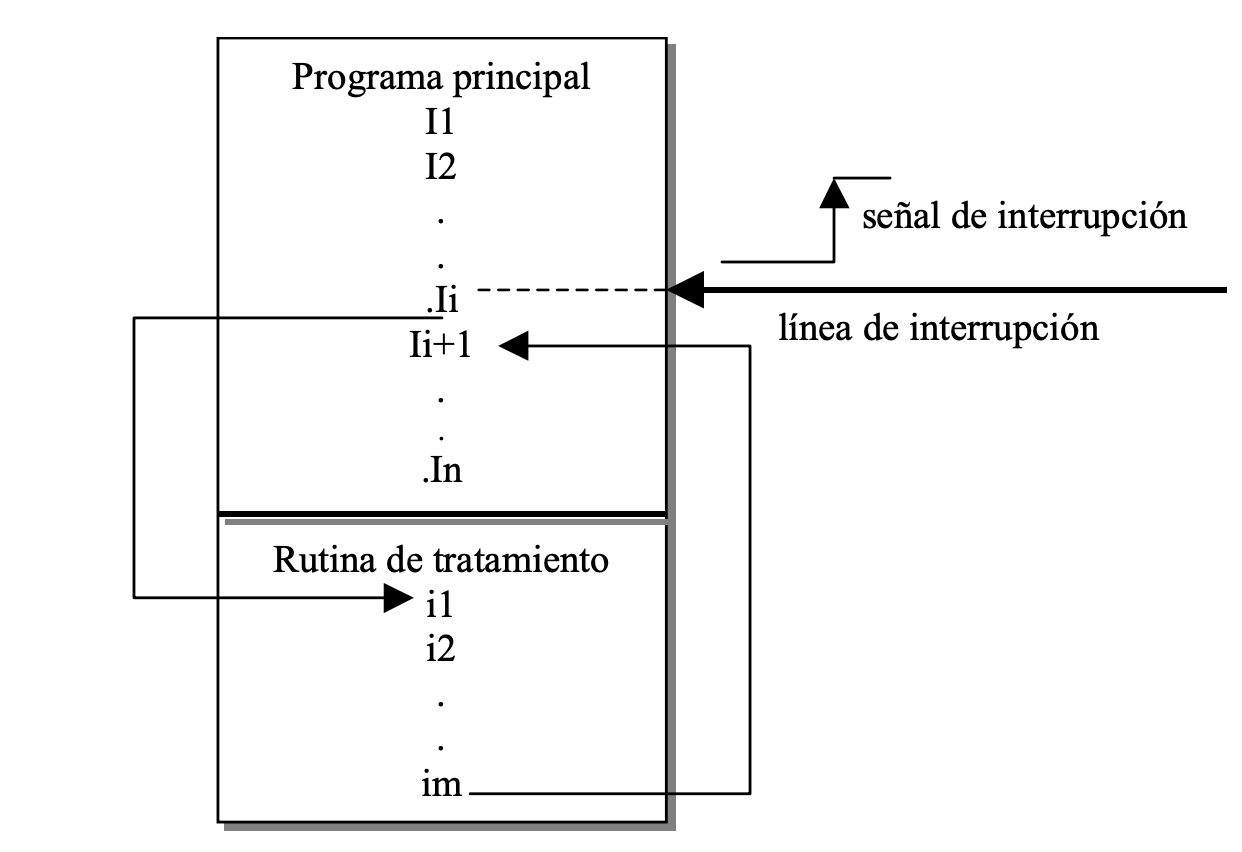
## “1960 Doug Engelbart en el Instituto de Investigación de Stanford inventó la interfaz gráfica de usuario, completa con ventanas, íconos, menús y ratón, encargado por Apple. Pero Steve Jobs, quien coinventó la computadora de Xerox PARC compró la interface gráfica de Engelbart para sus máquinas. En 1999, Apple adoptó un núcleo derivado de BSD UNIX que reemplazó al micronúcleo Mach de la Universidad Carnegie Mellon. Por eso el macOS de Apple no es un sistema operativo basado en UNIX”. Rectifique adecuadamente ese falso enunciado

“1960 Doug Engelbart en el Instituto de Investigación de Stanford inventó la interfaz gráfica de usuario, completa con ventanas, íconos, menús y ratón, encargado por Apple (NO FUE ENCARGADA POR APPLE, FUE POR XEROX PARCK). Pero Steve Jobs, quien coinventó la computadora de Xerox PARC (no coinventó la computadora de XEROC PARC) compró (NO LO COMPRÓ, CONSTRUYÓ UNO PROPIO) la interface gráfica de Engelbart para sus máquinas. En 1999, Apple adoptó un núcleo derivado de BSD UNIX que reemplazó al micronúcleo Mach de la Universidad Carnegie Mellon. Por eso el macOS de Apple no es (SI QUE ES) un sistema operativo basado en UNIX”.

## Explicar las diferencias claras entre driver de dispositivo y controlador de dispositivo.

Un controlador de dispositivo se encuentra en el propio dispositivo, es una parte de él, y tiene una parte de hardware y de programa (software). El driver de dispositivo, por el contrario, es un programa que sirve para comunicar ese dispositivo con el Sistema Operativo y que se pueda ejecutar.

## Mediante el esquema que se acompaña, explicar el concepto IRQ en el SO

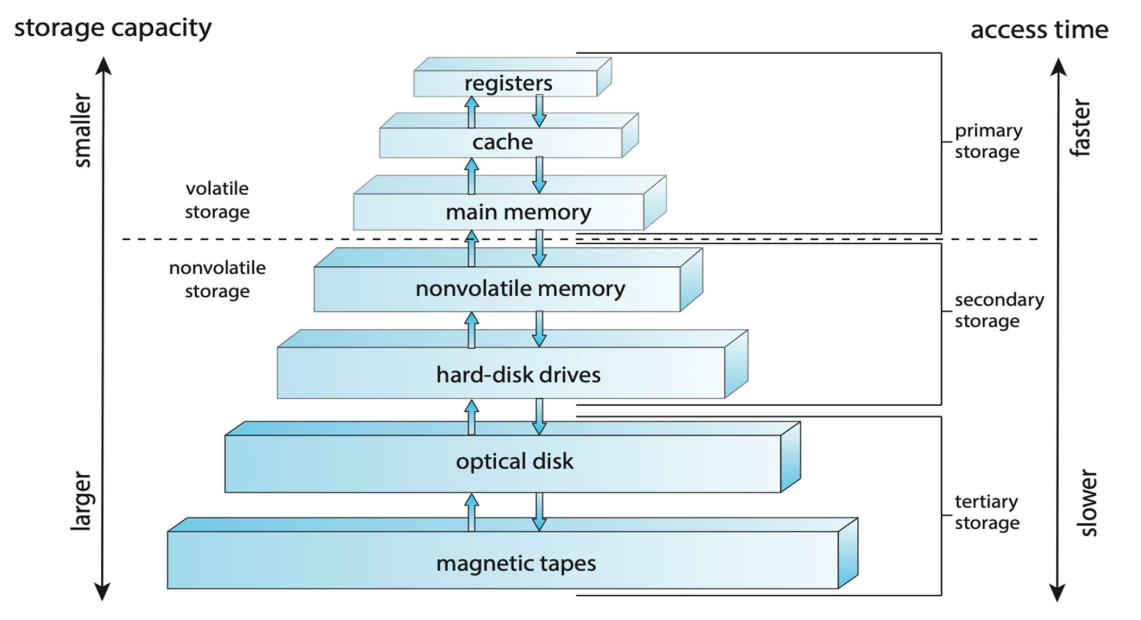


Una interrupción es una suspensión temporal de la ejecución de un proceso, para pasar a ejecutar una subrutina de servicio de interrupción, que, por lo general, no forma parte del programa, sino que pertenece al sistema operativo. Una vez finalizada dicha subrutina, se reanuda la ejecución del programa. Las interrupciones son generadas por los dispositivos periféricos habilitando una señal del CPU (llamada IRQ "interrupt request") para solicitar atención del mismo. Las interrupciones son una parte importante de la arquitectura de una computadora.

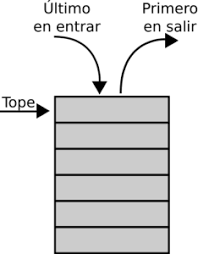
## Diferencia entre trampa (TRAP) e interrupción utilizando la siguiente tabla:

|  |  |  |
| --- | --- | --- |
|  | **Trampa** | **Interrupción** |
| **Método de Generación** | surge como evento durante la ejecución de una instrucción en el programa | generada como un evento por dispositivos de hardware |
| **Funcionalidad principal** | invoca la funcionalidad del SO quien transfiere su control al manipulador de trampas | activa el procesador, quien transfiere el evento al controlador de interrupciones |
| **Ocurrencia** | síncrona, tras la ejecución de una instrucción | asíncrona, no depende de la ejecución de instrucciones, al producirse por hardware no está directamente relacionada con una secuencia de instrucciones |
| **Sinónimos** | interrupción de software | interrupción de hardware |

## Respecto a la estructura de almacenamiento en niveles, sustituya las manchas rojas por los conceptos: rápido, lento, pequeño, grande, memoria volátil, memoria no volátil, almacenamiento primario, secundario y terciario

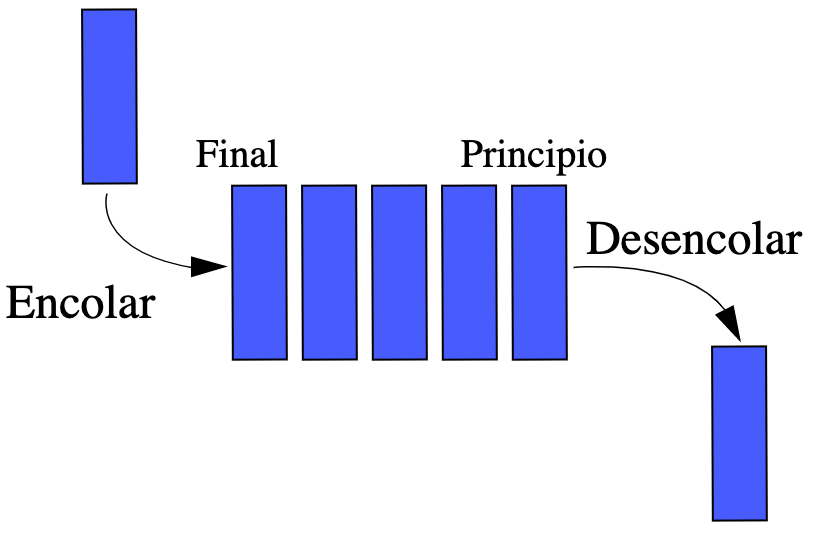


## Qué estructura de datos es la de la figura y ponga ejemplos donde el SO la emplea



Esta estructura de datos se trata de una pila ordenada secuencialmente y que utiliza el principio de último en entrar, primero en salir (LIFO). Un sistema operativo a menudo usa una pila cuando invoca llamadas a funciones. Los parámetros, las variables locales y la dirección de retorno se colocan en la pila cuando se llama a una función; regresar de la llamada a la función saca esos elementos de la pila.

## Qué estructura de datos es la de la figura y ponga ejemplos donde el SO la emplea



Es una estructura de datos ordenada en cola que utiliza el principio de primero en entrar, primero en salir (FIFO): los elementos se eliminan de una cola en el orden en que se insertaron.

## Un árbol rojo-negro es un árbol binario de búsqueda en el que cada nodo tiene un atributo de color cuyo valor es rojo o negro. Además de los requisitos impuestos a los árboles binarios de búsqueda convencionales, enuncie cuatro reglas para tener un árbol rojo-negro válido. Linux utiliza un árbol rojo-negro (rbtree) en muchos lugares del kernel, indique tres:

Además de los requisitos impuestos a los árboles binarios de búsqueda convencionales, se deben satisfacer las siguientes reglas para tener un árbol rojo-negro válido:

* Todo nodo es o bien rojo o bien negro.
* La raíz es negra.
* Todas las hojas (NULL) son negras.
* Todo nodo rojo debe tener dos nodos hijos negros.

Linux utiliza un árbol rojo-negro (rbtree) en muchos lugares del kernel, por ejemplo:

* El programador de E/S utilizan rbtree para realizar un seguimiento de las solicitudes.
* En el control de las unidades de datos en bloque de CD / DVD.
* El código del temporizador de alta resolución usa rbtree para organizar las solicitudes del temporizador.

# Capítulo 2:

**2. 1. ¿Cuál es el propósito del intérprete de comandos? ¿Por qué suele estar separado del núcleo?**

**2. 2. Ventajas de CLI**

Tiene un rendimiento más rápido: Si se sabe cómo usar CLI y se está familiarizado con diferentes comandos, el ciclo pregunta-respuesta se completará más rápido en comparación con el uso de GUI.

Consume menos memoria: CLI usa menos memoria en comparación con la GUI.

Se necesita un monitor de baja resolución para su uso a diferencia del GUI.

Un procesador lento puede funcionar: ya que no necesitan potencia de procesamiento adicional.

No necesita instalar complejos programas para gestionar entornos gráficos. Todos los SO importantes admiten CLI.

Autocompletar: Cuando escribe comandos en la CLI, la mayoría de los comandos se completan automáticamente al hacer clic en el botón TAB en el teclado.

Funciones avanzadas: En CLI se tiene posibilidades de acceso a más funciones avanzadas que en GUI.

Trabajo remoto: CLI es una opción ágil para comunicación entre dispositivos y operar diferentes comandos CLI en computadoras remotas.

Historial de comandos: Al ejecutar comandos en CLI, las órdenes se guardan en un recurso de la memoria. Al usar las flechas arriba/abajo del teclado se puede volver a usar esos comandos.

**(COPIADO LITERAL DE LOS APUNTES)**

**2. 3. Ventajas de GUI**

**2. 4. ¿Cuál es el propósito de las llamadas al sistema?**

**2. 5. Diferencias entre la llamada a una función en el dominio de aplicaciones de usuario y una llamada al sistema.**

**2. 6. Indique los seis tipos de llamadas al sistema que generalmente tienen los SO**

**2. 7. ¿Que es POSIX?**

POSIX ([acrónimo](https://es.wikipedia.org/wiki/Acr%C3%B3nimo) de Portable Operating System Interface, y X viene de [UNIX](https://es.wikipedia.org/wiki/UNIX) como seña de identidad de la [API](https://es.wikipedia.org/wiki/Application_Programming_Interface)) es una norma escrita por la [IEEE](https://es.wikipedia.org/wiki/IEEE), que define una interfaz estándar del [sistema operativo](https://es.wikipedia.org/wiki/Sistema_operativo) y el entorno, incluyendo un [intérprete de comandos](https://es.wikipedia.org/wiki/Int%C3%A9rprete_de_comandos) (o "shell").

El término fue sugerido por [Richard Stallman](https://es.wikipedia.org/wiki/Richard_Stallman) en la década de 1980, en respuesta a la demanda del [IEEE](https://es.wikipedia.org/wiki/IEEE), que buscaba un nombre fácil de recordar. La traducción del acrónimo es "Interfaz de Sistema Operativo Portable".

**(COPIADO LITERAL DE WIKIPEDIA, EN LOS APUNTES NO PONE NADA, SÓLO PONE: “API de POSIX para sistemas basados en POSIX”)**

**2. 8. ¿Cuáles son los tipos de SO según estructura de kernel? De una explicación de una línea de cada uno de ellos**

**2. 9. Coparación de un kernel monolítico de un microkernel?**

**2. 10. ¿Qué tipo de núcleo llevan los siguientes SO?**

|  |  |
| --- | --- |
| **Ubuntu** |  |
| **Windows 10** |  |
| **MacOS** |  |
| **MacOSX** |  |
| **Android** |  |
| **IOS** |  |

# Referencias:

Etc.